



Article

Design of Obstacle Avoidance for Autonomous Vehicle using Deep Q-Network and CARLA Simulator

Wasinee Terapapattomakol ^{1,†,‡}, Danai Phaoharuhansa ^{2,*}, Pramote Koowattanasuchat [†] and Jartuwat Rajruangrabin [‡]

¹ wasinee.t@mail.kmutt.ac.th

² danai.pha@kmutt.ac.th

* Correspondence: jartuwat.raj@nstda.or.th, pramote.koo@nstda.or.th, danai.pha@kmutt.ac.th

[†] Department of Mechanical Engineering, King Mongkut's University of Technology Thonburi, Bangkok Thailand

[‡] Rail and Modern Transports Research Center, National Science and Technology Development Agency, Pathum Thani, Thailand

Abstract: This paper introduces deep Q-network (DQN) to avoid obstacle on road, which consists of DQN model setting, action setting, reward setting, and hypermeter setting. CARLA simulator is used to demonstrate the motion of the autonomous vehicle in virtual environment that the obstacle vehicle is parked on the road and the autonomous vehicle is riding on the road as shown in Figure 3. 3D Lidar is mounted on the vehicle. DQN model in the simulation is defined that the number of hidden layers and dense are 32x32x32 and rectified linear unit (ReLU) is used as activation function. The action setting is defined as acceleration pedal, brake pedal, and steering wheel. The actions are sent to torque control in transmission system of vehicle. The reward consists of main reward, penalty reward, and extra reward as shown in Equation (2, 4, 6) respectively. The hyperparameters are defined as Table 1. The result presents that the vehicle can brake to avoid the crash after the model is trained since 1400 training episode. It decides the action wavering between brake and acceleration. The control system can confidently decide the action when the model is trained over 7000 training episode. The vehicle obtains smoothly avoidance motion when the training episode reached 7,000 episodes. The reward condition has been designed from the hyperbolic function for continuous value. Therefore, the amount of the episode effects to the vehicle motion and the reward condition also impacts to the development of the DQN model. The shape of obstacle may disturbs the reward value and the reward may be swayed.

Keywords: Obstacle avoidance; Autonomous vehicle; Deep Q-network; CARLA simulator

1. Introduction

In recent years, the automotive industry has been still growing, and road traffic injuries in Thailand and sadly over 13,000 deaths and over million injuries each year [1]. Most of road traffic accidents are caused by human error. Autonomous vehicles have already been sold in the public market, which can assist drivers such as adaptive cruise control, automatic parking control and lane keep assist system.

To improve autonomous control system, one of the important issues is about the safety of the driver and passengers. Conventional control system is based on is a math model [2], but it can perform vehicle with limit situation. So that, the machine learning algorithm has been applied in autonomous systems [3,4] in order to control vehicle in various situations. Machine learning has two main technologies that are reinforcement learning and Deep Q learning. Both techniques have an advantage is the algorithm can learn by itself. Deep Q learning can use in huge sample spaces and more complex problems than reinforcement learning. Thus, the past few years, the autonomous vehicle has been tested using Deep Q learning technique [5].

Citation: Terapapattomakol, W.; Phaoharuhansa, D.; Rajruangrabin, J.; Koowattanasuchat, P. Title. *World Electr. Veh. J.* **2022**, *1*, 0. <https://doi.org/>

Received:

Accepted:

Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Submitted to *World Electr. Veh. J.* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Summary of Comments on Design of Obstacle Avoidance for Autonomous Vehicle using Deep Q-Network and CARLA Simulator

Page: 1

Author: Reviewer	Subject: Highlight	Date: 19/09/2022 20:38:26
Abstract should be written in a high level to describe, briefly and concisely, what your hypothesis/theory is, how you have done the research, what achievement you have got, ...		
It is not a summary of the details of the work, as the readers have done here. The kind of words such as "as shown in ..." should not appear in the abstract as that is the detail of the main text. Readers need to go to that page to know what you refer to. Often they will not.		
Suggest to rewrite the abstract. Authors could read other papers' abstracts to appreciate how to write an abstract properly.		
Author: Reviewer	Subject: Cross-Out	Date: 19/09/2022 20:32:05
This should not be in abstract.		
Author: Reviewer	Subject: Cross-Out	Date: 19/09/2022 20:30:31
Author: Reviewer	Subject: Cross-Out	Date: 19/09/2022 20:30:03
Author: Reviewer	Subject: Highlight	Date: 20/09/2022 09:48:26
In the introduction, authors should tell readers what you have done and achieved. SO what you have done, what are your achievements?		
Author: Reviewer	Subject: Highlight	Date: 20/09/2022 09:42:55
grammatical error		
Author: Reviewer	Subject: Highlight	Date: 20/09/2022 09:43:06
errors		
Author: Reviewer	Subject: Highlight	Date: 20/09/2022 09:44:47
Grammatical error: the such as clause does not math the subject "drivers".		
Author: Reviewer	Subject: Highlight	Date: 20/09/2022 09:45:21
grammatical error.		
Author: Reviewer	Subject: Highlight	Date: 20/09/2022 09:46:23
awkward sentence.		

