# Adaptability of Deep Q-Networks in Autonomous Driving: Influences of Variable Obstacle Dynamics and Road Conditions

*Yuanjun* Sun*

Viterbi School of Engineering, University of Southern California, 3650 McClintock Ave Los Angeles, CA, United States

**Abstract.** Autonomous vehicles (AVs) have garnered significant attention due to their potential for enhancing transportation accessibility, and traffic safety. Despite numerous advancements, current AV systems still face challenges in fully autonomous operation under real-world conditions. This study modifies the CarRacing-v2 environment from the OpenAI Gymnasium to simulate realistic road conditions with obstacles. Deep Q-Network (DQN) combined with Convolutional Neural Networks (CNN) is utilized for perception and path planning. The approach investigates the impact of varying obstacle sizes, shapes, movement speeds, and road surface conditions on the agent's performance. The model is trained over 3000 episodes, with the evaluation focusing on cumulative rewards. The results show that the agent's performance is inversely related to obstacle size; the smaller the obstacle, the poorer the performance, which can be attributed to insufficient feature extraction. Irregular-shaped obstacles and reduced surface traction both caused a delayed learning response but will eventually achieve the desired reward. Variations in obstacle colors showed that red obstacles caused confusion due to similarities with red curbs of the road, indicating the model's over-dependence on luminance. In general, the agent shows adaptability to various road conditions, but enhancements in perception are needed for complex visual cues.

## 1 Introduction

In recent times, Autonomous Vehicles (AVs) have received significant attention and had breakthroughs in commercial deployment. In designated areas, people can find driverless ride-share vehicles in use. AVs aim to provide more accessible transportation for a boarder range of people, reduce traffic congestion, and most crucially, create a safer traffic environment. Statistics have shown that human error contributes to 95% of all traffic accidents in the UK and is the sole cause of 76% of these incidents [1, 2]. AVs have the potential to minimize human errors. However, according to NHTSA, all driving automation requires full driver attention, which reflects the instability of the current autonomous systems in handling real-world road conditions independently [3]. Thus, it has been the objective of

* Corresponding author: yuanjuns@usc.edu

computer scientists to create a more robust autonomous driving system, ultimately achieving full autonomy.

Autonomous Driving System (ADS) is structured into multiple layers. The system begins with the hardware sensors layer, which gathers environmental data, followed by the perception layer which processes this data for object detection and tracking. The localization and mapping layer then determines the vehicle's precise position in its surroundings. The subsequent layers include assessment for risk evaluation and anticipating surrounding drivers' decisions, planning and decision-making for safe route navigation, and finally, the control layer, which executes driving actions like steering and acceleration [4].

Machine learning has advanced some of the layers of ADS. Specifically, Convolutional Neural Network (CNN) and Deep Convolutional Neural Network (DCNN) are the most widely used methods in the perception layer of AVs. Research has been done using CNN for road segmentation, obstacle detection, and traffic light recognition [5-7]. This success is largely due to the widespread use of cameras as the hardware sensors and CNNs' ability to statistically extract complex features from vast amounts of data. You Only Look Once (YOLO) network and Single Shot Detector (SSD) are exemplary efficient object detection algorithms based on CNN [4, 8, 9]. Reinforcement Learning (RL) is a machine learning paradigm in which an agent acquires decision-making skills through interaction with an environment, receiving punishments or incentives for its actions, with the objective of maximizing cumulative rewards. RL has shown promising results in the decision-making and path-planning layer. For instance, Chang et al. proposed an improved Dynamic Window Approach (DWA) for mobile robot path planning using a Q-learning-based method. This approach modifies existing evaluation functions and introduces new ones to enhance navigation efficiency and adaptability in unknown environments. Simulation results show that the Q-learning-based DWA outperforms the original DWA, demonstrating higher navigation efficiency in both static and dynamic environments [4, 10]. Q-learning is further enhanced by using a neural network to approximate the Q-values for state-action pairs called Deep Q-Network (DQN), which enables learning in high-dimensional state spaces. Zhao et al. proposed a Deep Reinforcement Learning (DRL) framework using Double Deep Q-Network (DDQN), an extension to DQN which reduces the overestimation bias of Q-values, to model decision-making for autonomous highway driving. A simulation platform based on SUMO was developed to facilitate control algorithm variations. Results showed that the trained agent effectively avoided collisions and reached the highest safe driving speed, demonstrating the approach's potential for improving autonomous driving systems [4, 11].

