Optimizing Performance in Autonomous Racing Cars Using Deep Q-Learning: A Comparative Study of Reward Systems

Author Information

Pranave K.C[1], Naga Sai Shreya Kunda[2], Mayank Pandey[3], Nippun KumaarA.A.*[,4]

[1-4] Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru, Amrita Vishwa Vidyapeetham, India

* Corresponding author. Email: aa_nippunkumaar@blr.amrita.edu[*,4]
Contributing authors: BL.EN.U4AIE21106@bl.students.amrita.edu[1],
BL.EN.U4AIE21085@bl.students.amrita.edu[2],
BL.EN.U4AIE21079@bl.students.amrita.edu[3],

Optimizing Performance in Autonomous Racing Cars Using Deep Q-Learning: A Comparative Study of Reward Systems

Abstract

The use of Reinforcement Learning is gaining popularity due to its application in autonomous vehicles. Race cars, in specific can be well trained with the help of Reinforcement Learning in order to optimize several factors like speed, decision making, and steering and control. Through this study, the aim is to utilize Deep Q Learning to optimize the performance of autonomous racing cars. The race car is trained on a custom built environment and is evaluated based on two different reward systems. The outcome of the study reveals that the second set of rewards are more suitable and yield better performance when compared for 10,000 episodes, with the second set of rewards over performing with an average of 282 units of distance more than the first set of rewards. These findings reveal that structuring rewards is an important factor and plays a crucial role while designing Reinforcement Learning systems.

Keywords: Autonomous Vehicles, Reinforcement Learning, Decision Making, Racing, Rewards, Deep Q-Network

1 Introduction

Autonomous racing technology has brought new challenges in the field of autonomous driving vehicles. The ability to navigate[1], avoid obstacles and follow safety rules without human intervention [2] [3] is being possible with the help of Reinforcement Learning. This is achieved by combining data from lidar, radar and high definition cameras and using that data to help the autonomous car learn precisely.

The development of autonomous vehicles, especially in racing scenarios, benefits greatly from Reinforcement Learning (RL) [4]. By interacting with their environment and learning through trial and error, RL helps cars improve over time [5]. Algorithms like Deep Q-Network (DQN)[6] are crucial for autonomous racing because they help cars make the best decisions for braking, steering, and accelerating around turns [7]. Using RL, racing cars can adapt and optimize their strategies in fast-changing conditions, enhancing their efficiency and safety. This ongoing learning process not only speeds up the development of autonomous racing technology but also improves autonomous driving systems for regular cars.

Key Contributions of this Study are :

1. Design and create a custom race track environment for simulation of the race car.
2. Evaluate and compare two different reward systems to determine the most effective one.
3. Implement Deep Q Learning to train an autonomous race car.

The following sections will flow in a sequence beginning with section 2 that gives an overall summary of autonomous racing cars. Further on, section 3 will talk about the proposed solution of this study. The results will be highlighted in Section 4 and Section 5 will conclude with an overview and potential future scope.

2 Related Work

The section provides essential insights from similar and related work, highlighting the current state of knowledge for this study aimed at reducing decision-making latency in autonomous racing cars.

Balaji et al [8] addresses issues such as optimizing policy training by utilizing reinforcement learning algorithms and comprehensive assessment techniques. Users have successfully achieved high speeds and successfully exhibited sim-to-real navigation with minimal tuning. Remonda et al [9] contribute to lap-time minimization through the use of reinforcement learning (RL) in autonomous racing. The study has done a remarkable task of evaluating 10 variations of deep deterministic policy gradient (DDPG) in simulations and discovers which models trained with reinforcement learning (RL) outperform manually constructed bots and generalize to new tracks. Importance was given to enhance the model's adaptability with planned actions, advanced telemetry and image derived inputs. Evans et al [10] tackle the simulation to reality problem by developing a Viability Theory-based supervisor. High-performance racing and practical application are made feasible by the technique. Importance was given to research which will focus on additional safety-critical areas and enhance safety verification in the absence of localization.

