sustainability

MDPI

Article

Improving the Performance of Autonomous Driving through Deep Reinforcement Learning

Akshaj Tammewar [1], Nikita Chaudhari [1], Bunny Saini [1], Divya Venkatesh [1], Ganpathiraju Dharahas [1], Deepali Vora [1,*], Shruti Patil [2], Ketan Kotecha [2] and Sultan Alfarhood [3]

[1] Symbiosis Institute of Technology, Pune Campus, Symbiosis International (Deemed University) (SIU), Lavale, Pune 412115, India
[2] Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis Institute of Technology, Pune Campus, Symbiosis International (Deemed University) (SIU), Lavale, Pune 412115, India
[3] Department of Computer Science, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia
* Correspondence: deepali.vora@sitpune.edu.in

Abstract: Reinforcement learning (RL) is revolutionizing the artificial intelligence (AI) domain and significantly aiding in building autonomous systems with a higher level comprehension of the world as we observe it. Deep learning (DL) facilitates RL to scale and resolve previously intractable problems, for instance, allowing supervision principles designed for robots to be acquired directly from visual data, developing video game proficiency from pixel-level information, etc. Recent research shows that RL algorithms help represent problems dealing with high-dimensional, unprocessed data input and can have successful applications in computer vision, pattern identification, natural language analysis, and speech parsing. This research paper focuses on training a simulation model of a car to navigate autonomously on a racetrack using RL. The study explores several fundamental algorithms in Deep RL, namely Proximal Policy Optimization (PPO), Deep Q-network (DQN), and Deep Deterministic Policy Gradient (DDPG). The paper documents a comparative analysis of these three prominent algorithms—based on their speed, accuracy, and overall performance. After a thorough evaluation, the research indicates that the DQN surpassed the other existing algorithms. This study further examined the performance of the DQN with and without ε-decay and observed that the DQN with ε-decay is better suited for our objective and is significantly more stable than its non ε-decay counterpart. The findings from this research could assist in improving the performance and stability of autonomous vehicles using the DQN with ε -decay. It concludes by discussing the fine-tuning of the model for future real-world applications and the potential research areas within the field of autonomous driving.

Keywords: autonomous driving vehicles; reinforcement learning; DQN; DDPG; PPO

Citation: Tammewar, A.; Chaudhari, N.; Saini, B.; Venkatesh, D.; Dharahas, G.; Vora, D.; Patil, S.; Kotecha, K.; Alfarhood, S. Improving the Performance of Autonomous Driving through Deep Reinforcement Learning. *Sustainability* **2023**, *15*, 13799. https://doi.org/10.3390/su151813799

Academic Editors: Guangliang Cheng, Ting-Bing Xu and Xiangtai Li

Received: 5 August 2023
Revised: 11 September 2023
Accepted: 12 September 2023
Published: 15 September 2023

1. Introduction

With the rise of Deep Learning (DL), Reinforcement Learning (RL) has become a noteworthy learning architecture suitable for learning complicated methodologies in high-dimensional environments [1]. Recently, RL, also referred to as adaptive or approximate dynamic programming, has surfaced as a vital device for strategizing complex decision-making problems in high-dimensional environments [2]. Looking ahead, Deep Reinforcement Learning (DRL) will play a crucial part in several fields like transportation, robotization, finance, pharmacy, and other areas where less human interaction is required [3]. DRL has also become increasingly popular in recent times and has been applied to solve Markov Decision Processes (MDPs), where the aim is to find the most effective strategy for an agent to choose actions that maximize a reward function. This has attained relatively successful results in working out computer games, simple decision-making systems, and self-driving car technologies. Autonomous driving (AD) has been in the headlines for

over a decade and remains a fascinating topic to researchers, robotics organizations, and automobile companies.

Car racing has become a prominent domain for the evaluation and enhancement of autonomous driving through Reinforcement Learning (RL) methods. The controlled environment of closed tracks ensures safety and allows focused algorithm refinement. Rich sensory input and clear objectives facilitate the design of effective reward functions. Moreover, standardized racing tasks enable straightforward comparisons, rapid iterations, and potential real-world driving skill transfer. These factors collectively contribute to advancing autonomous vehicle technology responsibly and efficiently. Hence, in light of these advantages, the research exclusively focuses on car racing as the chosen domain to test and evaluate various RL methods for autonomous driving.

This paper gives an overview of RL techniques for addressing a dynamically changing environment like the driving scene. The objective is to showcase a model trained on a DRL algorithm and make it capable of navigation using the transmitted control commands to the vehicle. This process should operate seamlessly to follow a predetermined path. The project aims to train our model to complete a track in the racing game using DRL methods. The training procedure encompasses the iterative refinement of the lone training agent's performance on what is, initially, a randomly generated racetrack. The track remains consistent throughout the successive episodes. Moreover, the training regimen does not incorporate any provisions to account for obstacles nor other cars on the track. The objective is to maximize tile coverage while remaining purely on track, thus mirroring a paradigm akin to a time trial situation. The environment utilized in this study is CarRacing-v2, a 2D autonomous vehicle environment. An agent will be trained to learn this track using techniques of RL. One of the simulation environments used for this purpose will be OpenAI's Gym [4]. This study plans to use positive reinforcement to incentivize the vehicle to stay on the desired path. The article concludes by evaluating various RL systems that have been implemented and assessing the practicality of current approaches.

1.1. Structure

1.1.1. Reinforcement Learning

RL is a subdivision of the broader field of Machine Learning (ML) that deals with the challenges of automatically determining optimal decision-making policies over time. RL agents are well suited to respond and change according to varying operating conditions frequently. Single-agent RL research extensively works on developing algorithms that assume the relaxation of the stationary environment model. It also factors in other essential steps to make a successful model, like handling datasets and data sizes and anticipating future outcomes by utilizing input information. RL is the most effective for optimal trial-and-error problems where model-free and model-based learning approaches are applied [5]. The agent in RL takes decisions periodically after every new instance it faces by observing. This is performed by observing the rewards and automatically conditioning itself to realize the best policy [3]. This process is outlined in Figure 1.

Deep Neural Networks combine Artificial Neural Networks (ANN) with a reinforcement learning architecture. This facilitates the agents to learn the most effective actions of an environment and apply them to realize their defined goals. In other words, it combines function approximation and target optimization by associating state-action pairs with anticipated outcomes.

