JCTA Journal of Computing Theories and Applications ISSN:3024-9104

Research Article

Exploring DQN-Based Reinforcement Learning in Autonomous Highway Navigation Performance Under High-Traffic Conditions

Sandy Nugroho[1], De Rosal Ignatius Moses Setiadi[1,*], and Hussain Md Mehedul Islam[2]

[1] Faculty of Computer Science, Dian Nuswatoro University, Semarang, Indonesia; e-mail: sandynugroho214@gmail.com, moses@dsn.dinus.ac.id
[2] Software Engineer, The Mathworks, Inc., United States; e-mail: mehadi.cuet@gmail.com
* Corresponding Author : De Rosal Ignatius Moses Setiadi

Abstract: Driving in a straight line is one of the fundamental tasks for autonomous vehicles, but it can become complex and challenging, especially when dealing with high-speed highways and dense traffic conditions. This research aims to explore the Deep-Q Networking (DQN) model, which is one of the reinforcement learning (RL) methods, in a highway environment. DQN was chosen due to its proficiency in handling complex data through integrated neural network approximations, making it capable of addressing high-complexity environments. DQN simulations were conducted across four scenarios, allowing the agent to operate at speeds ranging from 60 to nearly 100 km/h. The simulations featured a variable number of vehicles/obstacles, ranging from 20 to 80, and each simulation had a duration of 40 seconds within the Highway-Env simulator. Based on the test results, the DQN method exhibited excellent performance, achieving the highest reward value in the first scenario, 35.6117 out of a maximum of 40, and a success rate of 90.075%.

Keywords: Autonomous Highway Navigation; Autonomous Vehicle Navigation; Crowded Traffic Autonomous; Deep-Q Networking; Reinforcement Learning.

1. Introduction

Autonomous vehicles, commonly referred to as self-driving cars, represent a concept in which vehicles can operate without human intervention. Incorporating this technology into cars and other small vehicles can potentially reduce accident rates, energy consumption, and pollution levels. According to data presented by the World Health Organization (WHO) in December 2023[1], approximately 1.19 million people die each year due to accidents, with human error being a major contributing factor. The implementation of autonomy is believed to potentially reduce accident rates by up to 90% if properly executed[2]. The Society of Automotive Engineers (SAE) International has designed a classification system for autonomous vehicles consisting of six levels, with Level 0 denoting vehicles without automation systems, requiring human control, and Level 5 representing the highest level where vehicles can operate autonomously without human intervention and perform critical safety tasks accurately. At Level 4, vehicles are expected to perform all driving tasks and monitor the driving environment under specific conditions, eliminating the need for drivers to pay attention.

The driving tasks that autonomous vehicles must perform include various basic maneuvers such as straight-line driving, overtaking, lane changing, and entering lanes[3]. These tasks are relatively easier to execute in light traffic and at low speeds. However, special handling is required when implemented on highways with high relative speeds. In the past decade, reinforcement learning (RL) has emerged as a favored method for solving problems in various domains, including video games, robotics, and intelligent transportation systems[4]. RL is an approach in which artificial intelligence, known as an agent, operates in an environment, whether known or unknown, to adapt and learn based on the points provided[5]. These points can be positive, such as rewards, or negative, such as punishments. The agent decides its actions by considering the values it can obtain. The concept of rewards can be used as a

Received: January, 11th 2024
Revised: February, 12th 2024
Accepted: February, 13th 2024
Published: February, 13th 2024

performance assessment benchmark in reinforcement learning because higher reward values are directly proportional to the model's performance when executing a task[6].

RL can be a solution for addressing behavioral planning in the context of autonomous driving, where much of it relies on hard-coded rules like Finite State Machines[7]. Thus, agents in autonomous driving vehicles have very limited options and are confined to specific situations. Research in the field of autonomous vehicles is divided into several subcategories that align with the steps and conditions experienced by the agent, including intersection conditions[8], trajectory calculations[9], lane merging[10], and straight-line driving[11]. This research primarily focuses on straight-line driving because it is a fundamental driving maneuver commonly encountered on highways. Additionally, performance during high-traffic situations is also analyzed.

Methods commonly used in the RL domain are divided into two categories: model-free reinforcement learning (MFRL), including Deep-Q Networking (DQN), Soft Actor-Critic (SAC)[10] and Deep Reinforcement Learning (DRL) [12] and model-based reinforcement learning (MBRL), including Dyna-style Algorithm[13] and Model Predictive Control (MPC)[14]. MBRL is a model in which a cognitive map or model of the environment is formed, which describes how different "states" in the environment are connected to each other so that it can calculate the value that will be obtained from all the available steps[15]. However, MBRL is susceptible to errors if the provided environment is not well-structured. On the other hand, MFRL is a model that directly learns action values through trial and error, without explicitly constructing a model of the environment, and thus lacks explicit estimates of the probabilities governing state transitions[16]. Models can handle complex environments and adapt to different situations using these methods.