Mao et al [11] develop a new framework for taxi dispatch optimization using deep reinforcement learning, which can handle situations where there is imbalance of data. The proposed algorithm utilizes neural networks to estimate policy and value functions, and it can be expanded to larger networks with recurrent or graph neural networks. It can also handle mixed fleets and investigate the effects of ride-sharing on dispatch strategies. Farooq et al [12] develop a model for autonomous vehicles that uses a type of CNN called Scale Invariant Faster Region-based CNNs to identify movements of people in the night. This model has enhanced detection accuracy when compared to other CNN-based methods, it provides almost real-time performance and efficiency across datasets. Importance is given to improve the systems ability to detect constantly changing weather conditions, which will enhance safety in self-driving cars. Zhao et al [13] have done an extensive review of the autonomous driving system. This study provides an overview of the key elements and technologies being used in autonomous driving. Emphasis was given on how sensors and artificial intelligence allow cars to safely and effectively navigate their surroundings. Importance was given to the effect of autonomous driving on transportation while addressing current issues. Reda et al [14]have done a comprehensive review on the six steps of the Autonomous Driving System (ADS), with a particular emphasis on path planning. It divides path planning strategies into three categories namely classical , machine learning and lastly meta-heuristic optimisation techniques. The study highlights the significance of adaptive parameter tuning and discovers a tendency towards hybrid algorithms. Importance was given to improve old methods by combining them with other approaches.

Elfahim et al [15] have brought it to notice that there is inefficiencies in emergency medical services (EMS) and speed up dispatch times. They have designed a solution by creating a model-free reinforcement learning strategy, The EMS procedure is defined as a Markov decision process, and emergency response actions are optimized. Efficient hospital transfers and shorter patient wait times are demonstrated by the suggested approach, which has been tested using simulations. Lin et al [16] present a domain-adversarial reinforcement learning technique to enhance the transferability of vision-based autonomous driving models from virtual to physical environments. The model outperforms earlier techniques. Importance was given to research for reinforcement learning and domain adaptation in order to overcome supervised learning approaches.

Sivakumar et al [17]develop a system GRAND, which makes use of the MO-API algorithm, graph-based feature learning, and reinforcement learning. It has shown remarkable performance in the CARLA simulator; Importance has to be given in the future to construct a physical model for evaluation of real-world information into the estimation system. Ergun [18] has explored in this study the ways to improve the performance of Autonomous Delivery Vehicles (ADV) by using a Multi-Agent Reinforcement Learning (MARL) approach that uses shortest-path data to achieve better efficiency and safety. The end goal is to enhance self-driving systems by allowing multiple vehicles to share and learn from each other's observations. Importance is given to conduct advanced studies and practical applications, such as figuring out the best way to deliver packages while keeping in mind the features of the vehicles.

The above studies highlight advancements in autonomous technology in the domains of racing, emergency response, and delivery vehicles. This study intends to improve autonomous racing car decision-making by addressing issues such as lowering decision latency via reinforcement learning (RL), applying optimised techniques to new situations.

3 Proposed Solution

This section presents the solution discussed in this study. First, the environment is built using Pygame, a Python module ideal for creating GUI and gaming functionalities. Subsequently, several essential modules necessary for training the agent are developed. The Deep Q Network module constructs the DQN and defines the layers and parameters for each layer. The Replay Buffer module creates a buffer that stores the experience tuple (state, action, reward, next state). The Agent module manages the agent's policy, balancing exploration and exploitation through the epsilon-greedy strategy. The Environment module sets up the environment, creates the track and checkpoints, and provides an interface for the agent to interact with. Figure 1 depicts the workflow as described above.

