# Deep Q-Learning-Based Trajectory Optimization for Vehicle Navigation in CARLA

Rami BOUMEGOURA(1,2)*, Youcef ZENNIR(1),Seif Ferhat TAMINE(2)
(1) Automatic Laboratory of Skikda (LAS), University of 20 August 1955, Skikda, Algeria
(2) Department of Petrochemical. University of 20 August 1955, Skikda, Algeria
r.boumegoura@univ-skikda.dz

**Abstract:** This paper presents a comprehensive study that focuses on simulating a vehicle within the autonomousvehicle simulator, CARLA. The primary objective of this research is to enable the vehicle to accurately follow apredetermined trajectory while effectively avoiding obstacles in its environment. Deep Q-Learning algorithms areemployed to achieve this goal, aiming to optimize the safety of the vehicle'snavigation. The simulation of the vehicleserves as a platform for studying the rules of Deep Q-Networks (DQN) and their impact on the vehicle's navigation.The objective is to identify the most suitable rule that leads to improved optimization of the vehicle's trajectory. Byleveraging the capabilities of CARLA as the simulation environment and implementing state-of-the-art DQNalgorithms, this research contributes to the advancement of autonomous vehicle technology. The findings of this studyhave practical implications for enhancing the safety and efficiency of autonomous vehicle navigation systems, makingthem highly relevant to industryprofessionals, researchers, and academic scholars in this field.

**Keywords:**Deep Q-Learning, CARLA, Autonomous vehicle navigation, Optimization.

## 1. INTRODUCTION

In the realm of autonomous driving, the quest for enhancing vehicle navigation efficiency and safety has driven researchers to explore innovative solutions rooted in artificial intelligence.Initially, early efforts in applying reinforcement learning (RL) to autonomous vehicles primarily concentrated on maneuver execution [1]. Agents learned steering and braking control directly from environmental data, enabling them to execute driving maneuvers in simple scenarios. Nevertheless, conventional algorithms for low-level control, as indicated by references [2], [3], have consistently demonstrated superior performance compared to RL approaches, which tend to exhibit slower learning rates and suboptimal performance. Consequently, we advocate for the utilization of RL in high-level decision-making tasks within more intricate settings involving multiple adversarial vehicles.

To manage this increased complexity, a proposed strategy involves the separation of the planning task into distinct high-level decision-making and maneuver execution layers [4]. In this hierarchical approach [5], the execution layer handles motion-related aspects, while the decision-making layer carries out high-level actions.

An alternative approach involves end-to-end learning [6], [7], where control commands are directly inferred from sensor-derived input. Leveraging advancements in deep learning algorithms and modern hardware capabilities, NVIDIA researchers introduced a supervised learning method based on convolutional neural networks (CNN) for end-to-end control of automobiles [8]. Subsequent studies have adopted and built upon this end-to-end learning approach [9], [10].

In our focus on RL-based approaches for high-level decision making at intersections, the utilization of Deep Q-Network (DQN) algorithms is prevalent [11]. Additionally, references [12], [13] employ DQN algorithms to address diverse driver behaviors. Some studies, exemplified by [14], [15], specifically train at occluded intersections using a risk-based reward function. The input vector encompasses details about the ego vehicle, existing lanes, and surrounding vehicles.

Motivated by these challenges and opportunities, our research aims to contribute to the advancement of autonomous driving technology by optimizing vehicle navigation through Deep Q-Learning within the CARLA autonomous vehicle simulator. Through simulations, our study seeks to identify effective rules for enhancing navigation performance, with a particular focus on achieving precise trajectory following and obstacle avoidance. This research holds significant promise for elevating the safety and efficiency of autonomous vehicles, addressing critical aspects of high-level decision-making in complex driving scenarios.