In this paper, CNN is utilized for perception tasks and DQN is used for path planning in a modified gymnasium environment designed to simulate realistic road conditions with obstacles. This study focuses on analyzing the impact of obstacle dynamics on the learning efficiency and decision-making capabilities of the path-planning algorithm, aiming to provide insights for real-world autonomous driving scenarios. This study provides further modifications and explorations based on the code from Govor [12].

The rest of this paper is organized into 3 sections. The method section introduced the network architecture and algorithm specifics used for training. The result section presents the data obtained from training using the proposed algorithm in varied environments, followed by a discussion section that analyzes these findings. The content of the paper and suggested future works are summarized in the final section.

# 2 Method

## 2.1 Game design

The CarRacing-v2 game environment from OpenAI's Gymnasium library was modified and used to train a reinforcement learning agent. DDQN with CNN was used for perception and decision-making in the environment. The game involves a continuous control environment where the agent must drive a car along a closed-loop, procedurally generated road. The agent is rewarded for staying on the track and driving forward, while penalties are given for crashing or driving off the track. The goal is to maximize cumulative rewards through repeated interactions with the environment. The road is composed of fixed tiles, and the termination of the game requires each tile to be traversed or in contact with an obstacle or boundary. The environment is modified to mimic real-world conditions incorporating various obstacles, pedestrian crossings, blind spot appearances, and road surface variation. Specifically, the obstacle sizes are defined relative to the track dimensions, and three different obstacle sizes were used: small (5/Default Track Scale), medium (10/Default Track Scale), and large (15/Default Track Scale). The small size is approximately 1/7th of the track width and the large size is about 1/3rd. The obstacle shapes are either rectangles, simulating a stationary vehicle, or irregular polygons to represent debris. Obstacle colors are varied in red, yellow, and blue. Pedestrians are simulated by obstacles that move across the road at preset speeds of (1/Default Track Scale) per frame. The default track scale in the current version of the gymnasium is 6. The “pedestrians” also can remain hiding and start crossing upon arrival of the vehicle. The hiding mechanism is achieved by masking the moving obstacle with the background color outside of the road to simulate pedestrians in the blind spot. Additionally, certain road sections are colored light blue to indicate areas with less traction, simulating icy road conditions. In each environment, there are exactly 10 obstacles. The main objective is to examine the impact of the environmental factors on the performance of the algorithm. An example game interface is provided in Fig. 1.

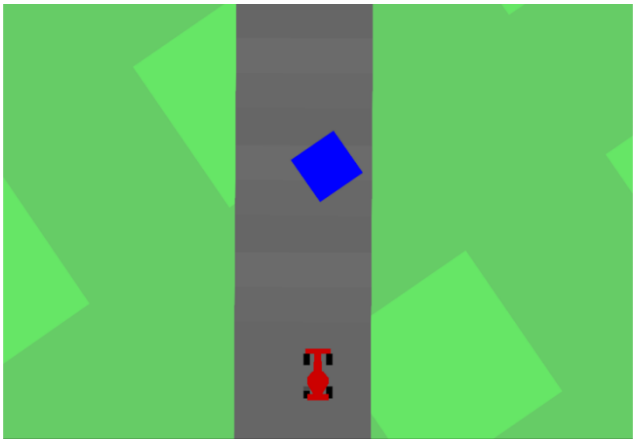

**Fig. 1.** Game Interface (Photo/Picture credit: Original).

## 2.2 Convolutional neural network-based feature extraction

Convolutional Neural Networks (CNNs) represent a category of deep learning algorithms that demonstrate superior efficacy in processing data with a grid-like topology, such as images. CNNs are composed of three layers, the convolutional layers, pooling layers, and fully connected layers. The convolutional layer is where computation occurs, primarily

serving as a feature extractor. The pooling layer handles the downsampling of input data and serves to reduce the number of input parameters and size. Finally, the fully connected layer makes final decisions and feature classifications.