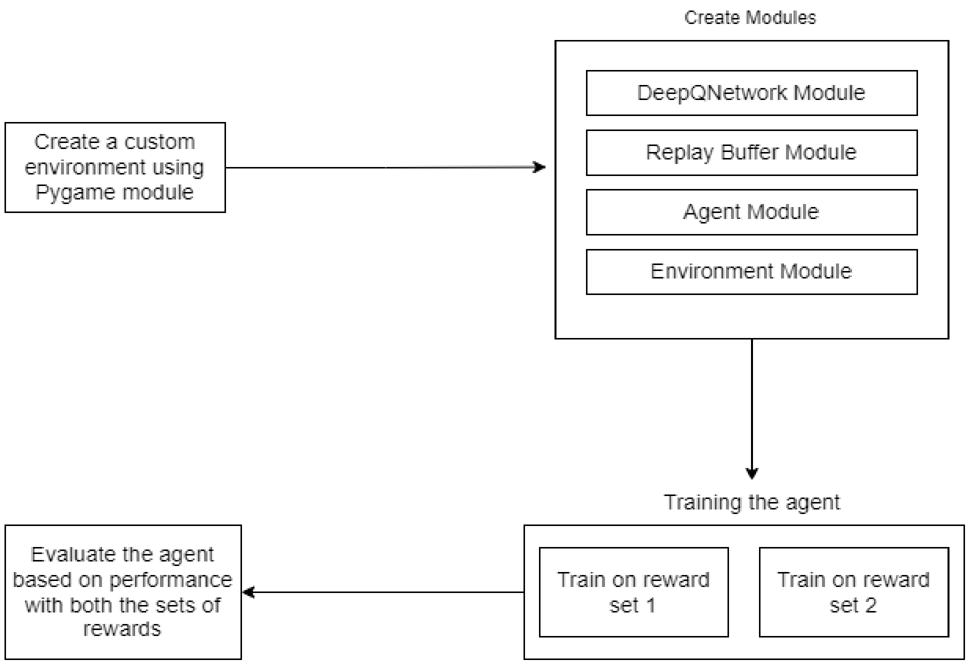

Figure 1. Workflow of the proposed model.

3.1 RL Formulation

The RL formulation provides a structured way to define and analyze RL problems, facilitating the development of algorithms and strategies for solving them. The below is a description of the formulation used in this particular study.

1. **State Space**

$$S : \{(x, y), \theta, \cos(\theta), \sin(\theta), d, i)\} \tag{1}$$

Where,
(x, y) is the agent position
θ is the car angle
$\cos(\theta)$ is the cosine of the car angle
$\sin(\theta)$ is the sine of the car angle
d is the distance travelled
i is the current checkpoint index

2. **Action Space**

$$A : \{\text{Forward, Forward and Left, Forward and Right}\} \tag{2}$$

3. **Reward Space**
The race car's performance will be assessed using two distinct sets of rewards. The details of the two reward sets are as follows:

(a) **First set of rewards $R1$:**

$$R1(s,a) = \begin{cases} \text{Step Reward} & = -1 \\ \text{Reward for crossing checkpoints} & = 100 \times 2^{\text{subsequent checkpoints}} \\ \text{Penalty for going off-track} & = -100 \\ \text{Penalty for backtracking} & = -50 \\ \text{Reward for reaching the goal} & = +1000 \end{cases} \tag{3}$$

(b) **Second set of rewards $R2$:**

$$R2(s,a) = \begin{cases} \text{Step Reward} & = +1 \\ \text{Reward for crossing checkpoints} & = 100 \times 2^{\text{subsequent checkpoints}} \\ \text{Penalty for going off-track} & = -100 \\ \text{Penalty for backtracking} & = -50 \\ \text{Reward for reaching the goal} & = +1000 \end{cases} \tag{4}$$

4. **Environment**
The Pygame environment is set against an expansive 800x600 pixel green backdrop, featuring a sleek black elliptical track with a generous width of 60 pixels. Encircling an inner sanctum with a radius of 200 pixels and an outer boundary stretching to 260 pixels, the track offers a challenging course for the red race car, measuring 50x25 pixels. The 15-pixel-wide start line spans the track's width at its midpoint, marking the beginning of each race. Along the track, ten equidistant white dots act as checkpoints, guiding the car's path and adding strategic depth to the thrilling simulation of speed and precision.