## 2. DEEP REINFORCEMENT LERANING METHODOLOGY

### *Reinforcement Learning (RL)*

As a form of machine learning, RL addresses the agent's interaction with the environment by learning strategies aimed at maximizing returns or achieving specific objectives[16]. The prevalent model in RL is the standard Markov Decision Process (MDP), which comprises a tuple *(S, A, R, p)* defining the control problem. Here, S and A represent the state space and control space, respectively, R signifies the reward function, and p denotes the state transition. A mapping from sets of states to sets of actions is denoted as: $\pi : S \rightarrow A$. At any given time t, the action and state of the agent in the environment can be represented as $a_t \in A$ and $s_t \in S$, respectively. The probability that the agent performs an action and transitions to the subsequent state $s_{t+1} \in S$ is represented as $p(s_t, a_t)$, with associated reward $r_t$. To prevent potential biases and infinite loops in an infinite time series caused by directly summing rewards, a discount factor $\gamma$ is introduced, prioritizing the current reward over future ones.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \qquad (1)$$

In the current state s, the agent executes action a according to strategy π, consistently following it to carry out operations. As a result, it acquires the state function $v_\pi(s)$ and the state-action function $Q_\pi(s, a)$, which are defined as follows:

$$\begin{aligned} v_\pi(s) &= E_\pi\left[R_t \middle| S_t = s\right] \\ Q_\pi(s,a) &= E_\pi\left[R_t \middle| S_t = s, A_t = a\right] \end{aligned} \qquad (2)$$

In this context, where π represents the control policy, the optimal strategy aims to maximize future returns. The solution process can be broadly divided into two stages: prediction and action. During prediction, a strategy is provided to assess the associated $v_\pi(s)$ and $Q_\pi(s, a)$. Subsequently, during action, the optimal action corresponding to the current state is determined based on the value function. Consequently, the state-action function is reformulated into a recursive form to update the algorithm.

$$Q_\pi(s,a) = E_\pi\left[r_t + \gamma \max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1})\right] \qquad (3)$$

The optimal policy is derived from the state-action value function.

$$\pi(s) = \arg\max_a Q(s,a) \qquad (4)$$

As an iterative process, RL faces two challenges each time: evaluating a policy function and updating the policy based on the value function. Traditional RL approaches aim to find the optimal policy by iteratively solving Bellman's equation, which involves making $Q_\pi(s,a)$ converge to obtain the optimal strategy. However, employing Bellman's equation to solve the Q function in larger state spaces can be computationally expensive in practical solution processes.

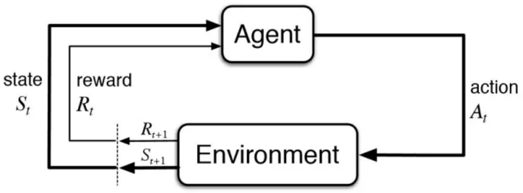

Fig. 1 Reinforcement Learning Scheme[]

### *Deep Q network*

To mitigate the computational costs associated with the iterative process, a neural network is employed to approximate the state-action value function. Initially, the update function of Q-learning can be expressed as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \qquad (5)$$

A varying learning rate α within the range [0, 1] is utilized to balance the significance of the current environment's learning experience against previous ones. Here, s' and a' represent the state and action quantities in the subsequent procedure. The Deep Q-Network (DQN), which integrates neural network techniques with Q-learning, was introduced to approximate the action-value function within high-dimensional state spaces.

$$Q(s,a|\theta) \approx Q(s,a) \qquad (6)$$

In contrast to Q-Learning, where only a neural network and a target Q network are utilized, DQN incorporates experience replay during training. The stochastic gradient descent (SGD) algorithm is employed to update the network parameters throughout the training process. The loss function of DQN is defined as follows:

$$L(\theta) = E\left[(targetQ - Q(s,a|\theta))^2\right] \qquad (7)$$

The optimization objective for the state-action function can be defined as:

$$targetQ = r + \gamma \max_{a'} Q(s',a'|\theta) \qquad (8)$$

Where θ represents a neural network parameter, the policy gradient method is a model-free approach aimed at optimizing the expected total return of a strategy, exploring the optimal strategy directly within the strategy space[17]. The greedy policy selects the action that maximizes the value function on each occasion. However, states and action values that were not sampled previously will not be chosen subsequently because they have not been evaluated. The ε-greedy policy combines the benefits of exploration and exploitation. Actions are chosen stochastically from all available actions with a probability of ε, while the best action is selected with a probability of 1-ε.

## 3. Simulation & Results

In our research, we deploy reinforcement learning, specifically deep Q-learning, in the CARLA simulation environment, where the agent, resembling a self-driving car, is tasked with navigating through a virtual environment by following predefined waypoints along a designated trajectory. This involves configuring the environment with vehicles, sensors, and waypoints, connecting to the CARLA server, and establishing vehicle spawn points. The fundamental objective is for the agent to efficiently maneuver the environment, adhering to specified rules, and avoiding obstacles while reaching the designated waypoints.