To experiment, they have to test on real traffic roads, which obtains a lot of costs to prepare the experiment such as environment control and measurement system. So that, the simulation software was greatly developed to simulate the vehicle motion in virtual environment before experimenting with actual conditions.

Carla simulator is an open-source software and it is introduced in several papers [4,6,7] that has been developed from the ground up to support the development, training, and validation of autonomous driving systems by simulating the response of vehicle in virtual situation. The advantages are that the program obtains the physic engine, and real-world map, which have many assets to create the experiment condition such as any type of vehicle, traffic attribute, onboard sensor, and environment that prepare for an autonomous system. It supports flexible specification of sensor suites, virtual environmental conditions, full control of all static and dynamic actors, maps generation, and much more by using python code for controlling.

The main motivation of this study is to reduce traffic fatality rates and accidents on the road by testing out the proposed method in the virtual environment to validate the accuracy of the technique to improve the autonomous driving system through the simulation program and then apply it in onsite experiments in the further work.

2. Autonomous vehicle

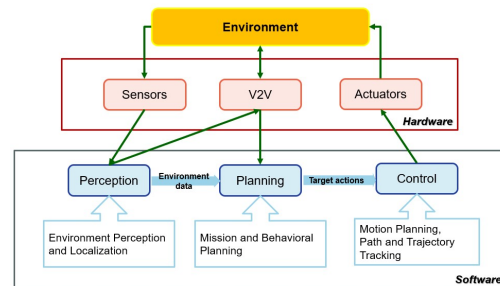


Figure 1. Architecture of autonomous driving system.

Autonomous vehicle technology is evolving rapidly with the continuous improvement of artificial intelligence technology. It can perform on highways with the exact route path [9]. Autonomous driving systems consists of 2 main parts [6] as shown in Figure 1 that which consists of hardware and software[8]. The hardware part obtains the mechanical components such as sensors, Vehicle to Vehicle (V2V) hardware, and actuators. Sensors are used to observe the environment. V2V hardware is set up on each vehicle for communication and sharing data between the vehicles, and actuators are used to power the subsystem in the vehicle such as powertrain, steer wheel, and brake. The software models contain perception, path planning, and control system [6]. The perception module collects data from the sensors and builds 3D mapping of environment around the vehicle that makes cars understand the environment as shown in Figure 2 then, path planning design based on the user's command. Finally, the control system will order action to vehicle hardware by observing the actual environment.

This page contains no comments

The onboard sensor consists of several sensors, which is famous device for autonomous vehicles application such as camera, radar, and LiDAR. Camera sensors in automated driving systems have 2 concern points that sampling rate of image and visibility conditions.

Radar technology uses high-frequency electromagnetic waves to measure the distance to objects based on the round-trip time principle. It is independence with light and weather conditions. It can measure up to 250 m in very adverse conditions.

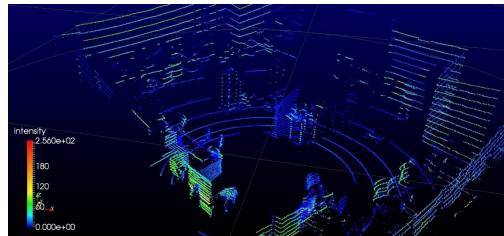
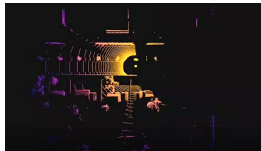


Figure 2. Example of point cloud output from a LiDAR.

Light Detection and Ranging (LiDAR) is an active-ranging technology that calculates the distance to objects by measuring the round-trip time of a laser light pulse [10]. Laser beams have a low divergence for reducing power decay and the measured distance is up to 200 m under direct sunlight. The 3D point cloud is shown in Figure 2. It represents 3D visualization of environment that it is generated by the information from LiDAR [11].