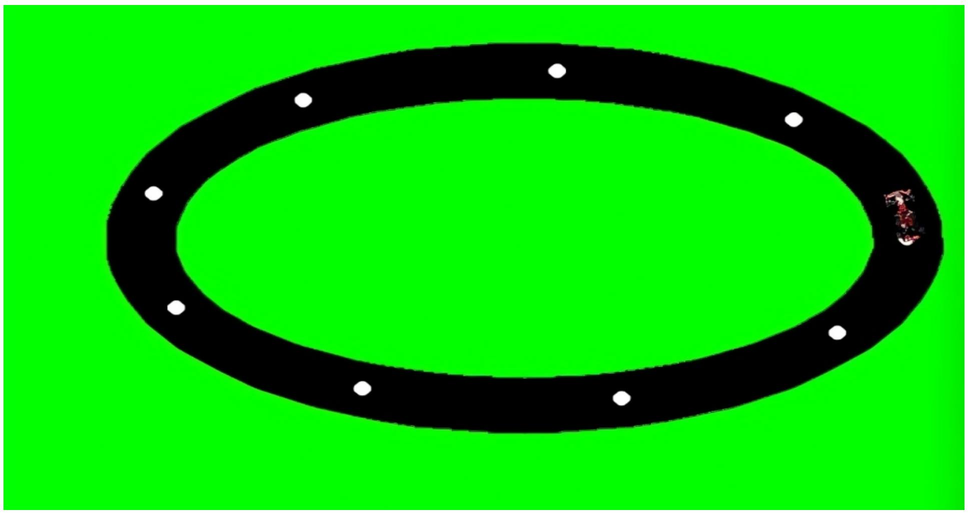

Figure 2. Custom Built Environment With Track.

3.2 Deep Q Network

The DQN design starts with an input layer handling a state space of 6, followed by two dense layers using the Rectified Linear Unit (ReLU) activation function to introduce non-linearity and mitigate the vanishing gradient problem. These layers capture complex patterns in the data. The output layer, with three neurons and a linear activation function, corresponds to the agent's possible actions, producing Q-values representing predicted future rewards. The model includes a 10,000-unit replay buffer, processing batches of 64 samples during training. Key hyper-parameters are an initial epsilon of 1.0 for exploration, decaying to 0.01 with a factor of 0.995 per episode, and a discount factor (gamma) of 0.99 to weigh future rewards. Each episode can last up to 1,000 steps or ends if the agent (race car) leaves the track.