In this model, there are two convolutional layers shown in Fig. 2. The input to the first layer is a 4-channel image and the height and width of each frame are 84x84. Each frame is converted to a grayscale image for processing to reduce computational complexity and avoid color overfitting. This layer applies 16 filters, each of size 8x8, with a stride of 4, followed by Rectified Linear Unit (ReLU) activation. The first layer is responsible for extracting fundamental features such as edges, corners, and textures. The second layer applies 32 filters of size 4x4 with a stride of 2, also with ReLU activation to recognize more complex structures. It is responsible for detecting relevant details like road boundaries, car components, or obstacles. The output is sent to a flatten layer that converts the 3D tensor into a 1D vector. The result is then processed by two fully connected layers for final decision-making. The final output is a vector, where each element represents the Q-value for each action.

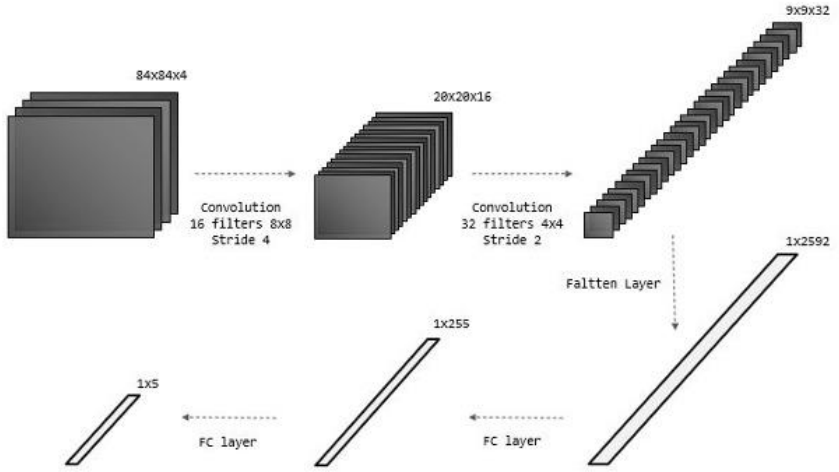


**Fig. 2.** Convolutional Neural Network (CNN) Architecture for Perception (Photo/Picture credit: Original).

## 2.3 DQN-based path planning

Q-learning is a model-free reinforcement learning algorithm that uses the Bellman equation to iteratively update the Q-value function. The Q-value measures the efficacy of an action performed in a specific state. Deep Q-Networks (DQN) is a prominent algorithm that integrates Q-learning with deep neural networks. Specifically, the Q-value function is approximated by a deep convolutional neural network, instead of in a tabular fashion as in Q-learning. As a result, it allows DQN to handle environments with large or continuous state spaces.

In this model, the DQN algorithm uses two identical CNNs, the target network and the online network. The online network, the main network, uses the feature extracted from previous steps to predict Q-values for each possible action. This network is timely updated whereas the target network has temporarily fixed parameters. The weights of the target network are updated periodically by synchronizing from the online network, which aids the training by providing more stable Q-values. The epsilon-greedy policy is applied when choosing from exploration (random action) or exploitation (action based on experience). The key to DQN is the use of Temporal Difference (TD) learning to update the Q-values, which

learns from the difference between the current and target Q-value. The target Q-value is calculated in the target network as follows:

$$y_j^{DQN} = R_j + \gamma \max_{a'} Q(s', a', \theta_t) \quad (1)$$

Where $y_j^{DQN}$ is the target Q-value that the network is trying to predict, $R_j$ is the immediate reward, $\gamma$ is the discount factor for future rewards, $\max_{a'} Q(s', a', \theta_t)$ is the maximum estimated future reward at future state $s'$and action $a'$, and $\theta_t$ represents weights of the target network at time step t. To further improve stabilization, a replay buffer is deployed that stores transitions (s,a,r,s′), and each training process would sample mini-batches from the replay buffer to update the Q-value.