3. Carla simulator



(a) The LiDAR information



(b) Third person visualization

Figure 3. CARLA simulator presents by LiDAR information and 3D visualization.

CARLA (Car Learning to Act) [14,15] is an open-source software for urban driving environment that it has been developed in order to simulate the vehicle and vehicle control system, which supports training, prototyping, and validation of autonomous driving model [12]. The simulator includes the visualization relative to vision of sensor and vision of third-person as shown in Figure 3. It can be separated the process to apply with autonomous vehicle control simulation as 3 sections as shown in Figure 4 and it consists of virtual environment, agent, and control system.

This page contains no comments

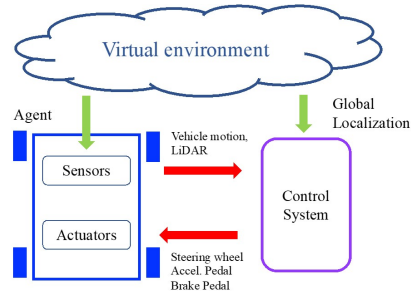


Figure 4. CARLA architecture for autonomous vehicle simulation

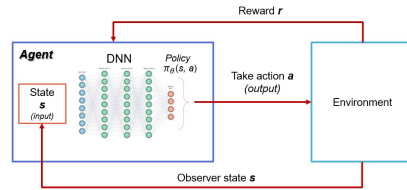


Figure 5. Deep Q-Learning process

3.1. Virtual environment

The virtual environment is presented as 3D models, which consists of static objects and dynamic objects. Static objects are such as road, buildings, traffic signs, and infrastructure, and dynamic objects denote the moving object in the environment such as pedestrians and other vehicles. All models are designed to reconcile visual quality and rendering speed. All environment models can be defined the dimension as same as real objects. They are presented in 3D visualization as shown in Figure 3.

3.2. Agent

The agent means that the object acts relative with control system such as pedestrian and vehicle. The agent is determined as vehicle and Tesla model 3 is represented instead of autonomous vehicle as shown in Figure 3b, dynamical model of which is included in CARLA. The agent consists of sensors and actuators in the vehicle. The sensor in the program obtains several kinds of sensor such as GPS, speedometer, IMU, LiDAR, camera, etc. The output of sensor can be applied the noise instead of the actual noise such as fog, rain, and smoke. The actuators denote steering wheel brake and traction motor.

In this study, LiDAR is added with conventional sensors and it is used to observe the obstacles and the environment. The attributes can be set the sensor parameters as Velodyne HDL-32E [14,16] and it obtains 32 channels and the sensor is mount in front of the vehicle. The range of detection is set at 40 meters. The vehicle motion is performed by steering

This page contains no comments

wheel, acceleration pedal, and brake pedal. The steering wheel can control the steering angle, which is between ± 70 degrees. The acceleration and brake pedals can be assigned from 0 to 100 percent.

3.3. Control system

The control system can be used feedback control or others. in this study is designed based on DQN algorithms, which is a kind of reinforcement learning. It consists of path planning, obstacle detection and DQN algorithm. The path planning is determined on the straight road. DQN algorithm received sensor data and it will determine the action to control the actuators.

4. Deep Q-network control

Deep Q-network is a kind of reinforcement learning, which applies to control the agent vehicle and it will be used to assign actuators such as acceleration pedal, brake pedal, and steering angle. It requires to define DQN model, training, and reward. Deep Q-learning can work well with complex decisions and a huge of data such as the large state input, the elaborate environment, and the delicate action.

Figure 5 presents that the process of Deep Q-learning that uses a neural network to approximate the Q-value function. The state is given as the input, and possible actions are generated as the output. DQN model will compute the reward to choose the action for a given state and store its experiences in a memory buffer. Then, the model will use the experiences from the memory buffer to train the target model, which calculates the best way to get more reward.

To train the DQN algorithm, the best reward and the state are collected in memory buffer as shown in Figure 6. Then, this section will be separated 4 parts such as model setting, action setting, reward setting, and hypermeter setting. These four parts are very important in enabling the DQN model to learn correctly and appropriately for the experiments. The details of designing the control system are listed are as below.