The simulation unfolds in several key stages: first, the generation of waypoints that define the desired path; second, the navigation of the agent through the environment, leveraging sensor data to make informed decisions; third, the implementation of a reward and penalty system where the agent receives feedback based on its actions, encouraging adherence to desired behaviors and discouraging deviations; fourth, the learning process, where the agent uses the received feedback to iteratively improve its navigation strategy over time; and finally, the iterative optimization phase, where the agent refines its behavior through repeated interactions with the environment.

By following waypoints and receiving feedback, the agent gradually learns to navigate the simulated environment effectively, ultimately achieving its goal of reaching designated destinations while adhering to predefined rules and constraints. This simulation framework provides a valuable platform for studying and advancing autonomous vehicle navigation technology.

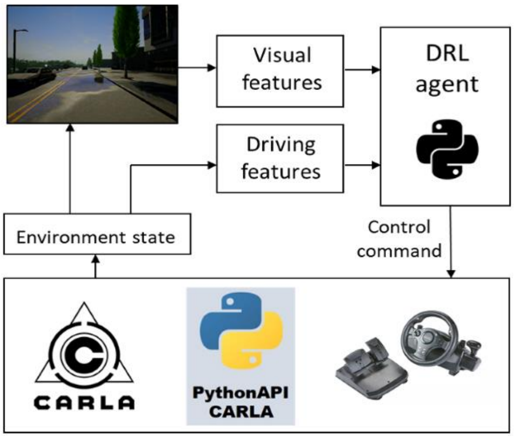

Fig. 2Framework Scheme [20]

*Case Study 1:*

The initial model, with basic conditions (4 actions, 6 reward rules), aims to observe its functionality and refine actions and rewards for improved accuracy. Test conditions for Model 1 are specified, including GPU usage, total learning time, trajectory type, actions, rewards, and sensors employed.

Table 1 Test Conditions (Model 1)

| **Test Conditions (Model 1)** | | | |
|---|---|---|---|
| **GPU Fraction Used** | 0.35 | **Total Training Time** | 1h 13min 8sec |
| **Refresh Rate (γ)** | 0.95 | **Learning Rate (α)** | Degradation:α*0.95 |
| **Trajectory Type** | Straight line | | |
| **Actions** | • Action 1:Accelerate<br>• Action 2: Turn right<br>• Action 3: Turn left<br>• Action 4: Decelerate | | |
| **Rewards** | • Reward1: Collision (-30)<br>• Reward 5: Roll to a final point (+50)<br>• Reward 6: Line following (+10)<br>• Reward 7: Near-line navigation (+1)<br>• Reward 8: Off-line navigation (-30) | | |
| **UsedSensors** | • Camera RGB<br>• Collision Sensor | | |

Figure 3.representing the records from the 1st model trained over 200 episodes, we observe an average reward curve displaying values between 150 and 200 points. These values appear unusually high and contradict the established rules. This discrepancy may be attributed to the agent's inclination towards action number 4 (deceleration), which slows down the vehicle, allowing for more time to accumulate rewards. Consequently, this phenomenon leads to

false learning, where the agent is rewarded without actively performing. Upon examining the model's performance, it became evident that certain additional conditions must be considered to optimize its actions.

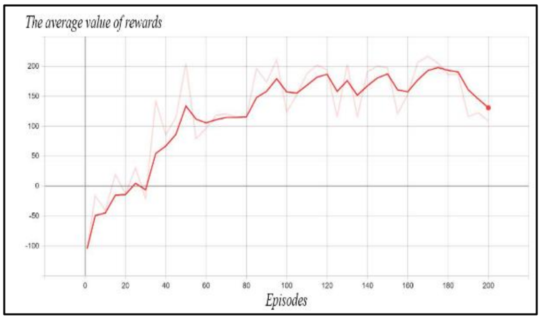

Fig. 3The average value of rewards (Case 1)

In comparison to other reward statistics (max and min) from Fig 4. And Fig 3, we observe a direct relationship with the average reward curve in terms of evolution, with some disturbances hindering the development of the max curve (not entirely stable). This disturbance may indicate that the agent did not select the best action to perform, suggesting slow learning.