### 2.4 Implementation details

This model is implemented using PyTorch as the primary framework. The learning process is governed by an Adam optimizer with a learning rate set at 0.0002. Smooth L1 loss is adopted as the loss function. The model trains 3000 episodes for each modified environment, with mini-batches of size 32 sampled from the replay buffer. The result is evaluated based on cumulative average reward, as it reflects the effectiveness of the agent in navigating through the environment.

## 3 Results and discussion

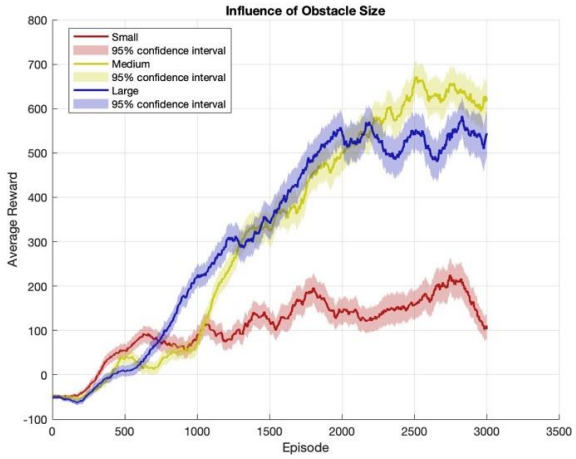


**Fig. 3.** Impact of Obstacle Size on Average Reward (Photo/Picture credit: Original).

Fig. 3 displays the average reward across training episodes for three different obstacle sizes. The overall increasing trend indicates a positive learning process. The agent performed the best when trained with medium to large obstacles. The reward rises steeply after 500 episodes and remains high. The agent struggled the most with small obstacles, showing minimal improvement and 5 times less average reward. The result contradicts expectations as the smaller the size, the easier to navigate around. The unexpected data could be explained by insufficient feature extraction or resolution in the CNN, which struggled to capture small, subtle visual cues. The environment involves converting images to grayscale and resizing

them to 84x84 pixels, which can lead to loss of fine details necessary for detecting smaller obstacles. Thus, the smaller objects might not be sufficiently represented in the agent's perception pipeline, leading to poor avoidance rates.

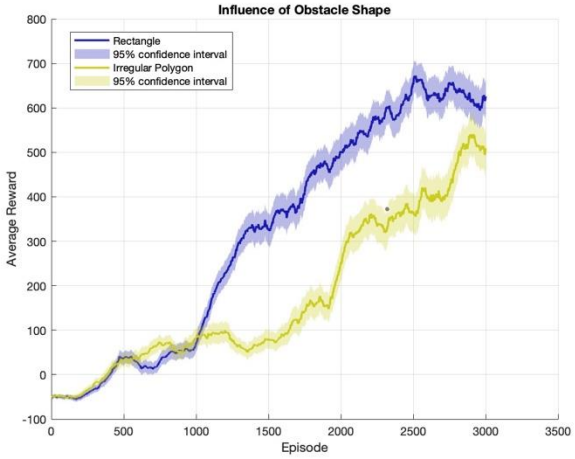


**Fig. 4.** Impact of Obstacle Shape on Average Reward (Photo/Picture credit: Original).

The medium obstacle is compared with irregular-shaped obstacles of similar size that try to simulate debris on the road in Fig. 4. The trend shows that the algorithm does improve on avoidance rate during the learning process for debris, but exhibits a delayed increase in reward, starting from approximately 2000 episodes whereas the rectangle starts from 500 episodes. Both eventually achieved similar rewards. The delayed reward can be attributed to added complexity from irregularity of shapes which requires more time for CNN to recognize and interpret.