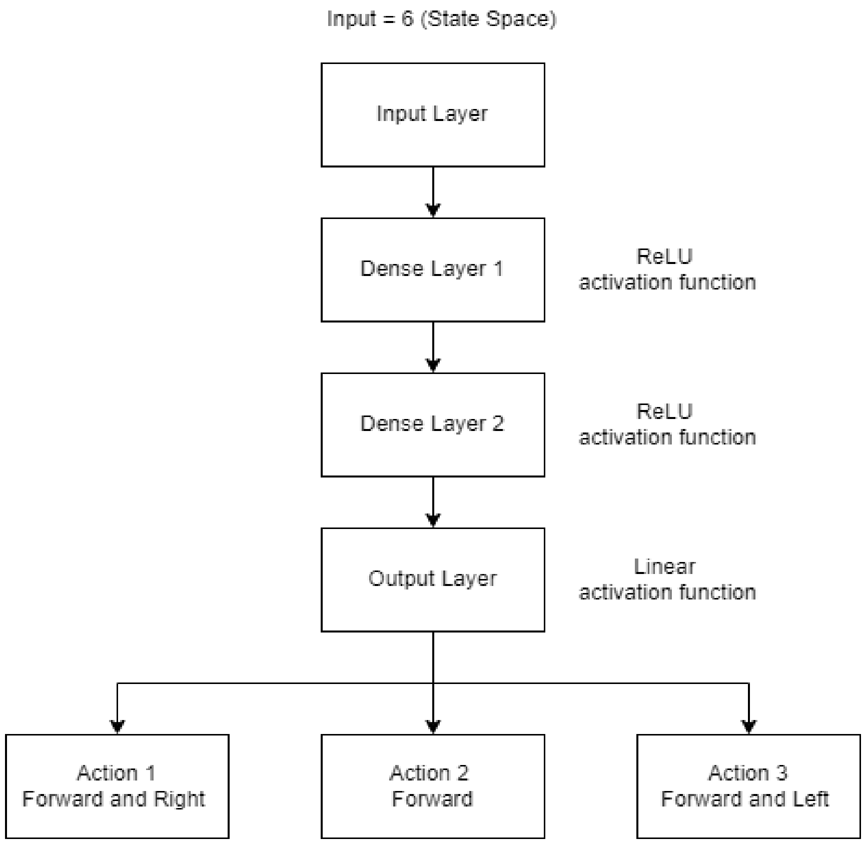

Figure 3. Deep Q Network Architecture.

4 Results and Analysis

The agent(race car) has been trained on two sets of rewards. After training the agent for 10,000 episodes on both the sets of rewards, it was observed that the agent performed better on the second set of rewards. The following Figure 4 depicts a plot for the episodes (plotted on the x-axis) versus average distance (plotted on the y-axis) travelled by the agent for both R1 and R2 sampled for every 500 episodes. R1 is plotted using a dashed red line and R2 using a solid blue line. It can be observed from this plot that R2 works better and the agent is able to learn and travel better distances than R1.

In the initial episodes (0-1000), both reward sets show an increase in the distance traveled, with R2 starting to outperform R1 early on, indicating that the positive step reward helps the car learn to travel further distances faster. During the middle episodes (1000-7000), R2 exhibits steady improvement, peaking around episode 6500, while R1 fluctuates, showing inconsistent performance and generally lower distances than R2. In the later episodes (7000-10000), R2 begins to decline after reaching its peak, suggesting possible over-fitting or new challenges, while R1 remains relatively stable but still under-performs compared to R2. Important milestones include around episode 1000, where R2 significantly outperforms R1, achieving distances over 400 units compared to R1's 200 units; around episode 2500, where R2 crosses the 500 units mark while R1 stabilizes around 300 units; peak performance around episode 6500, where R2 reaches about 700 units, while R1 fluctuates around 300 units, and the end of episodes (10000), where R2 declines to around 400 units, while R1 stabilizes slightly above 200 units.