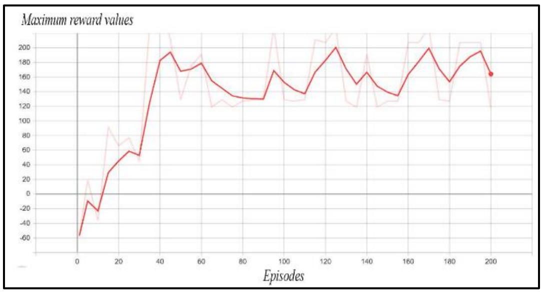

Fig. 4Maximum reward values (Case 1)

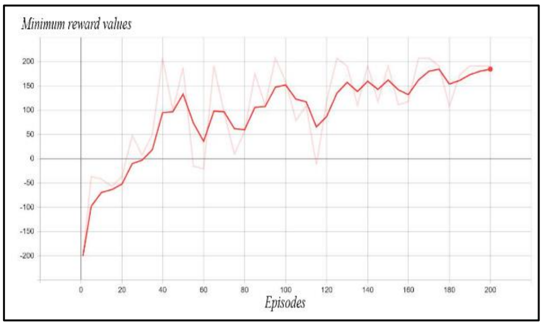

Fig. 5Minimum reward values (Case 1)

The overall conclusion of this test case is that it serves as a good starting point, but the model could greatly benefit from the addition of further rules.

*Case Study 2:*

Building upon the foundation of the previous model, a new reward rule was introduced to address the primary issue encountered in the initial model: gaining points without actively performing significant actions besides decelerating near waypoints. All other conditions remained unchanged as no apparent issues were identified.

Table 1 Test Conditions (Model 2)

| **Test Conditions (Model 2)** | | | |
|---|---|---|---|
| **GPU Fraction Used** | 0.35 | **Total Training Time** | 1h 13min 8sec |
| **Refresh Rate (γ)** | 0.95 | **Learning Rate (α)** | Degradation:α*0.95 |
| **Trajectory Type** | Straight line | | |
| **Actions** | • Action 1:Accelerate<br>• Action 2: Turn right<br>• Action 3: Turn left<br>• Action 4: Decelerate | | |
| **Rewards** | • Reward1: Collision (-30)<br>• Reward 4: Very slow navigation or no movement (-10)<br>• Reward 5: Roll to a final point (+50)<br>• Reward 6: Line following (+10)<br>• Reward 7: Near-line navigation (+1)<br>• Reward 8: Off-line navigation (-30) | | |
| **UsedSensors** | • Camera RGB<br>• Collision Sensor | | |

The model (second model) is trained for 200 episodes and depicted in the three curves (average-max-min). In the second model, we observe reasonably values ranging between 110 and 120, aligning with the set reward rule values, indicating the vehicle's behavior after adding the 4th rule. We can easily see how the vehicle learned to navigate realistically and logically compared to the previous model and its learning process.

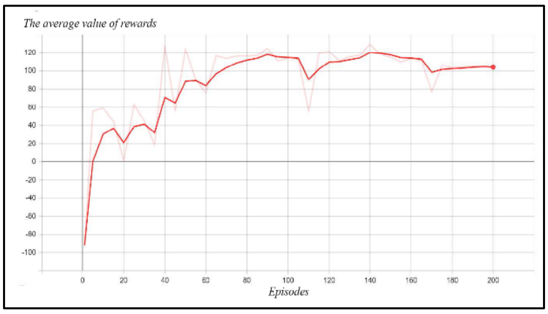

Fig. 6The average value of rewards (Case 2)

Concerning other reward statistics (max and min), especially the max curve, we notice stabilization after 60 episodes at a value of 120 points, indicating the agent has acquired optimal actions during its learning process.