4.1. Model setting

DQN model consists of dense and layer, which has nodes in the neural network form. Each layer obtains the activation function, which matches the type of data has been designed by the user. If the user doesn't config it, the activation function will be designated as linear function for default.

For selecting the activation function, it must be considered the problem of vanishing and the gradient explosion. The less complex neural network with Relu activation is initiated and the amount of dense and layer are increased to achieve the good results. They are defined as shown in Table 1 and The dense layers are given that 32 layers, 32 dense per layer, and 32 nodes per dense.

4.2. Action setting

In the action setting, this study defines 2 actions such as straight forward and brake. The steering wheel are bounded at ± 35 degrees. The acceleration and brake pedals are acted relative to DQN algorithm consideration.

4.3. Reward setting

Reward setting consists of 3 rewards such as main reward, penalty reward, and extra reward. They should be set as continuous function in order to make the algorithm learn in the correct and smooth direction. The hyperbolic function has been used in different weights for each reward condition. If the agent can success a good solution the reward will

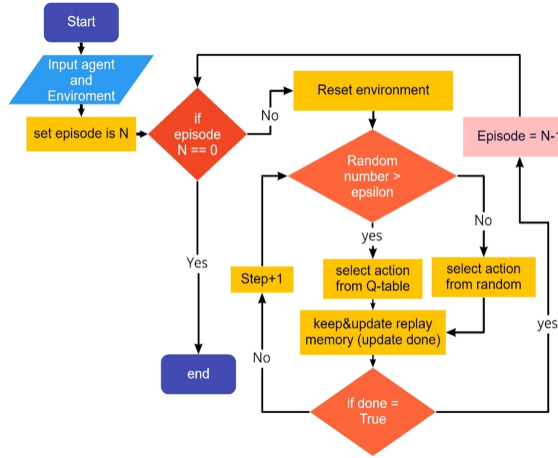


Figure 6. Process to select the action in DQN model

be given as double. If the agent did a bad solution the reward will be minus. The function of each reward condition effect to training period. Then, total reward (R_T) is given as

$$R_T = R_m + R_p + R_{ex} \quad (1)$$

Main reward (R_m) is designed relative with the distance from vehicle to destination (d_m) as shown in Equation (2). The reward will be highest when the vehicle arrived the desired position.

$$R_m = -2.00(d_m - 98.00)^2 + 3,000 \quad (2)$$

$$d_m = \sqrt{(x_d - x_v)^2 + (y_d - y_v)^2} \quad (3)$$

where, R_m denotes main reward. d_m denotes distance from vehicle to destination. x_d and y_d denotes desired position in X and Y axis. x_v and y_v denotes vehicle's position in X and Y axis.

Penalty reward (R_p) is computed by distance from LiDAR as shown in Equation (4). It depends on the minimum of vehicle and obstacle when the obstacle can be detected within the area that distance ± 3 meters in lateral axis, distance from road to 1.3 meters in vertical axis, and distance 0 to 70 meters in longitudinal axis. The equation is designed as fourth degree polynomial equation. It restrains the motion to destination and avoids the obstacle.

$$R_p = -0.0011d_l^4 + 0.15d_l^3 - 6.60d_l^2 + 145.00d_l - 3,040.00 \quad (4)$$

$$d_l = \min \sqrt{x_o^2 + y_o^2 + z_o^2} \quad (5)$$

This page contains no comments

where, x_o , y_o , and z_o denote nearest detected position, which is scalar of distance vector from vehicle to obstacle.

Extra reward (R_{ex}) is the special rewards, which is added to total reward when the agent can avoid the collision within the episode. This rewards will be added when the vehicle can brake within 10 meters from obstacle. The value of the rewards has been effective with the choice for making decisions. The reward is given as

$$R_{ex} = -200.00x_l + 2,200.00 \quad (6)$$

4.4. Hyperparameter setting

Table 1. List of hyperparameters and their values.