Thus, DRL-based algorithms can replace tabular approaches to estimating state values with an approximation function. The agent then uses said function to extrapolate state values it has partially observed or unobserved before by taking the aid of the values resembling states. This is accomplished with the aid of the values resembling states. The combination of DL methodologies and RL algorithms has exhibited its capability to address some of the most challenging tasks of AD. It is primarily used to make crucial decision-making and planning steps in the cars to ensure a safer and more leisurely drive.

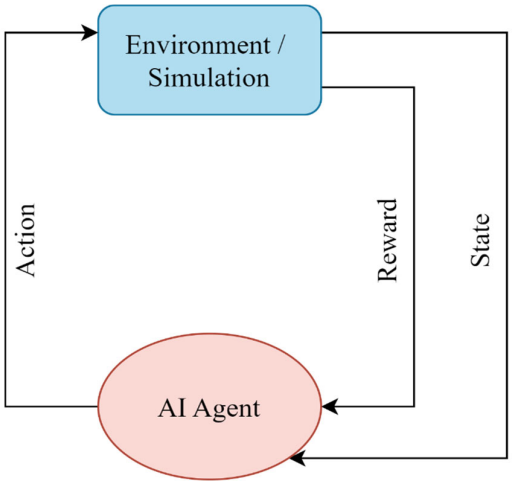

Figure 1. Reinforcement Learning Procedure.

1.1.2. Autonomous Driving

Autonomous cars continue to dominate the headlines; for more than a decade now, researchers, robotics organizations, and automobile companies are all interested in the domain of self-driving cars. An autonomous vehicle (AV), frequently used interchangeably with self-driving vehicles, can sense its surroundings and make decisions without human involvement. Figure 2 highlights the components of ADs. Autonomous driving plays a vital part in cracking congestion and mobility challenges like traffic congestion, roadblocks, and accidents in busy areas. It will radically reshape traveling in the years to come with promising solutions. Some important factors to focus on involve striking a balance between being prepared for unanticipated behavior from other cars and pedestrians whilst not obstructing the traffic flow by becoming overly cautious. Every year, around 1.25 million people lose their lives to automobile accidents worldwide. The increased usage of autonomous cars aims to reduce human error and intercept millions of fatalities that would otherwise be preventable [6].

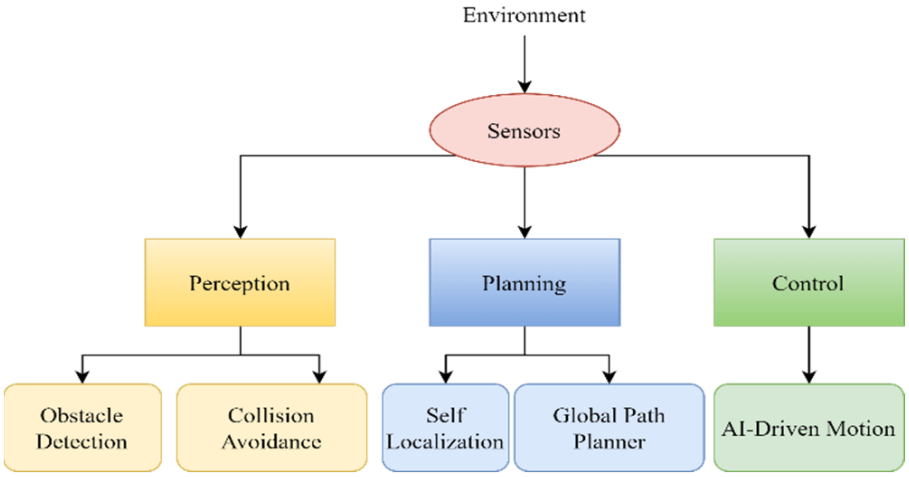

Figure 2. Components of autonomous driving vehicles.

In a typical autonomous vehicle (AV) architecture, the control layer refers to a group of operations responsible for executing vehicle control and navigation [7]. This paper investigates integrating AI methodologies into the control layer by implementing control-based DRL algorithms for autonomous vehicle navigation. The ideal way to model driving is as a succession of sequential choices accompanied by periodic environmental feedback.

Thus, RL is the optimal technique for the agent to gradually improve control policies through experience and system feedback.

The challenge of comprehending human drivers is quite daunting. It incorporates feelings rather than reasoning. A system that can forecast the behavior of other road users can be crucial for improving road safety, but since predicting the continuous action of vehicles or pedestrians around an entity is challenging, it is an active area of research. Methodologies for RL have shown some promise in the treatment of challenging control issues. In this paper, the authors have explored and implemented three reinforcement algorithms, namely, Deep Deterministic Policy Gradient (DDPG) [8], Proximal Policy Optimization (PPO) [9], and Deep Q-Learning (DQN) [10]. The problem aims to navigate the car as quickly as possible around the randomly generated racetrack; to achieve this, reward functions are naively modified to favor actions such as maximizing acceleration action for better car performance. These algorithms' performances and training times, focusing on early training stages, are compared to evaluate the best one, especially in low computational resource environments. This paper discusses an assemblage of algorithms in detail, along with their categorization and relative merits and demerits. The approaches used are not limited by the stationarity supposition of the autonomous agents and can be adjusted to different operating conditions. This is achievable through two primary methods: minimizing the rewards lost or maximizing the rewards gained during learning by the RL agent. This enables the agent to find a suitable policy for functioning, leading to the effective operation of the underlying system [11].

1.2. Contribution of Work

The primary aim of our research paper is to analyze the performance of various reinforcement learning algorithms within the context of a brief training period. Moreover, this study further undertakes an investigation into the influence of epsilon decay on the performance of DQN algorithms when confronted with the constraints of a limited training period, particularly emphasizing its ramifications on model performance during the initial phases of training.

Additionally, the paper analyzes the area of RL from a computer science perspective aimed at individuals with a background in Artificial Intelligence and Machine Learning. It covers both historical and recent research conducted in the field. This document surveys the literature and cohesively presents these algorithms. The authors intend to train a model car using DRL to complete laps of randomly generated tracks. This study will help us investigate the performance of models that currently exist in the research sphere and might have been often overlooked or not been worked upon. Our research aims to train an autonomous vehicle to navigate effectively within a specified environment, become accustomed to said environment, and make decisions based on its previous actions to complete the race around the track as swiftly as possible. Various RL techniques have been explored to train the model in the given environment and to achieve the target in the shortest possible time. The efficiency of such models will be tested against the ones that are largely implemented to discover if they have a better yield. The obtained findings can be used in autonomous driving applications, including optimizing the agent for car racing or gaming, urban driving, congestion control, lane-switching, and the integration of autonomous vehicles in society.