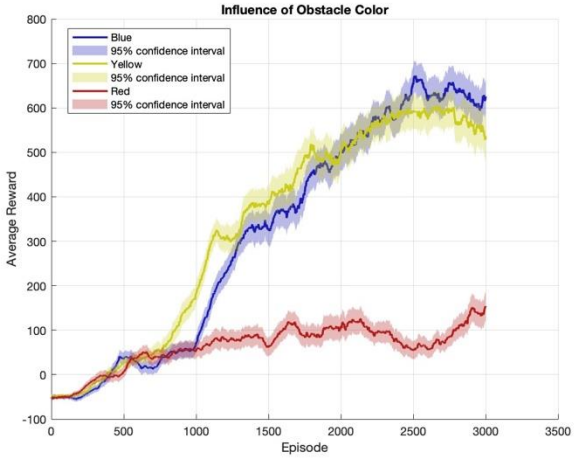


**Fig. 5.** Impact of Obstacle Color on Average Reward (Photo/Picture credit: Original).

The color of obstacles is varied during the training. Fig. 5 shows that the red color exhibits significant disadvantages with much lower average rewards than the other two colors. The preprocessing pipeline of CNN includes converting images to grayscale, a measure to allows the feature extraction to focus more on shapes rather than color. However, red-colored obstacles render more challenge suggesting that the agent is confusing the red on the obstacle

with the red on the curb of the road. The current CNN's dependencies on color information, luminance in this case, can excessively alter the result.

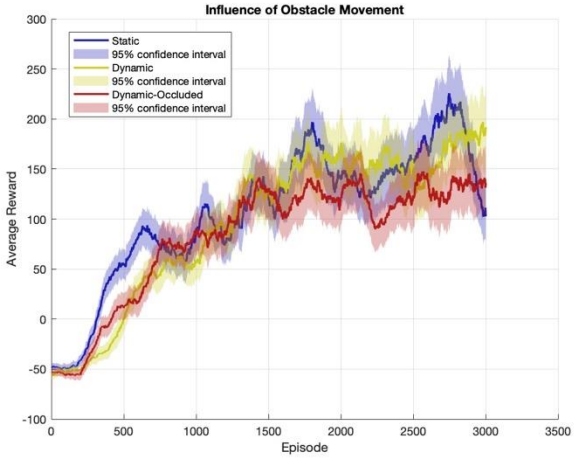


**Fig. 6.** Impact of Obstacle Movement on Average Reward (Photo/Picture credit: Original).

The data of small-size stationary obstacles previously analyzed is compared with same-size obstacles with movement. The motion trajectory is perpendicular to the direction of the road to represent a pedestrian crossing the road. In Fig. 6, the green curve represents moving obstacles with clear visibility, whereas the blue curve represents objects that are occluded and only start moving when the vehicle arrives. All three lines demonstrated a similar trend and strong reward accumulation considering the size, which indicates that movement does not pose significant challenges to obstacle recognition and avoidance.

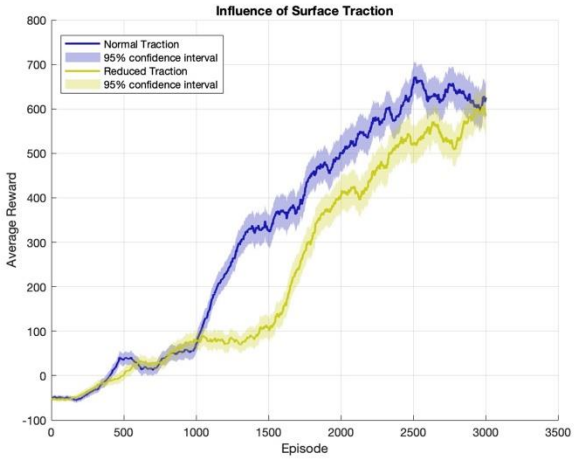


**Fig. 7.** Impact of Surface Traction on Average Reward (Photo/Picture credit: Original).

The medium-sized obstacle is placed onto a road with segments of reduced traction that simulate icy or wet road conditions. Compared with normal conditions shown in Fig. 7, the reduced traction shows a slower learning process but eventually reaches the same level of average reward. The accelerated learning phase is delayed by approximately 500 episodes for icy roads, which is expected given the additional complexity. The result indicates the agent is capable of adapting to road traction change given sufficient training time.

The inability of the CNN to adequately capture small obstacles suggests that, in real practice, AVs need advanced sensors and higher-resolution imaging. Additionally, the agent's over-dependence on color indicates that perception algorithms should be designed to be less sensitive to color variations and more focused on invariant features to prevent misclassification due to, for example, lighting change.