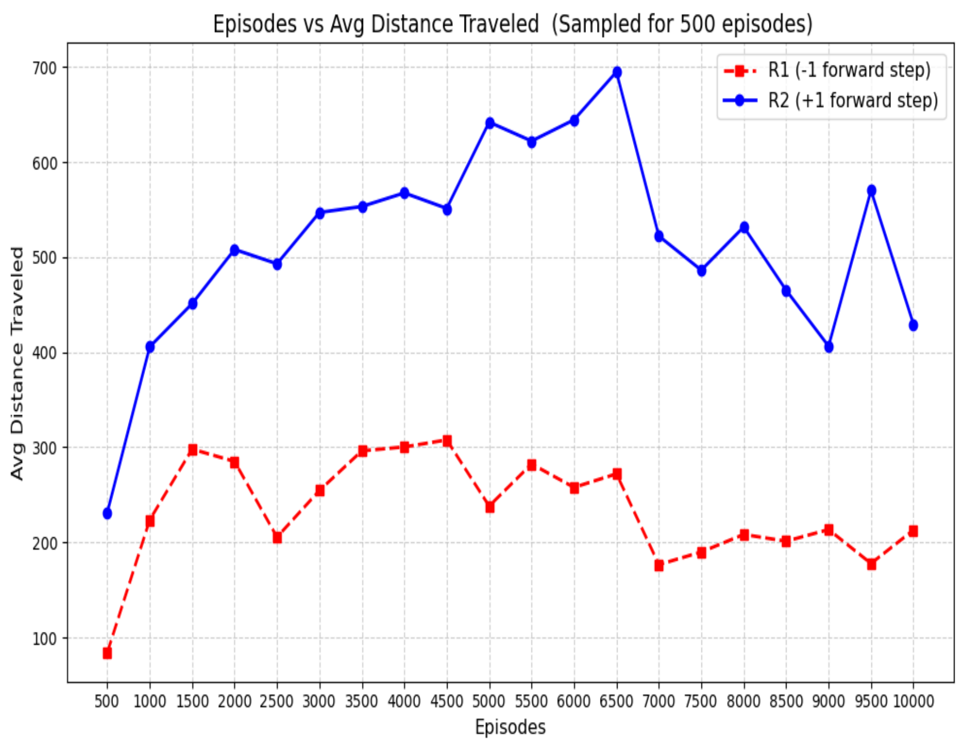

Figure 4. Episode versus Average Distance plot sampled for every 500 episodes.

Figure 5 shows the agent completing one lap, earning a +1 reward for each step. By this point, the agent has completed 5401 episodes, traveled a total of 1216 units, and accumulated a reward of 251. In episode 5401, the agent has finished one lap. Similarly, Figure 6 illustrates the agent at 5043 episodes, but with a -1 reward for each step. In this scenario, the agent is in a comparable episode with a +1 reward per step, having traveled 620 units and earned a cumulative reward of 61. Comparing the two, the agent with a R2 reward traveled a greater distance and accumulated a higher reward over a similar number of episodes.

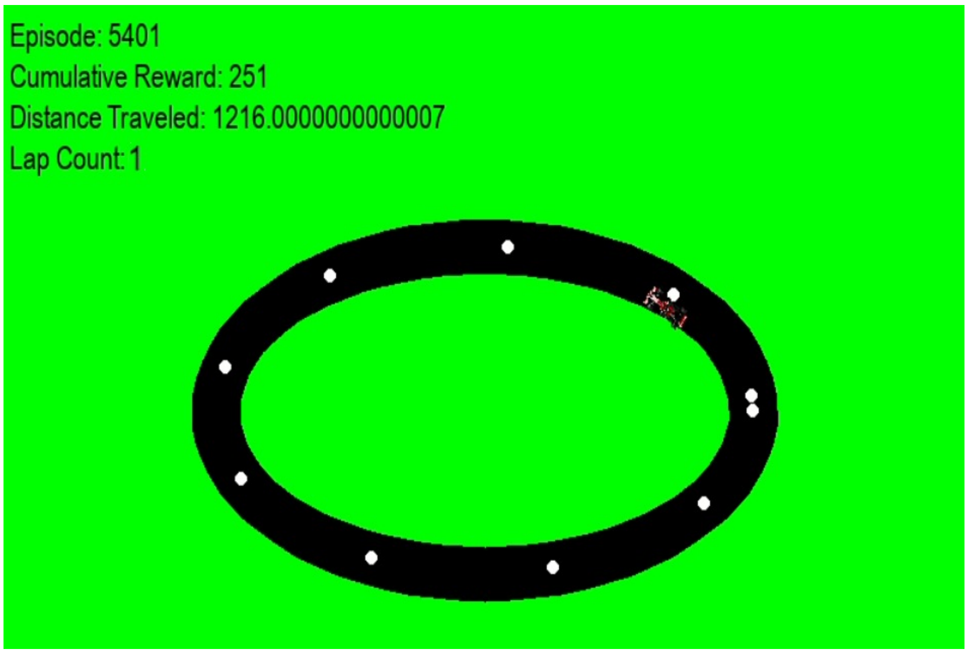
Episode: 5401
Cumulative Reward: 251
Distance Traveled: 1216.0000000000007
Lap Count: 1

Figure 5. Distance covered by the agent at 5000 episodes with R2 rewards.