1.3. Paper Organization

The paper details everything that went into this project and all the specifications of the criteria selected and respective outcomes. The problem statement, introduction to the topics of RL, DRL, AD, motivation, contribution, and paper organization are all found in Section 1. Section 2 thoroughly reviews 20 research publications in connection with the project's problem statement. This survey has been summarized and its conclusions were used to create a custom framework that caters to our specific needs. Section 3 describes the various algorithms and approaches available for use with our problem statement. It

gives a detailed overview of the three main categories of DL algorithms: value-based, policy-based, and state actor-critic, along with the various DRL methods. The platforms, frameworks, and tools needed to carry out the project are listed in Section 4, along with details on the data our model was trained on. The methodology used, the implementation procedure, including the data collection and preprocessing steps, the models utilized, the model design description, and the data flow diagrams are included in this section. All the outcomes that followed our project's successful execution are detailed in Section 5. It comprises the results and observations of the research. Section 6 explains this project's purpose, conclusion, and future scope.

2. Literature Survey

Understanding the trends and patterns in a study offers a data-driven perspective and enables a more complex and fact-based assessment of the latest developments within the context of the literature survey.

RL techniques for autonomous driving were incepted in the early 2000s as a potential means of addressing the complicated decision-making processes involved in self-driving cars. The subject, however, experienced a surge in interest in recent years. As shown in Figure 3, research and development in this area have significantly increased between 2019 and 2020 as more intricate applications are now possible due to the growing capacity of AI to process massive amounts of data and the accessibility of more advanced hardware. As a result, the field has rapidly expanded and significant progress has been made in this area.

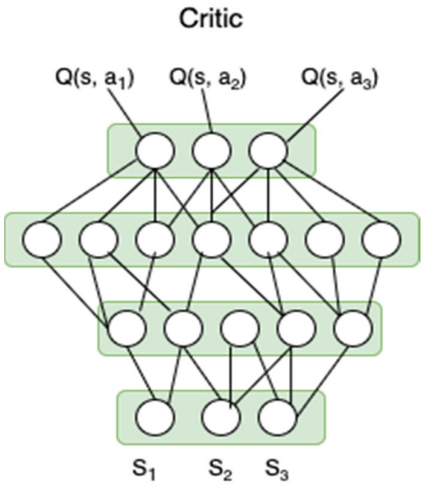

Figure 3. DQN Network Architecture [12].

In the realm of RL and autonomous driving, a diverse array of research has yielded valuable insights and advancements. Zhu et al. [13] introduced a velocity control framework utilizing a Deep Deterministic Policy Gradient, resulting in remarkable improvements in driving safety and efficiency, notably reducing the Time-To-Collision (TTC) and jerk values. Meanwhile, Chen et al. [14] employed the DDPG algorithm and hierarchical reinforcement learning to achieve quicker and safer lane change behaviors. Their innovative approach enabled more assertive lane changing, emphasizing the potential for safer and more efficient autonomous vehicles. In a bid to reduce instructional time and alleviate congestion in complex multi-lane scenarios, Yingfeng et al. [15] divided the autonomous driving task into sub-policies and recommended a responsive noise exploration approach, which exhibited promising results in terms of decision-making efficiency during heavy traffic congestion.

Wolf et al. [16] ventured into the realm of Deep Q-Networks (DQN) to maintain lane discipline while completing laps, highlighting the effectiveness of action-based reward systems over distance-based ones. Ghimire et al. [17] combined Deep Q-Networks with rule-based constraints, enhancing both low-level trajectory monitoring and high-level lateral decision making. Their approach fostered a safer lane shifting strategy.

Arvind et al. [18] took a different path, utilizing Multilayer Perceptron Neural Networks for obstacle detection and action prediction. Their work contributed to better comprehension of complex urban scenarios, particularly involving static obstructions. On a parallel track, Gopinath et al. [19] utilized Behavioral Cloning and Convolutional Neural Networks (CNNs) for steering angle prediction, a significant step in the development of self-driving technology, enabling vehicles to gain environmental awareness.

Yildirim et al. [20] innovatively integrated intention projections of other cars using DQN approaches and a Monte Carlo Tree Search decision-making model, substantially reducing the risk of collisions. Sallab et al. [21] adopted an end-to-end Deep RL pipeline with Recurrent Neural Networks (RNNs) to enable autonomous driving with partial observability, successfully achieving lane keeping behavior in challenging scenarios. Future work across these studies includes enhancements in traffic rule adherence, addressing dynamic environments, and exploring more sophisticated reward functions, highlighting the versatility of RL techniques in the complex landscape of autonomous driving research. These contributions collectively advance our understanding and capabilities in the realm of RL and autonomous driving, paving the way for safer, more efficient, and smarter autonomous vehicles.

In the context of extending the application of Reinforcement Learning (RL) from simulation to reality, two noteworthy studies have made significant strides. Jakubowski et al. [22] employed simulated RL techniques, integrating U-Net for scene semantics segmentation while employing Proximal Policy Optimization (PPO) trainers to develop a driving system capable of operating a full-size real-world car. Their goal was to ascertain whether simulator data could effectively be utilized to construct a real-world driving system. The results indicated noisy trial outcomes with suboptimal indicators of real-world performance. However, promising improvements were observed with regularization and augmentation techniques, signifying potential enhancements in system efficiency in future work, which may involve dynamic randomization and the integration of model-based methods. On a related note, Candela et al. [23] focused on implementing multiagent autonomous driving (AD) regulations in real-world scenarios, comparing the efficacy of these policies with traditional rule-based methods. They utilized the domain randomization levels of the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm, and the results indicated a distinct advantage of multiagent policies over the rule-based approach, which struggled to adapt to erratic agent behavior. The real-world performance exceeded expectations and demonstrated robustness even without varying domain randomization levels.