This research will employ the MFRL method with the DQN model. The choice of the DQN model as the primary approach in this research is based on careful consideration of various methods in the field of reinforcement learning. The first consideration is the advantage of DQN in performing calculations using an experience replay mechanism. This feature allows the model to reuse past experiences, enhancing learning efficiency and stability. In this regard, DQN outperforms some other MFRL methods that may be less efficient in utilizing prior experiences. Additionally, DQN has an advantage in processing complex data through integrated neural network approximations within the model. This enables DQN to handle environments with high levels of complexity, which may be challenging for some other MFRL methods[17]. However, it is acknowledged that there are weaknesses in previous methods, such as SAC, which may not be as efficient as DQN in utilizing prior experiences, and MBRL, which is prone to errors if the environment is not well-structured. In summary, this research contributes to analyzing the performance and simulation of the DQN model in the Highway-Env simulator environment [24], ranging from light to heavy traffic, to determine the maximum performance achievable by DQN.

2. Literature Review

2.1 Related Works

Several related studies were also reviewed before finalizing the research design. In a study [18], three models, named Policy Gradient (PG) models, Advantage Actor-Critic (A2C), and Dueling DQN (DDQN), were employed and compared for their performance. These models were trained using video feeds with RCNN under three conditions: straight roads during daylight, curved roads during daylight, and curved roads at night. The rewards were based on the model's ability to detect lanes and obstacles in the videos. Notably, these models were designed to replicate driving behavior and maneuvers using videos and The Open Racing Car Simulator (TORCS) environment, with a total of 1500 training episodes.

In another study [19], the DDQN model underwent modifications incorporating additional reward algorithms to regulate optimal actions. This model was trained in the Highway-Env environment with three lanes, where the model predominantly used the leftmost lane. The model received a reward of 0.8 for each step, and the training included 2000 episodes. Furthermore, this study quantified the model's maximum speed and distance traveled before potential collisions with other vehicles, demonstrating speeds of up to 36 m/s and covering distances of up to 3000 meters.

In a distinct study [20], the Deep Deterministic Policy Gradient (DDPG) algorithm was utilized with modified reward functions to attain an optimal reward value of 6000. The training consisted of 2000 episodes, and the model was subject to speed limits ranging from 17-40 m/s and acceleration limits of -2 to 2 m/s². The environment used was IPG CarMaker, and each training episode lasted for 5 minutes or until a collision occurred. The model also had an additional safety cage algorithm to control braking levels.

In a separate study [21], the Proximal Policy Optimization (PPO) algorithm was used with the AutomotiveDrivingModels environment. The study encompassed scenarios with vehicle quantities ranging from 1 to 100, with desired speeds maintained between 2 and 5 m/s. The reward value reached 120, with a timestep of 1,000,000 steps. Reward measurements included evaluating lane changes and the model's capability to change lanes behind other vehicles within specified time frames. In addition to reward-based evaluations, the study used a success rate criterion, where the model had to change lanes within a designated time and remain in that lane for at least 5 seconds without colliding.

Most of the studies mentioned above-utilized models falling under the MFRL category, but the testing was primarily based on the models' ability to achieve predefined rewards. However, MFRL offers other advantages, such as adaptation to changing conditions. This research employs the DQN model within the MFRL category, and the evaluation is based on the rewards obtained and the model's adaptability to various levels of traffic density.

2.2. Deep-Q Networking (DQN)

The Deep Q-Network (DQN) is a prominent MFRL algorithm that combines the Q-learning algorithm with deep neural networks [22]. DQN possesses essential features like experience replay and deep learning (DL). Experience replay involves the storage of an agent's experiences within a designated memory, enabling the reuse of these experience samples during training. This approach seeks to minimize the dependence on sequential experiences and enhance the efficiency of training data utilization. The aspect of DL is found in the Deep Neural Network (DNN), serving as an approximator for action-value functions.

Consequently, DQN can model complex relationships between states and Q-values, effectively addressing more intricate problems. In the domain of reinforcement learning, the concept of reward is a feedback value to the agent as an evaluation when making decisions[23]. Rewards are provided when the agent or model takes a specific action or step in a task. In this case, rewards are given when the agent can pass other vehicles and maintain the speed limit. Typically, rewards can be defined using the Equation (1).

$$r_t = R(s, a, s') \tag{1}$$

Where r_t represents the reward value obtained at each timestep, s and s' denotes the state in the environment, and a represents the action chosen by the agent. The transition involves the action that leads to a transition from the initial state (s) to the next state (s').

3. Proposed Method

This research aims to test the performance of the DQN model in a case study of straight roads on highways, and Figure 1 shows the research stages that will be carried out in this research. The research object used is the Highway-Env environment[24]. Next, arrangements were made for an environment with three-lane roads with the dominant lane on the right. Then, the duration is 40 seconds for each episode, and as obstacles, 20-80 vehicles are used, which will then be rendered simultaneously.

3.1. Model and Environment Preparation

At this stage, the first step is to set up the environment so that it can be understood and studied by the DQN agent. This setting involves two main aspects, namely the form of observation that the model uses and the action space available for the model. This observation takes the form of kinematics arranged in the form of a 2-dimensional array with size $V \times F$. Here, V refers to the number of vehicles observed, while F represents the number of features or attributes observed on each vehicle. To provide a clearer illustration, refer to Table 1.