Parameters	Values	Descriptions
Number of hidden layers and dense	32x32x32	The number of nodes in the neuron network. In this case, are 3 hidden layers and 32 dense of each.
Activation function	ReLu	Function which defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.
Replay memory size	1,000,000	SGD updates are sampled from this number of most recent steps.
Target network update frequency	400	The frequency (measured in the number of parameter updates) with which the target network is updated.
Discount factor	0.95	Discount factor gamma used in the Q-learning update.
Learning rate	0.001	Tuning parameter that determines the step size at each iteration while moving toward a minimum of a loss function.
Initial exploration	1	Value of ϵ in ϵ -greedy exploration.
Final exploration	0.0001	Data
Final exploration step	1,840 episodes (600,000 steps)	The number of steps over which the initial value of ϵ is linearly annealed to its final value.
Replay start size	4,000	A uniform random policy is run for this number of steps before learning starts and the resulting experience is used to populate the replay memory.
Input data size	6	The number of the data that given as input in Q-network
Output data size	4	The number of the action that given as output in Q-network

The hyperparameter setting is separated into 2 parts relative to network structure and training algorithm. They are defined as shown in Table 1 and the hyperparameters for network structure are the number of hidden layers and units, dropout, network weight initialization, and activation function. The hyperparameters for training algorithms are

This page contains no comments

learning rate, number of epochs, and batch size. ϵ -greedy exploration is an exploration strategy in reinforcement learning that takes an exploratory action with probability and a greedy action [13].

5. Case scenario

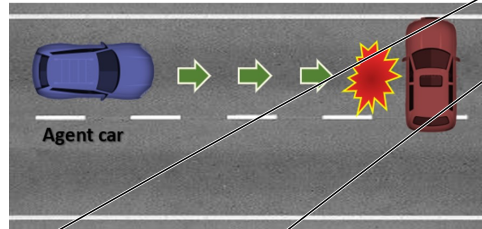


Figure 7. The scenario obstacle avoidance in low-speed control.

Thailand is one of the countries that have the most injuries and fatalities in car accidents in the world. The main cause is that the driver cannot avoid the obstacle in time and about 20,000 people dies in road accidents each year.

One of highest accident case in Thailand is that the agent vehicle performs on the main road and the obstacle vehicle drives across with the main road as shown in Figure 7. The agent vehicle cannot brake or avoid in the time, which has challenge to design the function for avoiding the avoid the obstacle vehicle using DQN algorithm.

Therefore, the case scenario in this study is defined that the initial velocity of the agent car is 40 km/h and the destination is at 100 meters. The front wheel's angle is bounded at ± 35 degrees. Then, the agent vehicle is driven to the destination and the obstacle is placed on the trajectory. The agent vehicle should learn to brake or avoid without the crash.

6. Simulation result

According to the case scenario, the parameters are configured as Table 1. DQN model has been developed relative to the number of training episodes. The results as shown in Figure 9 and Figure 10 present distance from vehicle to destination, velocity, and rewards. The motion will be presented by 3D visualization in CARLA simulator as shown in shown in Figure 8.

Considering distance and velocity of agent vehicle in Figure 9, the distance from the vehicle to destination is shown in Figure 9a. The training episode is increased, the final position of vehicle becomes nearly to the destination. The decision of control system may be switching between stop and forward, after it stops the motion as shown in blue line in Figure 9. It presents the vehicle motion for DQN model at 1400 training episode. The other cases the vehicle will permanently stop. Therefore, the decision of control system using 1400 training episode may not enough to use as obstacle avoiding because some nodes in DQN model may still blank node so that the control system may be finding the better path, which tries to move forward after the vehicle stop.

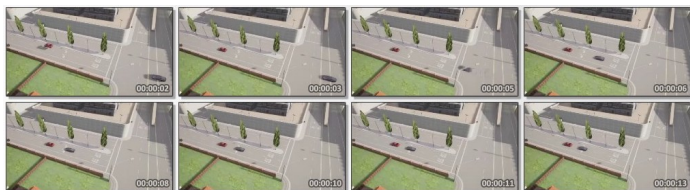
The velocity profile of each case is shown in Figure 9b. The initial velocity in all cases perform about 40 km/h and the velocity using DQN model at 1400 and 5000 training episode will decrease with low deceleration after that the deceleration will increase to zero velocity. The velocity using DQN model at 7000 and 8000 training episode almost decrease with constant deceleration to zero velocity.

Page: 8

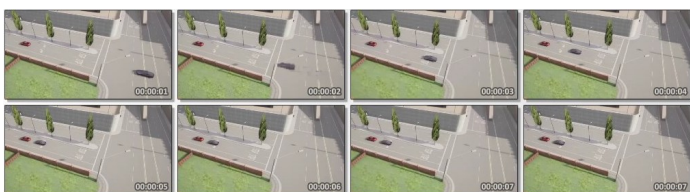
Author: Reviewer Subject: Highlight Date: 20/09/2022 09:57:03
need reference to support this statement.