In the domain of Urban Driving, a series of impactful studies have harnessed Reinforcement Learning (RL) to address various facets of driving performance and safety. Li et al. [24] introduced a deep RL framework aimed at enhancing traffic efficiency, demonstrating its effectiveness in tasks such as navigating ring roads and adaptive lane changes. In a parallel effort, Xiong et al. [25] tackled collision avoidance by integrating deep RL with safety-oriented control mechanisms, successfully preventing accidents, particularly in complex scenarios like curve negotiation. Dagdanov et al. [26] proposed a continuous learning approach to improve driving performance, significantly reducing instances of stuck agents and blocked cars through imitation learning for fault scenarios. Meanwhile, Durmus et al. [27] devised a continuous learning pipeline with periodic rare-event tests, effectively reducing system failures and accidents during simulations. These collective endeavors underscore the versatility of RL in addressing critical challenges within urban driving, with a common thread of enhancing traffic management, collision avoidance, and overall safety.

The research identified notable limitations, which include:

- It is necessary to examine and optimize the transferability of simulations to actual driving systems to ensure their seamless and safe incorporation into the real world.

- Understanding and successfully mapping the intents of surrounding entities. Developing thorough knowledge of the different behaviors and intents of pedestrians and other vehicles requires further investigation.
- Several limits arise in the modelling of environmental features to accurately recreate real-world conditions and sensor responses.

The primary and distinctive contributions of this research are as follows:

- Train an autonomous vehicle to complete a race across randomly generated tracks while effectively navigating and making judgments in a given environment using DRL.
- The paper highlights the performance of the DQN with and without ε-decay demonstrating that the DQN with ε-decay is more suitable for the specified objective and exhibits greater stability.
- Systematic presentation of recent advancements in DRL for AD along with efforts to integrate the domain into reality, enhancing accessibility and clarity within the field.
- The obtained findings have practical implications in various real-world applications, including optimizing autonomous agents for car racing, thereby contributing to advancements in the field.

3. Algorithms

Reinforcement Learning (RL) is crucial for RL agents operating in dynamic environments that require adaptive responses. The research in RL emphasizes the development of algorithms that go beyond fixed environmental models, utilizing model-free learning and model-based methods to enable trial-and-error learning for optimal outcomes. RL involves managing datasets of varying sizes, forecasting future outcomes from input data, and enabling agents to adapt their strategies by monitoring rewards, making it versatile across domains like robotics, medical treatment, education, and game development. RL's primary objective is to optimize policies for each state, understanding opponents' capabilities and maximizing rewards, often represented as coins, within diverse applications.

3.1. Value-Based Models: Deep Q Network

Q-learning, a model-free Temporal Difference (TD) algorithm, iteratively refines state-action pair utility estimates, theoretically converging to optimal values within a Markov Decision Process (MDP) with sufficient sampling. It maintains convergence regardless of initial Q-value choices, making it versatile for estimating optimal Q-values from various initial action value function estimates [28]. DQN, a Deep Q-Network variant, leverages Deep Neural Networks (DNNs) to approximate the Q function in high-dimensional state spaces without relying on handcrafted features or context-specific details [5]. Introduced by DeepMind in 2015, DQN bridges Deep Learning (DL) and Reinforcement Learning (RL), offering an efficient alternative to memory-intensive Q-value approaches. Figure 3 illustrates the DQN network architecture [5,28].

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q\left(s',a\right) - Q(s,a)\right] \quad (1)$$

$$Q(S_t,\ A_t) \leftarrow Q(S_t,\ A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1},\ a) - Q(S_t,\ A_t)] \quad (2)$$

where:

$Q'(S_t, A_t)$: new Q value for that state and that action
$Q(S_t, A_t)$: former Q value estimation
α: learning Rate
R_{t+1}: reward for taking that action at that state; Immediate Reward
γ: discount Rate
maxaQ(S_{t+1}, a): maximum Q value from the next state
γmaxaQ(S_{t+1}, a): Discounted Estimate Maximum Q Value of Next State

3.2. Policy-Based Models: Proximal Policy Optimization

Value-based and policy-based methods differ in their approaches to optimality: value-based techniques focus on cumulative rewards to derive optimal value functions indirectly, while policy-based approaches adapt policies directly for efficient actions. Policies, often neural networks, are optimized using gradient descent to maximize expected rewards, yielding stochastic or deterministic policies. Deterministic policy gradients are suited for continuous action spaces, offering a model-free approach for tracking action-value function gradients [29]. The REINFORCE algorithm [30] directly computes discounted cumulative rewards within individual episodes in OpenAI's Car Racing environment, updating model parameters through gradient ascent with a learning rate for stable incremental updates while identifying state-action pairs yielding optimal outcomes, preserving all references [29,30].

In Figure 4, S denotes state, u implies action taken by the agent, and S_{t+1} says the environment changes the state with reward r_{t+1}. g is the equation to adjust the trajectory based on total rewards.

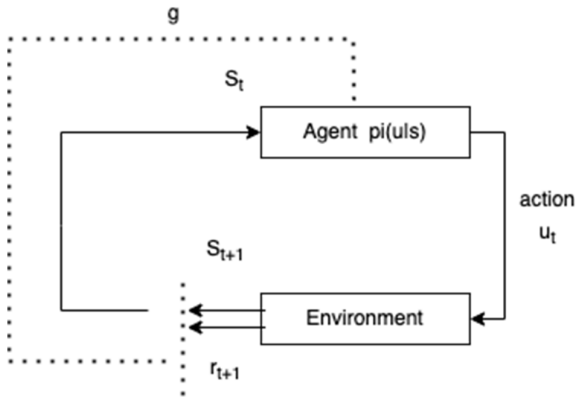

Figure 4. DQN diagram [31].

Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a policy gradient algorithm that updates an existing policy incrementally to enhance specific parameters [32]. It employs a bounded update range, avoiding extreme changes between old and new policies. Like other policy gradient methods, PPO computes output probabilities in the forward pass and improves decisions or probabilities via gradient calculations in the backward phase [33]. The main challenges that were faced during this study are listed below:

- Extremely slow training. While different algorithms would train at different rates, to evaluate them against each other in a manner as far as possible, training time was limited to 6–7 h. Ideally, the training period should be longer to gain a better understanding and insight into the selected models.
- Deprecated libraries. Various libraries were deprecated, causing a significant number of issues during implementation that required rectification. Despite this, various deprecation warnings still occur.
- Hardware limitations. Limitations in computational resources limited training due to the slow training speed and prevented investigating longer training periods and investigating the impact of other hyperparameters apart from the ones selected.