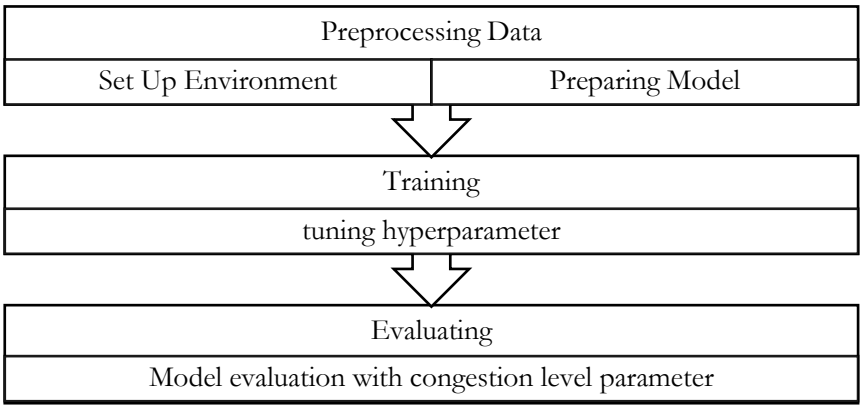

Figure 1. Research stages

Table 1. Examples of Agent Observations in the Environment.

Vehicle	Position $X(x)$	Position $Y(y)$	Velocity $X(v_x)$	Velocity $Y(v_y)$
Ego-vehicle	6.0	5.0	14.0	0
Vehicle 1	-11.0	5.0	12.0	0
Vehicle 2	12.0	7.0	11.5	0

Table 1 illustrates the kinematic observations used by the agent, formatted as a 2-dimensional array with dimensions $V \times F$. The observations include positions and velocities of vehicles within the simulation environment, with 'Ego-vehicle' indicating the vehicle is directly controlled by the agent. Where position $X(x)$ and $Y(y)$ are the vehicle position based on the x-axis and y-axis, respectively. Velocity $X(v_x)$ and $Y(v_y)$ is the vehicle velocity along the x-axis and y-axis, respectively. Ego-vehicle refers to a vehicle controlled by an agent.

Action space is an array consisting of a collection of actions that agents provided by the environment can take. Several action space options are available in a selected environment, such as continuous actions and discrete meta-actions. Continuous action encompasses the agent's ability to adjust steering angle and throttle control. However, these continuous action space options are deemed less optimal due to the limited control over crucial actions such as braking and maintaining speed. On the other hand, discrete meta-actions present a predefined set of actions for the agent to select from, including changing lanes to the left, changing lanes to the right, idling, increasing speed, and reducing speed. For this research, the discrete meta-action in the form of action space will be used primarily due to its advantages in facilitating rapid decision-making and expediting training processes [25].

Moreover, adjustments can be applied within the reward section, commonly referred to as reward shaping, The aim is to influence the agent's behavior, specifically in aspects such as the agent's intensity in overtaking and the agent's speed limit. Some common reward-shaping parameters are:

- 'collision reward': This parameter penalizes the agent with a negative reward in the event of collisions, thus incentivizing collision avoidance.
- 'right lane reward': This reward assigns a reward value to the agent when it chooses to move to the right lane. This parameter is particularly relevant in the context of highways, where the right lane is often used for overtaking other vehicles and typically maintains higher speeds.
- 'high-speed reward': This reward is intended to motivate the agent to use high-speed following the specified speed limit. The agent receives a positive reward when maintaining or exceeding the desired speed limit.
- 'lane change reward': This parameter incentivizes the agent to execute lane changes judiciously by providing a reward when the agent performs such maneuvers to overtake other vehicles.

- 'reward speed range': This reward sets the minimum and maximum speed limits allowed for the agent. The agent should strive to adhere to these speed limits, and rewards will be given if the agent stays within the desired speed range.

3.2. Model Training

The model used in this research is DQN based on MFRL. DQN incorporates several techniques to stabilize training with neural networks, such as replay buffers, target networks, and gradient cutting. Given its support for box-shaped observations and discrete-shaped actions, DQN aligns well with the selected environment.. Figure 2 shows the structure of reinforcement learning, namely the relationship between the agent and the environment. The agent will provide an action for each state the agent receives from the environment. The action given by the agent will be assessed, and the agent will be given feedback in the form of a reward as an evaluation value for the action chosen.

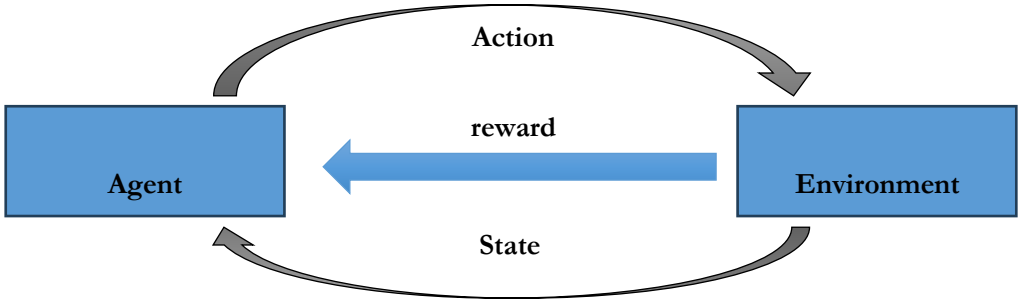

Figure 2. This Reinforcement Learning System Framework

Then hyperparameter tuning is carried out on the model, some of the hyperparameters that are set include:

- 'MlpPolicy' : Policy is a rule that guides agents to make decisions within a given environment. In the realm of reinforcement learning, policies are divided based on the type of observation environment, including MLPpolicy and CNNpolicy. MLPpolicy will be used in this research because it uses box type observations. MLP (multi-layer perceptron) is represented as a function that is useful for mapping state(s) to probability distributions(a) of actions taken by the agent. Using a deterministic policy, agents can choose certain actions without requiring probability calculations. So, the agent can provide optimal value in selecting actions.
- 'net_arch': This is an additional parameter in the DQN algorithm that functions to create a neural network that resembles deep learning.
- 'Buffer_size': This parameter sets the size of the replay buffer or the stored agent experience. Agents can use these saved experiences to speed up the training process. At the beginning of training, the 'learning_starts' parameter is trying various exploratory actions to understand the environment and provide experience to the replay buffer. So, the model does not experience stabilization problems and lacks experience data.
- 'batch_size': The batch size represents the number of data samples used in one training iteration, so it can speed up the training process.
- 'gamma': gamma is a parameter that determines the value for the discount factor. The discount factor functions to determine how important rewards obtained in the future are compared to rewards obtained now. Calculations are carried out using Equation (2).

$$\mathrm{Q}(s,a) = \mathrm{R} + \gamma \cdot \max_{a'} \mathrm{Q}(s',a') \tag{2}$$

Where

$\mathrm{Q}(s,a)$: Q values for state (s) and action (a).

R: reward received after taking action a in state s.

γ: Discount factor measures the degree to which an agent considers future rewards. γ ranges from 0 to 1, where 0 means the agent only considers current rewards, while one means the agent fully considers future rewards.

$\max_a Q(s', a')$: The maximum Q value for the next state (s') and all possible actions (a'). It describes the best-expected value that the agent can achieve in the next state.

3.3. Model Evaluation

The trained model was evaluated using the random sampling method. Evaluation criteria included the model's ability to accumulate reward value and its endurance within the environment. Rewards served as the primary metric for assessing model performance. Testing is carried out using the following parameters:

- 'n_eval_episodes': This parameter specifies the number of episodes for evaluation. The assessment was iterated across multiple runs to reduce bias. 'return_episode_rewards': This parameter sets the function to display a list of rewards obtained during the ongoing evaluation and the time the agent can survive the evaluation.
- 'Deterministic': This parameter sets the agent to map each state (s) directly into action (a) without using randomness calculation elements. This ensured that, under specific circumstances, the agent's actions remained consistent. The use of deterministic is intended if the task requires the agent to take actions with high precision.
  The obtained reward value is calculated using Equation (3).

$$R(s, a) = a\frac{v - v_{\min}}{v_{\max} - v_{\min}} - b \text{ collision} \quad (1)$$

Where v, $v_{\max}$, $v_{\min}$ which is agent speed, minimum agent speed, and maximum agent speed, respectively; a and b represent the coefficients of the amount of reward given when the environment is set.

In addition to the parameters mentioned, the assessment involves extracting the 'info' variable at every step the agent executes. Within the 'info' variable, q_values for each action are available, guiding the agent's decision-making process by prioritizing actions with the highest q_values. Furthermore, the 'info' variable encompasses the reward value assigned to the agent for each step, including rewards for collisions, adherence to a prioritized path, speed-related achievements, and maintaining lane position.

4. Results and Discussion

4.1. Preprocessing Environment

In the preprocessing stage, the environmental configuration is tailored to replicate highway conditions, particularly those characterized by high-density traffic as found in Indonesia. Furthermore, reward-shaping adjustments are also applied to regulate the model's behavior during training. Table 2 presents the configuration implemented in the environment.

Table 2. Environment Configuration.

Parameter	Value	Description
Action type	DiscreteMetaAction	Type of action to be used by the model
target_speeds	[11,19,27] (m/s)	Speed limits to be used by the agent
Lanes count	3	Number of lanes in the environment
Duration	40 (seconds)	Duration of one iteration of the environment
Vehicle count	20 until 80	Number of vehicles to be used as observations
Vehicle density	1	Number of vehicles generated as other vehicles based on the vehicle count parameter
Collision reward	-1	Reward obtained by the agent if it collides
Right lane reward	0.1	Reward obtained by the agent when it is in the right lane
High-speed reward	0.3	Reward is obtained when the agent is at the specified speed
Lane change reward	0.2	Reward obtained when the agent changes lanes
Reward speed range	[16, 27] (m/s)	Specified speed range that the agent must achieve

In the action type parameter, the actions used are discrete in the form of an array containing a list of actions that the model will learn. This environment has five actions in one array, namely [idle, increase speed, decrease speed, turn left, turn right]. The model selects its actions from this array. In a simulation lasting 40 seconds that incorporates 40 vehicles across three lanes, the traffic density effectively translates to a 'vehicle density' of 1, considering the multi-lane setup. This configuration aims to simulate realistic traffic conditions by evenly distributing the vehicles across the available lanes, as illustrated in Figure 4.