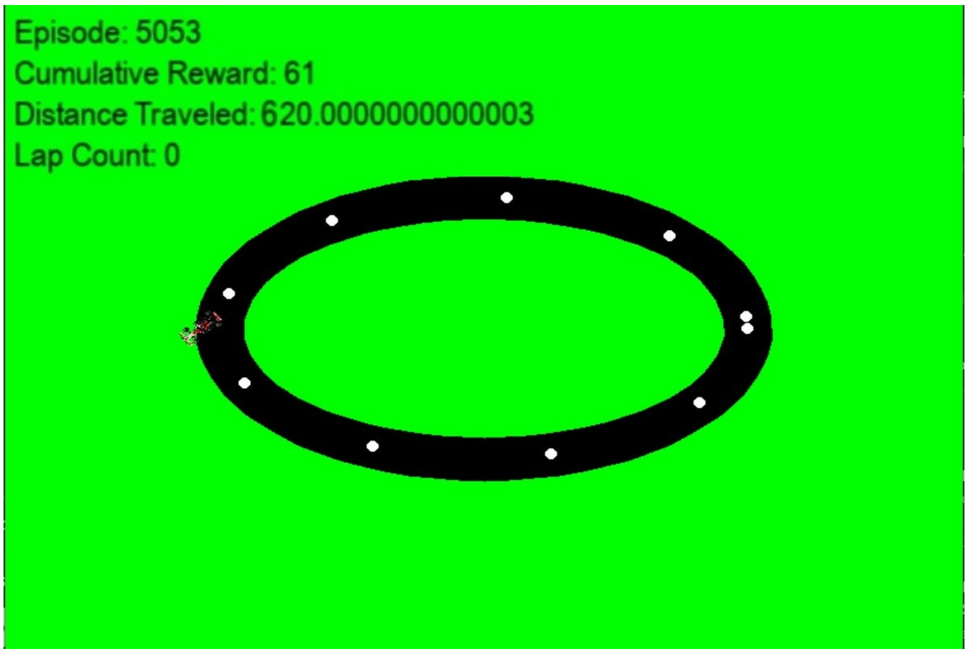
Episode: 5053
Cumulative Reward: 61
Distance Traveled: 620.0000000000003
Lap Count: 0

Figure 6. Distance covered by the agent at 5000 episodes with R1 rewards.

Table 1 Agent's behaviour with both the sets of rewards.

Metrics for evaluting the agent	**R1**	**R2**
Average distance	234.26	516.19
Maximum distance travelled by agent	824	1384
Episode with the maximum distance	6017	5402

Table 1 compares the agent's behavior under two different reward systems, R1 and R2. The average distance traveled per episode is significantly higher with R2 (516.19 units) compared to R1 (234.26 units). The agent can go a longer distance under R2 (1384 units) than under R1 (824 units). The episodes with the longest distance covered by the agent with R1 is 6017 and 5402 with R2. The table clearly illustrates how the two reward systems differ in terms of the agent's performance measures, with the R2 system outperforming the R1 system.

5 Conclusion

Through the study, it is evident that the agent with the second set of rewards (R2) performs better, with a positive step reward. This was facilitated by the effective and sensitive learning which is influenced by the reward structure.Quantitatively, the agent's average distance per episode with R2 was 516.19 units, more than double the 234.26 units achieved with the first reward set (R1). The agent that uses R2 reward system has reached to a maximum average distance of 1384 units at the 5402nd episode, while the agent with the R1 reward system has just made it to 824 units at 6017th episode. This proves that the R2 reward system has not only reached the maximum average distance but also learns faster as it reaches its maximum distance earlier when compared to that of R1.

Future work could include refining hyper-parameters, a task that has the potential to significantly boost performance. This includes tweaking the exploration rate, learning rate, state and action space values. Making the neural network level changes like, employing a Double Deep Q-Networks(DDQNs) might be a worthful upgrade and might lead to more efficient and robust training of the agent. The robustness can also be achieved by designing a more complex neural network that has more layers and its compatible activation functions. The state space being the input for the neural network, increasing its size, by including more metrices that governs the behaviour of the agent will result in a an agent that is more sensitive for the learning and with the environment. Comprehensively, even though the study has extensively analysed, how impactful and crucial it is to design a reward system, yet it packs a lot of future work that can be worked upon to make the study better and more application oriented to the daily life .

6 Compliance with Ethical Standards

Conflict of Interest: Pranave KC, Naga Sai Shreya Kunda, Mayank Pandey and Nippun Kumaar AA all state that they have no conflicts of interest.