3.3. State Actor-Critic Methods

SAC methods, which integrate policy-based and value-based advantages in temporal difference learning while discarding traditional value functions, employ Actor and Critic networks to assess actions, approximating rewards and validating action outcomes. The DDPG, a model-free, off-policy approach for continuous action spaces, merges actor-critic strengths using deep neural networks. The DDPG introduces noise for exploration [21], employs replay buffers for stability, and incorporates target networks for Q-value correction

post-action selection, distinguishing it from Q-learning. In contrast to discretization methods for managing motion actions, DRL approaches like the DDPG learn policies directly without explicit discretization. However, the DDPG poses challenges in validation and reproducibility for continuous actions, varying implementations leading to inconsistent performance and generalization [34], catastrophic forgetting resulting in large fluctuations in training [35]. A3C [34], employing asynchronous gradient descent with concurrent environment instances, enhances training stability through parallel agents, while A2C [34], its synchronous counterpart, balances exploration-exploitation with a focus on uncertain value estimate states, often measured using entropy, and both demonstrate comparable performance in various scenarios. Figure 5 depicts a brief overview of the working of the DDPG algorithm.

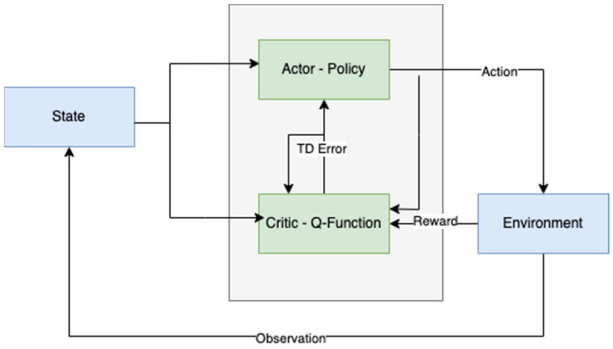

Figure 5. Schematic overview of the DDPG algorithm.

3.4. Deep Reinforcement Learning (Drl)

Deep Reinforcement Learning (DRL) integrates the power of deep learning and reinforcement learning for effective decision making in tasks with high-dimensional input perception, such as video game proficiency and robotics control policies [36]. Prominent algorithms in DRL, including DQNs, Double DQNs, Dueling DQNs, DRQNs, DDPGs, and TRPO, have addressed challenges like the "curse of dimensionality" through various techniques [13,37,38]. However, DRL faces interpretability issues, necessitating solutions such as interpretable policies with hard constraints [32], representation learning [39], incorporating auxiliary tasks [40] and contrastive unsupervised representation learning [41]. These advancements in deep learning also benefit other fields like self-supervised learning [42], hierarchical RL, multiagent RL, and imitation learning, contributing to the development of autonomous driving applications. In our study, we employ a DQN, a key component of DRL, to train a virtual car in a simulated environment, highlighting its relevance in autonomous driving research.

4. Methodology

The primary methodology of this system is to help a car in a virtual environment achieve autonomous driving. Firstly, as the system is based on a virtual computer-generated environment, the environment used is Open AI's Gym's Car Racing Environment.

4.1. Openai's Gym's Car-Racing Environment

One of the most famous APIs for RL is OpenAI's Gym. It is built in Python. It provides a plethora of test environments with shared interfaces for general algorithms. Gym makes use of the agent-environment loop. An agent performs a specified action in an environment and observes the changes in the environment's state, where each action-response exchange is referred to as a timestep. On an agent's selected action, a positive or negative reward may be awarded. If progress is made towards a specified goal it is positive, and vice versa. The end goal, therefore, is to maximize the cumulative reward. This procedure is represented in Figure 6.

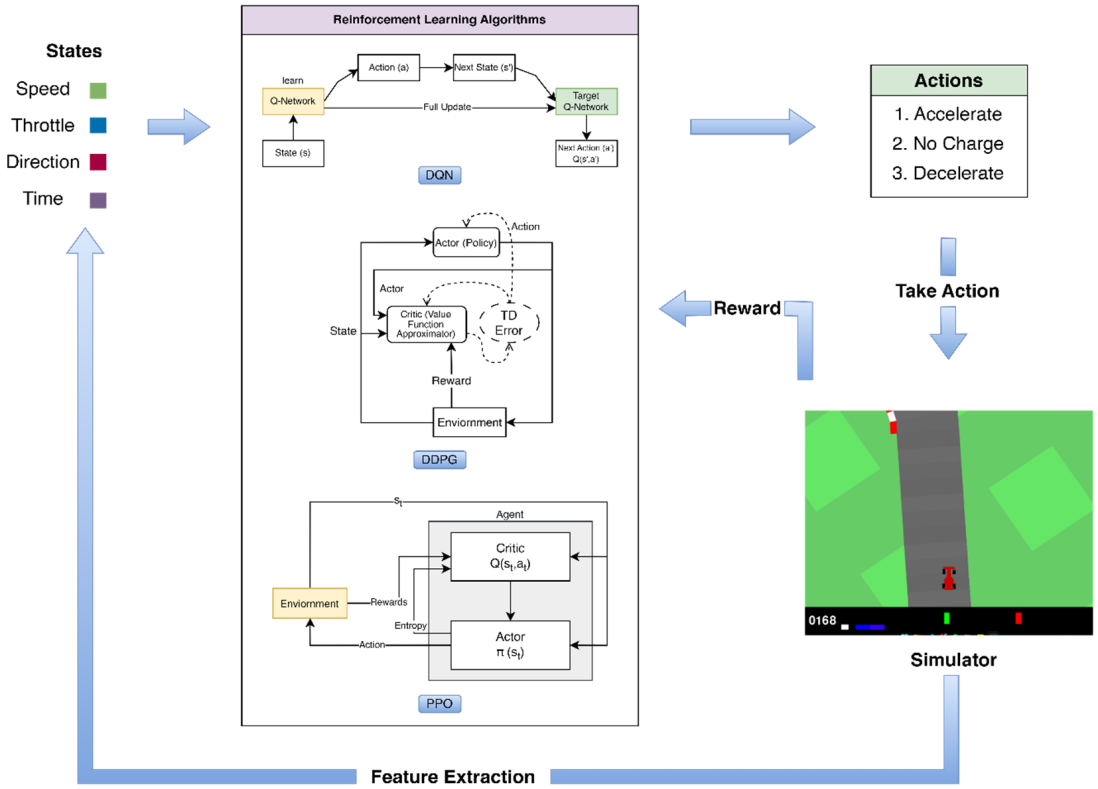

Figure 6. Implementation.