The traffic density serves as a metric for congestion level and is incorporated into the model's evaluation criteria. The environmental setup not only mirrors highway conditions but also incorporates reward shaping to guide the model's behavior toward optimal outcomes. The reward structure encompasses penalties for collisions, incentives for maintaining appropriate speeds, and bonuses for preferred lane usage, thereby aligning the model's actions with desired driving behaviors.The Collisions are assigned a negative reward, as they represent critical events that the model must learn to avoid. A modest positive reward value (0.1) is designated for traveling in the rightmost lane, encouraging the model to favor this lane. To ensure adherence to speed regulations, rewards are assigned for high-speed travel and lane changes, with a greater emphasis placed on maintaining and achieving the speed limit rather than on lane-switching maneuvers. The speed range for which the model receives rewards is set between 16 m/s (approximately 60 km/h, the minimum speed limit) and 27 m/s (approximately 97 km/h, the maximum speed limit), aligning with the speed restrictions of Indonesian toll roads. The high-speed reward value given is greater because, in the context of autonomous vehicles, the time aspect is the main priority, so the model must deliver passengers to their destination quickly.

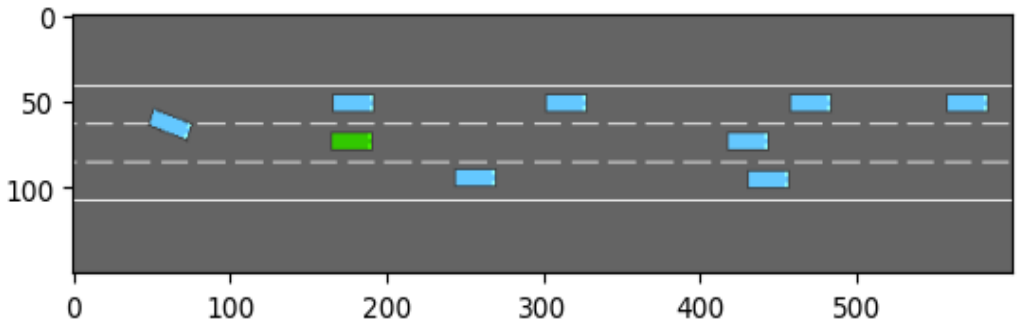

Figure 3. Example of the configured environment (40 vehicles).

4.2. Training Model

In the training phase, the model's performance is evaluated based on reward values, which are quantified as points reflecting the efficacy of actions taken by the agent within the simulation environment. These points are awarded based on specific criteria such as the agent's ability to avoid collisions, maintain optimal speeds, and adhere to preferred lanes, thereby aligning the model's learning objectives with desirable driving behaviors. The proposed model is trained using several hyperparameters that determine the model training process. The use of the MLPpolicy parameter at this stage is due to the type of observation in the form of a vector. The learning rate of the model is set to 0.0005, and the buffer size is set to 15000. Both buffer size and learning rate parameters significantly influence the stability of the training process.

The 'learning start' parameter represents the initial point at which the agent begins the learning process after conducting a certain number of exploration iterations in the environment. The value of 'learning start' determines how many exploration iterations are carried out before learning commences, measured in terms of iteration count. During this exploration phase, the agent gathers experiences that aid in the initialization process and help avoid instability in the early stages of learning. Learning begins at the 100^{th} iteration. The batch size is set to 32, while gamma is assigned a value of 0.7. The batch size represents the amount of memory available to the agent during training, accelerating the learning process. Gamma plays a crucial role in the model's short-term and long-term rewards decision-making. Table 3 displays the parameter values used during the model training.

Table 3. Parameter and Value for Models training.

Parameter	Value
Policy	MLP Policy
Learning rate	0.0005
Buffer size	15000
Learning starts	100
Batch size	32
gamma	0.7
Train frequency	1
Gradient steps	1
Target update interval	50
Verbose	1

The model is trained in 20,000 timesteps, and the training process is saved using the tensorboard logging feature, see Figure 4. The log captures vital parameters, including episode duration, accrued rewards, and training loss.