Author: Reviewer Subject: Highlight Date: 20/09/2022 09:58:34
use plural form of case.

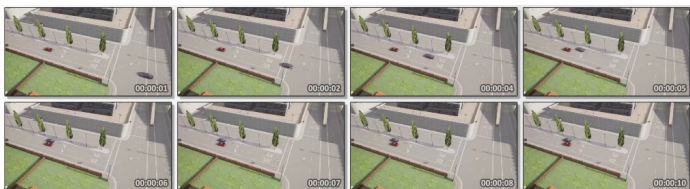
This page contains no comments



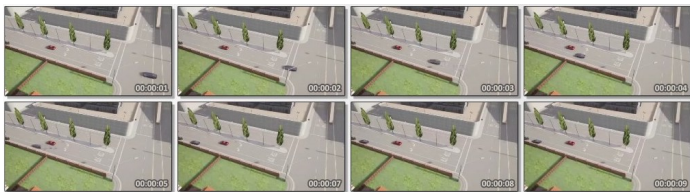
(a) 1400 training episodes



(b) 5000 training episodes

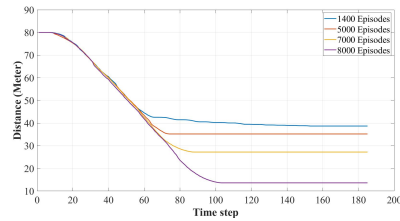


(c) 7000 training episodes

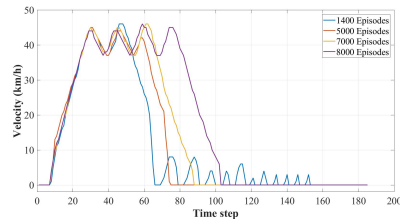


(d) 8000 training episodes

Figure 8. The agent performs using DQN model at 1400, 5000, 7000, and 8000 training episodes.



(a) Distance from agent to destination.



(b) Velocity of agent

Figure 9. The motion of the agent relative to the number of training episode for straight motion test.

Figure 10 illustrates the reward during the motion from initial position to goal. The rewards increase when the vehicle becomes to destination but the rewards for DQN model at 1400, 5000, and 8000 training episode increased to the convergent values and the oscillation of rewards may be occurred by penalty reward because the minimum distance from vehicle to obstacle may be swayed by the shape of obstacle.

As above mentions, the amount of training episode effects to the vehicle's behavior. The control system uses DQN model, which is trained at 1400, 5000, 7000, and 8000 training episode, respectively. The demonstration is shown in Figure 8. The vehicle control system using DQN model at 1400 and 5000 training episodes can be stop before the vehicle crash with the obstacle but it is not good enough to avoid the obstacle as shown in Figure 8a and Figure 8b, respectively.

After DQN model is trained at 7000 and 8000 training episode, the vehicle's behavior will be improved that the vehicle can avoid the obstacle as shown in Figure 8c and 8d. The vehicles become avoid the obstacle with deceleration and they can stop the destination. The vehicle using model at 7000 training episode immediately avoids the obstacle when LiDAR detected the obstacle. The vehicle using model at 8000 training episode avoids the obstacle when the vehicle nears the obstacle rather than other cases.

7. Conclusion

According to result, this paper successes to develop control system using DQN algorithm for obstacle avoidance. CARLA is used to present the result by visualization and graph. The vehicle motion using DQN model at high number of training episode is greater than low number of training episode.

This page contains no comments

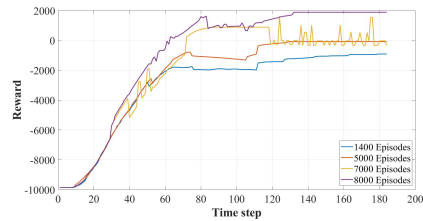


Figure 10. Reward for each number of training episode for straight motion test.

To train DQN model, the best training result is recognized to replay memory size. If the model is trained by TensorFlow version 1, the recognized memory may be divergence and some memories may not best results. It causes the failure of the simulation results. TensorFlow version 2.8 is used instead of version 1, and the results are better than the first version. The learning curve of

Referring to ϵ -greedy exploration, it influences the quality of replay memory. If the model uses small initial exploration value, the model may not find a new solution. If the final exploration value is large value, the model won't make the decision by itself.