The current CNN perception layer utilized in this study employs a one-size-fits-all approach, which only revealed deficiencies in specific conditions such as small obstacle size and color variations but not tailored to handle them. Refinement can be done such as increasing the resolution of input images or using smaller kernel sizes and strides in convolutional layers to better capture finer details of small obstacles and debris. The current environment does not include 3D objects and could not represent certain complexities such as sensor noise. Improvement is necessary in using physical robotic platforms beyond software simulations.

## 4 Conclusion

This study assesses how different factors affect the path-planning and obstacle avoidance of an automated driving system based on CNNs and DQN. The factors investigated include obstacle size, shape, color, movement, and road traction, each aimed at simulating real-life conditions. The agent performed optimally with medium to large obstacles, showing rapid learning and maintaining high rewards. However, the agent struggled with small obstacles, achieving five times lower average rewards compared to large size due to insufficient feature extraction in the CNN. When faced with irregular-shaped obstacles simulating debris or reduced road traction simulating icy road conditions, the agent exhibited delayed learning, but eventually achieved the optimal level of reward. The agent's performance was also hindered with red obstacles, likely due to grayscale preprocessing diminishing color distinctions, causing confusion between red obstacles and similar-colored road elements. Obstacle movement, representing pedestrian crossing and animals jumping from blind spots, did not pose a significant challenge to the algorithm. Overall, the DQN algorithm performs well in decision-making as all training shows a positive learning curve. However, this study finds that CNN needs to be tailored toward recognizing fine details. Relying on a “single-sized” CNN may pose challenges in feature extraction in real-life scenarios where obstacles vary widely. Moreover, the feature extraction shows over-dependence on luminous pose threat for object distinction. In the future, the algorithm should be put to test under environments that include 3D objects and physical testing platforms. Also, methods to diminish luminous dependency and tailoring CNN for specific feature extraction can be investigated.

## References

1. Department for Transport, The Pathway to Driverless Cars: Summary Report and Action Plan (2015) https://assets.publishing.service.gov.uk/media/5a7f198ae5274a2e87db3ca9/pathway-driverless-cars-summary.pdf
2. Kent County Council J., Crash and Casualty Data: Tech. Rep. (2022) https://www.kent.gov.uk/roads-and-travel/road-safety/crash-and-casualty-data
3. National Highway Traffic Safety Administration, Automated Vehicles for Safety (2024), https://www.nhtsa.gov/vehicle-safety/automated-vehicles-safety

4. M. Reda, A. Onsy, A. Y. Haikal, A. Ghanbari, Path planning algorithms in the autonomous driving system: a review. Robotics and Autonomous Systems, **174**, Article 104630 (2024)
5. S. Kong, J. Park, S.-S. Lee, S.-J. Jang, Lightweight traffic sign recognition algorithm based on cascaded CNN, in Proceedings of the 2019 19th International Conference on Control, Automation and Systems (ICCAS), Jeju, South Korea, October 15–18, pp. 506–509 (2019).
6. A. Datta, T. I. Meghla, T. Khatun, M. H. Bhuiya, S. R. Shuvo, M. M. Rahman, Road object detection in Bangladesh using Faster R-CNN: a deep learning approach, in Proceedings of the 2020 6th IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), Bhubaneswar, India, December 26–27, pp. 352–355 (2020).
7. Y. Lyu, L. Bai, X. Huang, Road segmentation using CNN and distributed LSTM, in Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, May 26–29, pp. 1–5 (2019).
8. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, June 27–30, pp. 779–788 (2016).
9. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: Single Shot Multibox Detector, in Computer Vision – ECCV 2016, Amsterdam, The Netherlands, October 11–14, **9905**, pp. 21–37 (2016).
10. L. Chang, L. Shan, C. Jiang, Y. Dai, Reinforcement-based mobile robot path planning with improved dynamic window approach in unknown environment. Autonomous Robots, **45**(1), pp. 51–76 (2021).
11. J. Zhao, T. Qu, F. Xu, A deep reinforcement learning approach for autonomous highway driving, in Proceedings of the 3rd IFAC Workshop on Cyber-Physical and Human Systems (CPHS), Beijing, China, December 3–5, **53**(5), pp. 542–546 (2020).
12. Github, DQN Car Racing, (2024), https://github.com/wiitt/DQN-Car-Racing