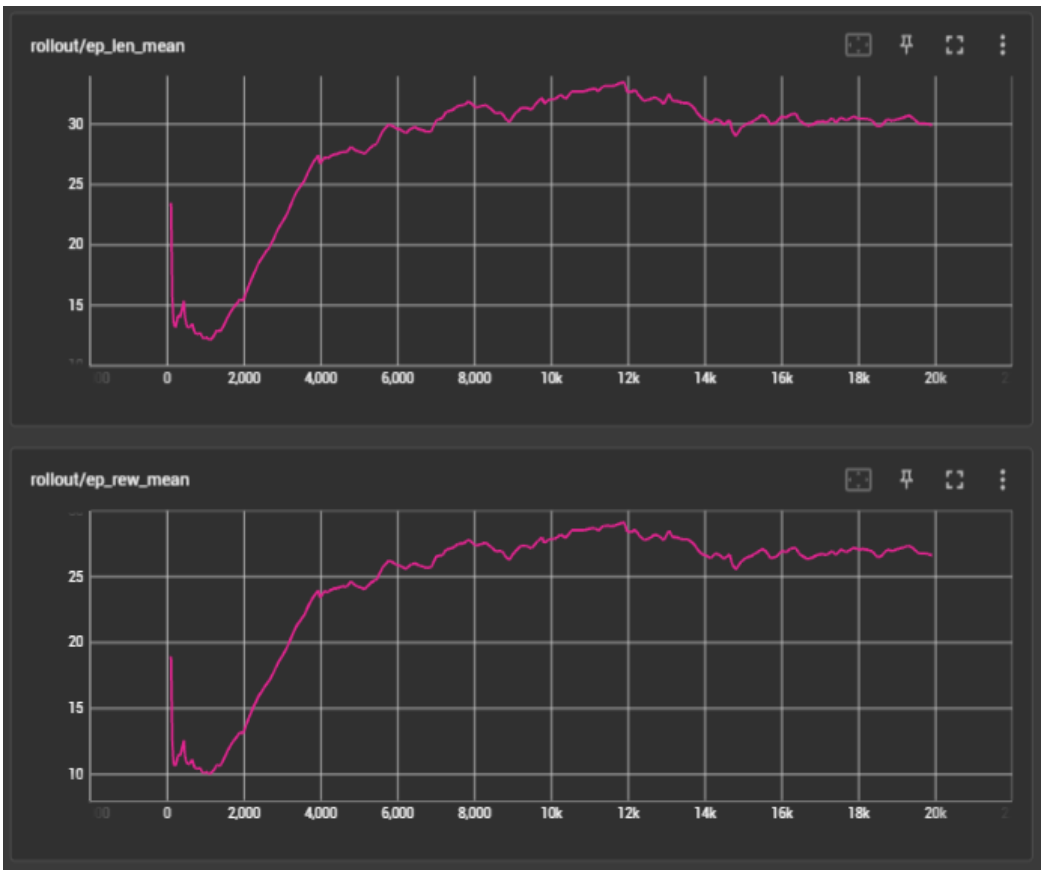

Figure 4. Tensorboard logging feature.

In the diagram stored in the logging file, the 'ep_len_mean' metric in the log charts the average duration of episodes throughout training. During the initial 0 to 2000 steps, the average model can operate for less than 15 seconds, with the rewards obtained being below a certain threshold value rather than time duration. Then, at steps 2000-4000, the model begins to understand and adapt to the environment so that in the diagram, the model can survive in the environment with a value of 14 seconds to 27 seconds, and the model can get a reward of 14 to 24 points. Up to timestep 8000, the model can adapt to the environment and maintain the reward obtained above 25 points for each iteration.

4.3. Evaluation Model

The trained model is assessed using the evaluate_policy function from the stable baseline library. The evaluation employs a random seed method, repeated 20 times, as specified by the n_eval_episodes parameter. Furthermore, the return_episode_reward parameter is set to true, allowing the display of both reward values and the duration taken by the agent for each iteration. The evaluation also employs the deterministic attribute, which is set to true. This means that the agent is obligated to map each action without introducing random elements, ensuring that the agent's decisions align with its predictions. The model undergoes evaluation four times, each involving an escalating vehicle density scenario. Tables 4, 5, 6, and 7 present the results of each scenario tested.

Table 4. First Scenario Results (using 20 /vehicles).

Test No.	Reward	Duration (Seconds)
1	39.8092	40
2	39.4899	40
3	39.6555	40
4	38.8546	40
5	12.9577	14
6	39.8794	40
7	37.9800	40
8	29.7185	31
9	24.2757	26
10	39.7398	40
11	13.7755	15
12	39.5981	40
13	38.9948	40
14	39.7179	40
15	39.5961	40
16	39.8471	40
17	39.9910	40
18	38.7287	40
19	39.7767	40
20	39.8481	40
Average	**35.6117**	**36.3 success rate (90.075%)**

In Table 4, in the first scenario where there are 20 vehicles, the model smoothly navigates past other cars, with most tests achieving a maximum duration of 40 seconds. The agent can maintain its maximum speed of 27 m/s, with an average reward value obtained for each step being 1. The success rate of the model is also still very high, namely 90.075%.

Table 5. Second Scenario Results (using 40 obstacles/vehicles).

Test No.	Reward	Duration (Seconds)
1	35.0472	40
2	25.9574	29
3	36.0744	40
4	36.3061	40
5	34.5682	40
6	37.3809	40
7	36.7206	40
8	36.9120	40
9	36.9202	40
10	37.7678	40
11	37.6310	40

12	35.4174	40
13	37.3294	40
14	37.1676	40
15	36.1953	40
16	13.1944	15
17	35.0669	40
18	34.9520	40
19	35.7802	40
20	32.7443	35
Average	**34.4567**	**37.95 success rate (94.875%)**

In Table 5, the number of vehicles is increased to 40, the model can still pass other cars smoothly, and the success rate is even better than the first scenario, namely 94.875%, and the duration reaches 37.95 seconds. However, there is a slight decrease in the average reward to 34.4567. In this scenario the agent is still able to accelerate to a maximum speed of 27m/s.