Car Racing makes use of Box2D-based physics and PyGame-based rendering. Box2D is a rigid body simulation library for game development that enables realistic object motion and makes game environments more interactive. Box2D environments have become a popular benchmark owing to their simplicity and extensive configurability.

Car Racing provides a top-down view of a randomly generated racetrack. Figure 7 shows sample images of the environment rendered. Gym's environment provides two spaces, namely, action space and observation space. These describe the valid format of the action and state parameters for the environment. In Car Racing, the default observation space is a state of 96 × 96 pixels in RGB, and hence the observation shape is (96, 96, 3). Two possible action spaces depend on whether or not it is discrete. In the case of discrete, there are five actions: do nothing, steer left, steer right, gas, and brake. In the case of continuous, there are three actions: steering (−1 to 1), gas (0 to 1), and brake (0 to 1).

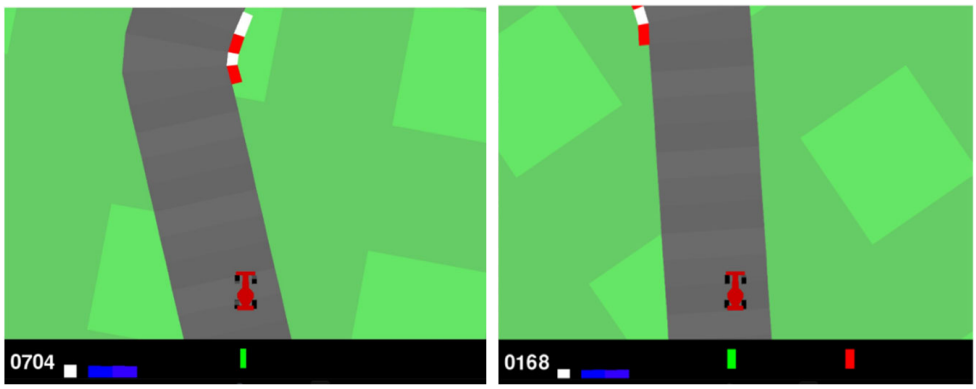

Figure 7. Car Racing—V2 Environment.

The car at rest in the center of the track is the starting state of this methodology. True speed, ABS sensors of four number, steering wheel, and gyroscope are the additional indicators at the bottom which provide additional information.

The reward per frame is -0.1 and for each visited track tile, it is +1000/N; N stands for the number of visited track tiles. The episode terminates when all the track tiles are visited. A mechanism is also in place to consider the car going too far off the track, in which case it receives a -100 reward and dies. The environment is regarded as solved once the agent achieves an average score of 900 or above over 100 consecutive games.

4.2. Dataset Preprocessing

Data preprocessing follows a certain number of steps taken to transform unprocessed data into a "clean" and "tidy" dataset before using it in our models [28]. This step is generally directly followed by the data extraction phase, as most data collected need to be processed so that the models can train on the best version of data to learn only the most significant results and give the highest accuracy. The most common preprocessing step that is generally needed for all computer vision problems is resizing and re-scaling the images. For the problem requirements, the dimensions are resized to 96×96 so that it fits the grid perfectly. The next step was to convert the data into grayscale so that it is easier for the model to focus on only the most significant features present in the dataset and train on them. This step helps in reducing the noise and disturbance in the training process and ensures good model accuracy.

The proposed system works according to the following procedure. Firstly, feature extraction is implemented to identify the virtual car racing environment features such as speed, throttle, and direction. Then, these features are fed into the model, which is a DRL algorithm. According to the features, three major actions are generated, they are acceleration, no change, and deceleration. One of the three actions is performed by the car in the virtual environment and a reward is sent back to the algorithm. Based on the reward and the features as input, the action to be performed on the environment will be generated by the model. For the hardware requirements, initially, the system is tried on an Intel i5-12400F@2.5 GHz processor having 6 cores and 12 threads. The GPU requirement is $1\times$ NVIDIA RTX 3060 Ti, compute 8.6, having 4864 CUDA cores, 8 GB GDDR6, a RAM of 16 GiB DDR 4 and a storage of 1 TiB SSD.

The software requirements are Windows 11 Home/Pro Edition 22H2 or Mac OS 12 Monterey, Python 3.7.14 scripting language, and TensorFlow 1.15.2 deep learning library. A secure shell is used as a server administrator. Google Collaboratory and Google Drive are used as cloud services and storage services, respectively.

5. Results and Discussion

5.1. Outcomes

In this section, representations of Reward vs Episode and tables showing the ten episode moving average of each algorithm have been shown and discussed.

The primary objective of our investigation is to evaluate the performance nuances of distinct algorithms over a consistent training period lasting approximately 6 h. Owing to disparities in their respective implementations, there exist divergent quantities of episodes completed within this uniform temporal framework. Notably, the DDPG algorithm exhibited the necessitation of the lengthiest episodes as it could only accomplish 100 episodes during the designated training interval. In sharp contrast, the training regimen of the DQN spanned just below 7 h, yet succeeded in completing a substantial 2000 episodes. The discrepancy in episode count highlights our deliberate attempt to emphasize the importance of training duration. As a result, our comparative analysis of the diverse algorithms takes into consideration the varying number of episodes.

5.1.1. Proximal Policy Optimization (Ppo)

Our implementation of PPO was trained for approximately 6 h on our machine, tallying 1100 episodes of training. Figures 8 and 9 represent the reward plots generated over the 1100-episode training period. Table 1 details the ten episode moving average for this algorithm.

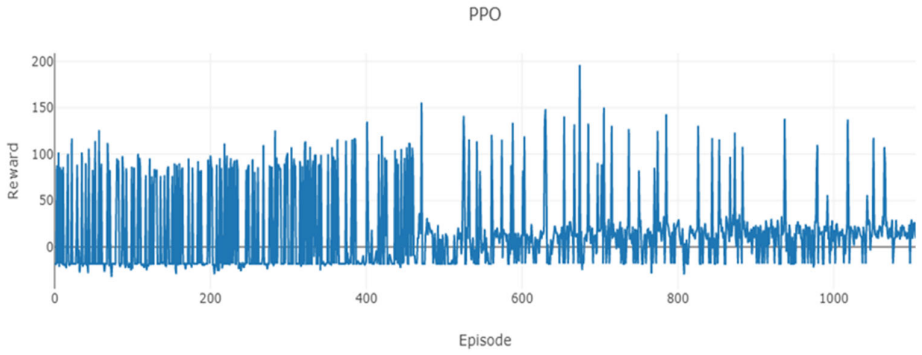

Figure 8. Representation of Reward vs. Episode of PPO algorithm.