For 0.5 ϵ -greedy exploration value, the model is slowly improving but it can not be suitable with the scenario case and it uses a lot of time in training. Thus, the preference in this project has been designed to slowly decay from 1 to 0.0001 as shown in table 1. The purpose aims to let the model search for the new solution and after that try to gradually increase to make the decision by itself. The replay memory size has been designed to make sure that it can collect all the states that happen enough to generate in DQN-model.

Author Contributions: Conceptualization, J.R., D.P. and P.K.; methodology, J.R., W.T., P.K. and D.P.; software, W.T. and P.K.; validation, J.R., P.K., D.P. and W.T.; formal analysis, D.P.; investigation, J.R.; resources, J.R.; data curation, W.T.; writing—original draft preparation, W.T.; writing—review and editing, W.T., D.P., and J.R.; visualization, W.T.; supervision, J.R. and D.P.; project administration, W.T.; funding acquisition, J.R. and D.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research project is supported by Thailand Science Research and Innovation (TSRI) Basic Research Fund: Fiscal year 2022 under project number FRB650048/0164.

Institutional Review Board Statement: Not applicable

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

References

- World Health Organization (WHO). Thailand's status against 12 Global Road Safety Performance Targets. 2020. Available online: <https://www.who.int/docs/default-source/thailand/roadsafety/thailand-status-against-12-global-road-safety-performance-targets.pdf?sfvrsn=92a24b064> (accessed on 10 June 2021).
- Kiran, G.R.K.; Deo, I.K.; Agrawal, S.; Haldar, S.; Shah, H.; Rudra, S.; Maheshwari, H.; Rathore, A.; Shah, P.; Nehete, A.; et al. Design and Implementation of Autonomous Ground Vehicle for Constrained Environments. In *Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC) 2019*, 236–239.
- Benterki, A.; Boukhniif, M.; Judalet, V.; Choubella, M. Prediction of Surrounding Vehicles Lane Change Intention Using Machine Learning. In *Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) 2019*, 839–843.
- Min, K.; Han, S.; Lee, D.; Choi, D.; Sung, K.; Choi, J. SAE Level 3 Autonomous Driving Technology of the ETRI. In *Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC) 2019*, 464–466.
- Sanil, N.; Venkat, P.A.N.; Rakesh, V.; Mallapur, R.; Ahmed, M.R. Deep Learning Techniques for Obstacle Detection and Avoidance in Driverless Cars. In *Proceedings of the 2020 International Conference on Artificial Intelligence and Signal Processing (AISP) 2020*, 1–4.

This page contains no comments

6. Serban, A.C.; Poll, E.; Visser, J. A Standard Driven Software Architecture for Fully Autonomous Vehicles. In *Proceedings of the 2018 IEEE International Conference on Software Architecture Companion (ICSA-C)* **2018**, 120–127. 276
7. Dworak, D.; Ciepiela, F.; Derbisz, J.; Izzat, I.; Komorkiewicz, M.; Wójcik, M. Performance of LiDAR Object Detection Deep Learning Architectures Based on Artificially Generated Point Cloud Data from CARLA Simulator. In *Proceedings of the 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)* **2019**, 600–605. 277
8. Wang, C. 2D Object Detection and Semantic Segmentation in the Carla Simulator, 2020. 278
9. Sung, K.; Min, K.; Choi, J. Driving Information Logger with In-Vehicle Communication for Autonomous Vehicle Research. In *Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT)* **2018**, 300–302. 279
10. Marti, E.; de Miguel, M.A.; Garcia, F.; Perez, J. A Review of Sensor Technologies for Perception in Automated Driving. *IEEE Intelligent Transportation Systems Magazine* **2019**, 94–108. 280
11. Warren, M.E. Automotive LIDAR Technology. In *Proceedings of the 2019 Symposium on VLSI Circuits* **2019**, 254–255. 281
12. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Conference on Robot Learning (CoRL 2017)*, Mountain View, United States; November 10 2017. 282
13. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning, **2013**. 283
14. CARLA Documentation, Sensors reference. 2020. Available online: <https://carla.readthedocs.io/en/latest/refsenors/> (accessed on 22 June 2021). 284
15. Carla Simulator, Open-source simulator for autonomous driving research. 2020. Available online: <https://carla.org/> (accessed on 1 June 2021). 285
16. Velodyne Lidar, High Resolution Real-Time 3D Lidar Sensor. 2020. Available online: <https://velodynelidar.com/products/hdl-32e/> (accessed on 1 June 2021). 286

This page contains no comments