Table 7. Third Scenario Results (using 60 obstacles/vehicles).

Test No.	Reward	Duration (Seconds)
1	30.9687	40
2	31.8669	40
3	31.7298	40
4	31.9541	40
5	24.3245	32
6	10.5865	14
7	32.9552	40
8	32.8557	40
9	22.8038	30
10	30.6021	40
11	31.7992	40
12	30.7320	40
13	30.9346	39
14	17.1827	23
15	30.5704	40
16	24.1935	31
17	14.7175	20
18	5.4038	8
19	33.0354	40
20	31.3056	40
Average	**26.5261**	**33.85 success rate (84.625%)**

In Table 6, the number of vehicles increases to 60. The model can still pass other cars smoothly, but there is a noticeable decline in both the average duration, at 22.85 seconds, and the success rate, at 84.625%. The average reward also decreases to 26.5261. Despite this, the model maintains a high average speed of 21 m/s, which occasionally results in insufficient braking distance to prevent collisions.

Table 6. Third Scenario Results (using 80 obstacles/vehicles).

Test No.	Reward	Duration (Seconds)
1	30.4366	40
2	33.0515	40
3	5.4125	8
4	33.5156	40

5	32.7335	40
6	4.2284	6
7	7.6440	11
8	34.5674	40
9	31.3138	40
10	2.5056	4
11	3.5760	5
12	9.1004	12
13	28.9950	38
14	23.8914	31
15	3.7068	5
16	32.8471	40
17	12.1257	16
18	24.1116	30
19	0.9923	2
20	31.2931	40
Average	**19.3024**	**24.4 (success rate (61%)**

The fourth scenario, depicted in Table 7 with 80 vehicles, presents significant challenges for the model due to the limited space for maneuvers at high speeds, leading to frequent crashes. The average duration sharply falls to under 25 seconds, and the average reward drops below 20. Notably, in all scenarios, the model tends to prioritize actions that sustain high speed over braking and changing lanes, attributed to the higher rewards associated with speed.

4.4 Discussion

In this experiment, we used four scenarios representing different levels of traffic density on a straight road. The first scenario has 20 vehicles, and the number of vehicles increases gradually until it reaches 80 in the fourth scenario. The aim of using these four scenarios is to understand the extent of the model's adaptability to increasing traffic density on toll roads. Thus, the model was tested where the number of obstructing vehicles continued to increase until the model could no longer function effectively. The model's performance was evaluated based on the rewards obtained and the time duration for each iteration.

After going through training, the model can successfully pass scenarios 1 to 2 with an average reward above 30 out of a maximum total reward of 40 points. Suboptimal results were observed when vehicles spawned randomly in the simulation, appeared too close to the model, thus impeding its ability to avoid them effectively.

In the final scenario, where the number of vehicles reached 80, the model struggled to navigate the environment effectively., The diminished reward values and the considerable variation in the duration of each experiment evidenced this. The compromised performance is attributed to the excessive density of vehicles, which were randomly distributed within the simulation environment, constraining the model's ability to make effective maneuvering decisions. Furthermore, the vehicles were often positioned parallel to one another across all lanes, as depicted in Figure 5, forcing the model to decelerate to the minimum speed limit of 11 m/s.

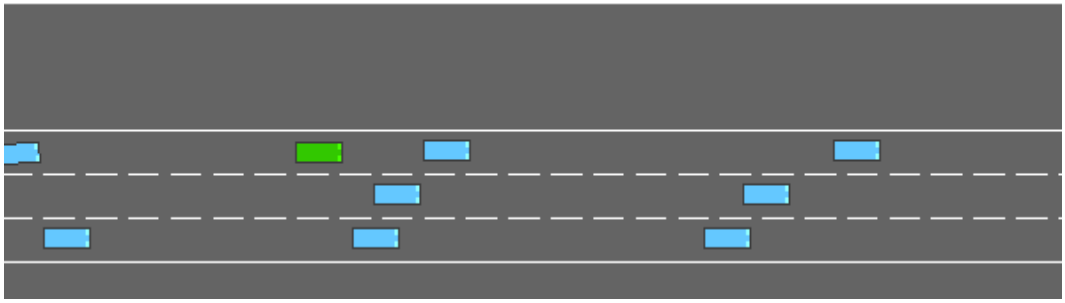

Figure 5. Example of road conditions in scenario 4

5. Conclusions

Based on the above evaluations, we can infer that the DQN model can effectively navigate environments with traffic densities of up to 60 vehicles, maintaining high speeds that conform to the standard regulations for toll roads (especially in Indonesia). However, when the number of vehicles reached 80, the model's performance decreased with shorter durations and some iterations with braking problems. These findings suggest a potential limitation of DQN model's ability to handle high traffic volumes at elevated speeds in simulated highway conditions. It is important to note that these results only reflect the model performance under simulation conditions to determine the performance limits of the DQN model facing high traffic at high speeds. The actual implementation of autonomous cars on real roads will involve more complex factors. Therefore, further testing and development of this DQN model must be carried out in more adaptive real-world situations. In further research, the environmental configuration can be carried out by providing a minimum and maximum speed model over a distance that is not too far away. Enhancements to the model's performance may also be achieved through refined reward shaping strategies or by optimizing hyperparameters such as gamma and batch size prior to training. Such advancements are anticipated to refine the model's decision-making capabilities, leveraging deep reinforcement learning techniques to adapt more effectively to diverse and dynamic driving environments.