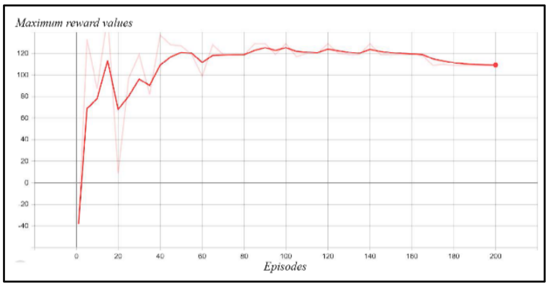

Fig. 7Maximum reward values (Case 2)

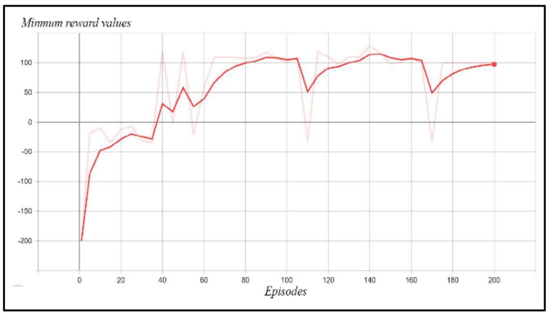

Fig. 8Minmum reward values (Case 2)

Based on these two experiments, we can conclude that optimizing vehicle control heavily depends on the number of chosen actions, adjusting perfect reward rules, and hyperparameter tuning of the chosen methods, all of which lead to the perfect outcome for the agent to adopt the optimum behavior consistently, whether it's following a trajectory or avoiding an obstacle.

## 4. CONCLUSION

In conclusion, our analysis of simulation results highlights the significant potential for optimizing vehicle driving using DQN in conjunction with logical and realistic rule adjustments. This optimization offers promising prospects for achieving realistic and optimal driving and navigation in CARLA's simulated environments.Overall, our research emphasizes the effectiveness of Deep Q-Learning within the CARLA framework for advancing autonomous driving capabilities. By refining learning strategies and rule settings, we can further enhance the performance and realism of autonomous vehicles, contributing to the development of safer and more efficient transportation systems in the future.

### *References*

[1] P. Wang, H. Li, and C.-Y. Chan, "Continuous Control for Automated Lane Change Behavior Based on Deep Deterministic Policy Gradient Algorithm." arXiv, 05-Jun-2019.

[2] 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." arXiv, 25-Apr-2016.

[3] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[4] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2018, pp. 2156–2162.

[5] F. Ye, S. Zhang, P. Wang, and C.-Y. Chan, "A Survey of Deep Reinforcement Learning Algorithms for Motion Planning and Control of Autonomous Vehicles." arXiv, 31-May-2021.

[6] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end Learning of Driving Models from Large-scale Video Datasets." arXiv, 23-Jul-2017.

[7] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to Drive in a Day." arXiv, 11-Sep-2018.

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars." arXiv, 25-Apr-2016.

[9] M. Moghadam, A. Alizadeh, E. Tekin, and G. H. Elkaim, "An End-to-end Deep Reinforcement Learning Approach for the Long-term Short-term Planning on the Frenet Space." arXiv, 25-Nov-2020.

[10] R. Chopra and S. S. Roy, "End-to-End Reinforcement Learning for Self-driving Car," in *Advanced Computing and Intelligent Engineering*, vol. 1082, B. Pati, C. R. Panigrahi, R. Buyya, and K.-C. Li, Eds. Singapore: Springer Singapore, 2020, pp. 53–61.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[12] T. Tram, A. Jansson, R. Gronberg, M. Ali, and J. Sjoberg, "Learning Negotiating Behavior Between Cars in Intersections using Deep Q-Learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2018, pp. 3169–3174.

[13] T. Tram, I. Batkovic, M. Ali, and J. Sjoberg, "Learning When to Drive in Intersections by Combining Reinforcement Learning and Model Predictive Control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, 2019, pp. 3263–3268.

[14] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-Aware High-level Decisions for Automated Driving at Occluded Intersections with Reinforcement Learning." arXiv, 09-Apr-2020.

[15] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe Reinforcement Learning with Scene Decomposition for Navigating Complex Urban Environments," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, 2019, pp. 1469–1476.

[16] M. S. Frikha, S. M. Gammar, A. Lahmadi, and L. Andrey, "Reinforcement and deep reinforcement learning for wireless Internet of Things: A survey," *Comput. Commun.*, vol. 178, pp. 98–113, Oct. 2021.

[17] Y. Fu, C. Li, F. R. Yu, T. H. Luan, and Y. Zhang, "A Decision-Making Strategy for Vehicle Autonomous Braking in Emergency via Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5876–5888, Jun. 2020.