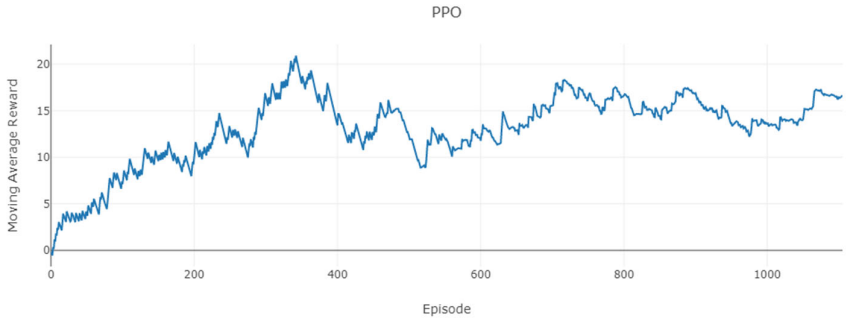

Figure 9. Representation of ten episode Moving Average Reward vs. Episode of PPO algorithm.

Table 1. Tabular representation of ten episode moving average of PPO.

PPO	
Episode	**10-Episode Moving Average Reward**
100	7.11
200	10.13
300	16.2
400	14.64
500	12.22
600	11.77
700	15.53
800	16.65
900	16.88
1000	13.58

5.1.2. Deep Deterministic Policy Gradient (DDPG)

Like PPO, the DDPG was also trained for approximately 6 h, but its manner of implementation resulted in very slow training episodes, tallying only 100 episodes. Figures 10 and 11 represent the reward plots generated over the 100 episode training period. Table 2 represents the ten episode moving average of the DDPG.

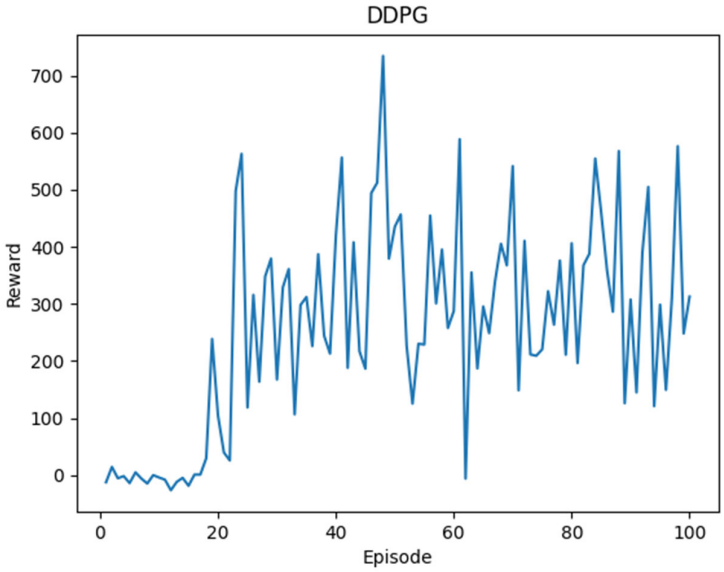

Figure 10. Representation of Reward vs. Episode of DDPG algorithm.

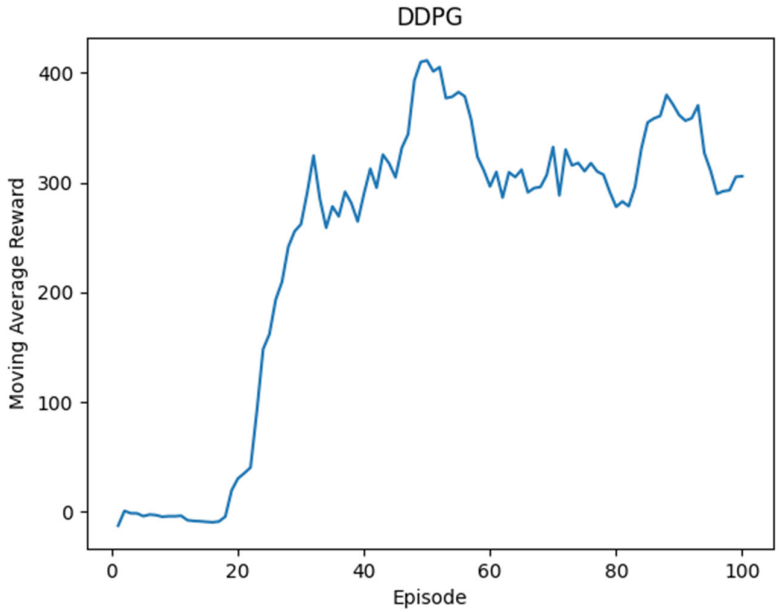

Figure 11. Representation of ten episode Moving Average Reward vs. Episode of DDPG algorithm.

Table 2. Tabular representation of ten episode moving average of DDPG.

DDPG	
Episode	**10-Episode Moving Average Reward**
20	30.47
40	289.82
60	309.63
80	277.94
100	305.71

5.1.3. Deep Q-Learning Network (DQN)

Two variants of DQNs are implemented. The second implementation involves the elimination of a decaying epsilon (ε) hyperparameter to investigate the effect, especially in relatively shorter training periods (7 h). Table 3 compares the ten episode moving average of the DQN with and without ε-decay.

Table 3. Tabular representation of ten episode moving average of DQN.

Episode	DQN 10-Episode Moving Average Reward	
	With ε-Decay	No ε-Decay
200	2.36	−1.89
400	8.32	0.88
600	5.17	206.61
800	13.12	307.39
1000	30.27	416.60
1200	48.97	336.11
1400	82.13	469.29
1600	317.98	541.61
1800	493.05	495.01
2000	415.80	613.44

The first implementation of the DQN was trained for 7 h, tallying 2000 episodes with ε-decay. The reward plots are presented in Figures 12 and 13.