Author Contributions: Conceptualization: S.N. and D.R.I.M.S.; methodology, S.N. and D.R.I.M.S.; software: S.N.; validation: S.N. and D.R.I.M.S.; formal analysis: S.N. and D.R.I.M.S.; investigation: S.N. and D.R.I.M.S.; resources: S.N.; data curation: X.X.; writing—original draft preparation: S.N.; writing—review and editing: D.R.I.M.S and H.M.M.I.; visualization: S.N.; supervision: D.R.I.M.S. and H.M.M.I.

Funding: This research received no external funding.

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

References

[1] World Health Organisation, "Road traffic injuries," 2023. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (accessed Jan. 30, 2024).

[2] M. Bertoncello and D. Wee, "Ten ways autonomous driving could redefine the automotive world," *McKinsey*, 2015. https://mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world (accessed Jan. 30, 2024).

[3] H. Detjen, S. Geisler, and S. Schneegass, "Maneuver-based Driving for Intervention in Autonomous Cars," in *CHI'19 Workshop on "Looking into the Future: Weaving the Threads of Vehicle Automation,"* 2020.

[4] S. Zhang, L. Wen, H. Peng, and H. E. Tseng, "Quick Learner Automated Vehicle Adapting its Roadmanship to Varying Traffic Cultures with Meta Reinforcement Learning," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, Sep. 2021, pp. 1745–1752. doi: 10.1109/ITSC48978.2021.9564972.

[5] A. Nandy and M. Biswas, *Reinforcement Learning*. Berkeley, CA: Apress, 2018. doi: 10.1007/978-1-4842-3285-9.

[6] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep Reinforcement Learning With Optimized Reward Functions for Robotic Trajectory Planning," *IEEE Access*, vol. 7, pp. 105669–105679, 2019, doi: 10.1109/ACCESS.2019.2932257.

[7] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016, doi: 10.1109/TIV.2016.2578706.

[8] E. Leurent and J. Mercat, "Social Attention for Autonomous Decision-Making in Dense Traffic," Nov. 2019, doi: 10.48550/arXiv.1911.12250.

[9] N. Carrara, E. Leurent, R. Laroche, T. Urvoy, O.-A. Maillard, and O. Pietquin, "Budgeted Reinforcement Learning in Continuous State Space," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 9299–9309. [Online]. Available: http://arxiv.org/abs/1903.01004

[10] B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 18808–18821, Oct. 2022, doi: 10.1109/TITS.2022.3160936.

[11] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road With Traffic and Driver Interaction," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 589–606, Sep. 2010, doi: 10.1109/TITS.2010.2046037.

[12] A. Kusari, "Assessing and Accelerating Coverage in Deep Reinforcement Learning," Dec. 2020, doi: 10.48550/arXiv.2012.00724.

[13] Y. Pan *et al.*, "Understanding and Mitigating the Limitations of Prioritized Experience Replay," Jul. 2020, doi: 10.48550/arXiv.2007.09569.

[14] E. Leurent, D. Efimov, and O.-A. Maillard, "Robust-Adaptive Interval Predictive Control for Linear Uncertain Systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, Dec. 2020, pp. 1429–1434. doi: 10.1109/CDC42340.2020.9304308.

[15] J. Gläscher, N. Daw, P. Dayan, and J. P. O'Doherty, "States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning," *Neuron*, vol. 66, no. 4, pp. 585–595, May 2010, doi: 10.1016/j.neuron.2010.04.016.
[16] N. D. Daw, Y. Niv, and P. Dayan, "Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control," *Nat. Neurosci.*, vol. 8, no. 12, pp. 1704–1711, Dec. 2005, doi: 10.1038/nn1560.
[17] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.5602
[18] A. Amballa, A. P., P. Sasmal, and S. Channappayya, "Discrete Control in Real-World Driving Environments using Deep Reinforcement Learning," Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.15920
[19] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang, and D. Cao, "Decision-Making Strategy on Highway for Autonomous Vehicles Using Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 177804–177814, 2020, doi: 10.1109/ACCESS.2020.3022755.
[20] S. Kuutti, R. Bowden, and S. Fallah, "Weakly Supervised Reinforcement Learning for Autonomous Highway Driving via Virtual Safety Cages," *Sensors*, vol. 21, no. 6, p. 2032, Mar. 2021, doi: 10.3390/s21062032.
[21] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in Dense Traffic with Model-Free Reinforcement Learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 5385–5392. doi: 10.1109/ICRA40945.2020.9197132.
[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.
[23] Y. Li, "Deep Reinforcement Learning," Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.06339
[24] E. Leurent, "An Environment for Autonomous Driving Decision-Making," *GitHub repository*. GitHub, 2018.
[25] G. Dulac-Arnold *et al.*, "Deep Reinforcement Learning in Large Discrete Action Spaces," Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.07679