Ethical Approval: None of the authors of this article have conducted any studies on humans or animals.

References

[1] Kumaar, A.N., Kochuvila, S.: Mobile service robot path planning using deep reinforcement learning. IEEE Access (2023)

[2] Gupta, A., Anpalagan, A., Guan, L., Khwaja, A.S.: Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. Array **10**, 100057 (2021)

[3] Ignatious, H.A., Khan, M., *et al.*: An overview of sensors in autonomous vehicles. Procedia Computer Science **198**, 736–741 (2022)

[4] Nambisan, A., Chandran, A., Preetha, P., Nair, M.G.: Advanced body controlled safety system for connected and autonomous vehicle. In: 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 507–514 (2021). IEEE

[5] Kumaar, A.N., Kochuvila, S.: Reinforcement learning based path planning using a topological map for mobile service robot. In: 2023 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–6 (2023). IEEE

[6] Rao, P.S., Polisetty, V.R.M., Jayanth, K.K., Manoj, N.S., Mohith, V., Kumar, R.P.: Deep adaptive algorithms for local urban traffic control: Deep reinforcement learning with dqn. In: 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), pp. 592–598 (2024). IEEE

[7] Shafiei, S., Gu, Z., Grzybowska, H., Cai, C.: Impact of self-parking autonomous vehicles on urban traffic congestion. Transportation **50**(1), 183–203 (2023)

[8] Balaji, B., Mallya, S., Genc, S., Gupta, S., Dirac, L., Khare, V., Roy, G., Sun, T., Tao, Y., Townsend, B., *et al.*: Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2746–2754 (2020). IEEE

[9] Remonda, A., Krebs, S., Veas, E., Luzhnica, G., Kern, R.: Formula rl: Deep reinforcement learning for autonomous racing using telemetry data. arXiv preprint arXiv:2104.11106 (2021)

[10] Evans, B.D., Jordaan, H.W., Engelbrecht, H.A.: Safe reinforcement learning for high-speed autonomous racing. Cognitive Robotics **3**, 107–126 (2023)

[11] Mao, C., Liu, Y., Shen, Z.-J.M.: Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. Transportation Research Part C: Emerging Technologies **115**, 102626 (2020)

[12] Farooq, M.S., Khalid, H., Arooj, A., Umer, T., Asghar, A.B., Rasheed, J., Shubair, R.M., Yahyaoui, A.: A conceptual multi-layer framework for the detection of nighttime pedestrian in autonomous vehicles using deep reinforcement learning. Entropy **25**(1), 135 (2023)

[13] Zhao, J., Zhao, W., Deng, B., Wang, Z., Zhang, F., Zheng, W., Cao, W., Nan, J., Lian, Y., Burke, A.F.: Autonomous driving system: A comprehensive survey. Expert Systems with Applications, 122836 (2023)

[14] Reda, M., Onsy, A., Haikal, A.Y., Ghanbari, A.: Path planning algorithms in the autonomous driving system: A comprehensive review. Robotics and Autonomous Systems **174**, 104630 (2024)

[15] Elfahim, O., Youssfi, M., Barakat, O., Mestari, M., *et al.*: Deep reinforcement learning approach for emergency response management. In: 2022 International Conference on Intelligent Systems and Computer Vision (ISCV), pp. 1–7 (2022). IEEE

[16] Lin, S., Li, Y., Fang, Y.: Adversarial reinforcement learning for steering cars from virtual to real world. In: Chinese Conference on Image and Graphics Technologies, pp. 361–372 (2023). Springer

[17] Sivakumar, A., Gopalakrishnan, S., Ramachandran, A., Chitrakala, S.: Grand: Graph laplacian-based reinforcement learning for autonomous navigation decisions. In: 2023 IEEE World Conference on Applied Intelligence and Computing (AIC), pp. 204–209 (2023). IEEE

[18] Ergün, S.: A study on multi-agent reinforcement learning for autonomous distribution vehicles. Iran Journal of Computer Science **6**(4), 297–305 (2023)