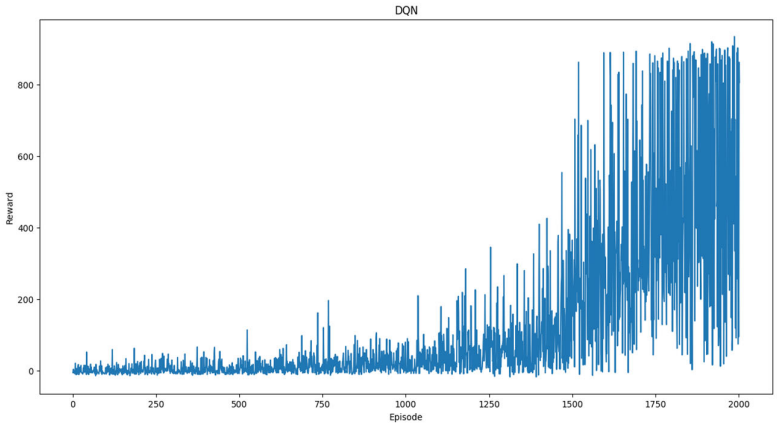

Figure 12. Representation of Reward vs. Episode of DQN algorithm.

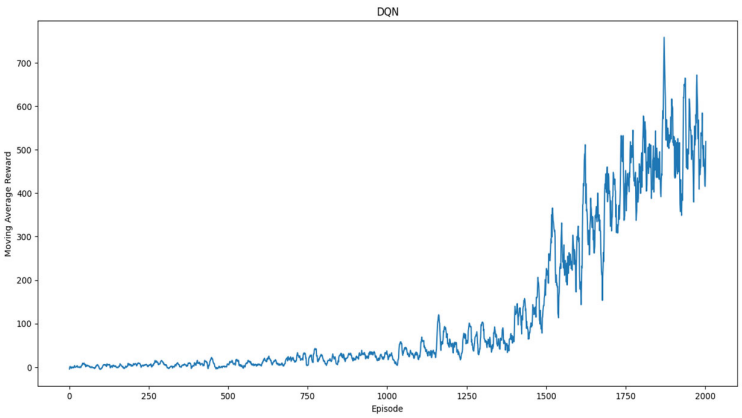

Figure 13. Representation of ten episode Moving Average Reward vs. Episode of DQN algorithm.

To further investigate DQNs and the effect of hyperparameters such as ε, the DQN is implemented a second time, identical to the first implementation, barring the ε-decay hyperparameter. Figures 14 and 15 represent the reward plots generated.

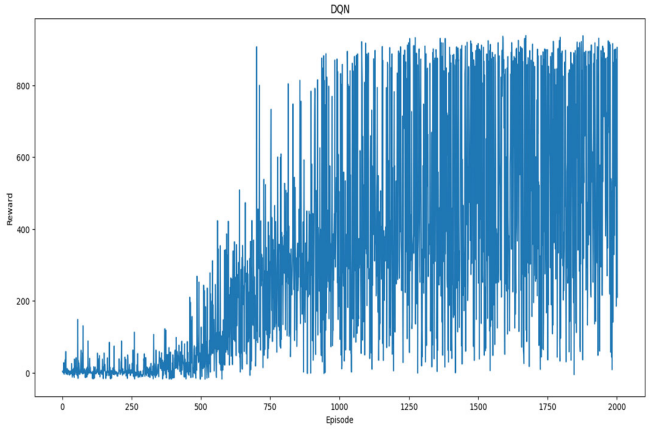

Figure 14. Representation of Reward vs. Episode of DQN (no ε-decay) algorithm.

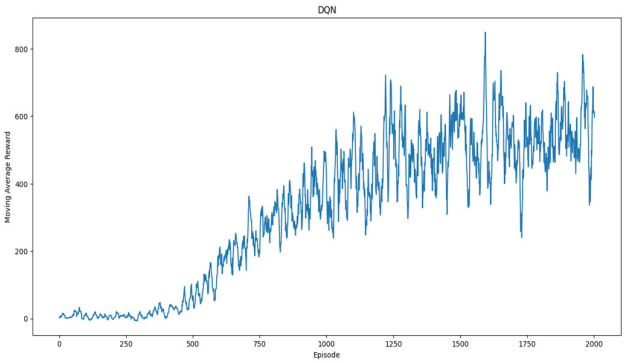

Figure 15. Representation of ten episode Moving Average Reward vs. Episode of DQN (no ε-decay) algorithm.

5.2. Discussion

PPO produced underwhelming results over the 6 h training period, with the best-moving average of approximately 21 around the 340-episode mark. The Moving Average Reward grows slowly but consistently up to that episode; however, following that, there is a sudden drop that fails to recover over the next 800 episodes. The moving average stays between 11 and 18 from the 600th episode onwards. Although Table 1's moving average from 600 to 1000 shows a standard deviation of only 1.9, Figure 8 demonstrates the vast disparity in reward generated. Post the first 400 episodes, the reward is generally below 50 and negative, with the occasional outlier achieving a reward above 100.

The DDPG took the longest time to train per episode; however, it shows promising results. Despite the first 20 or so episodes generating negative rewards, a sudden escalation in generated rewards pushed the moving average to over 300 in the span of 20 episodes between the 20th and 40th episodes. The upward trend continues achieving a moving average of over 400 around the 50th episode, but drops and nearly recovers in the following episodes. Figure 10 shows that, although the reward tends to jump, it does maintain an upward trend and negative rewards are sparse after the first 20 episodes. The plots' trends suggest longer training periods would improve the generated rewards.

The training period per episode of the DQN was the shortest, with it able to attain 2000 episodes in 7 h of training. Although the Moving Average Reward does not improve drastically in the first 1200 episodes with negative rewards not uncommon, the results are promising, especially after the first 1200 episodes as the Moving Average Reward begins to consistently increase, as shown in Figure 13. Although there does appear to be a slight drop-off towards the final episodes of training, it can be assumed that, with further training, the drop-off will be rectified.

The training period per period of the DQN with no ε-decay is like the vanilla DQN achieving 2000 training episodes over 7 h. Figures 14 and 15 show the model produced strong results; despite its poor performance in the first 500 episodes, it was able to show steady and consistent growth for the next 1000 episodes. The remaining 500 episodes from 1500 to 2000, however, do not show much growth overall despite distinct variations. Moreover, the plots display randomness in the rewards generated concerning the deviations with each episode.

5.3. Observations

The DQN proves to be the best model achieving the best Moving Average Reward across a similar time frame to PPO and the DDPG of 6–7 h. PPO is the worst-performing model as it not only scores low rewards, but the plots do not suggest an upward trend in the improvement of the model performance. The DDPG is an improvement over PPO. Unfortunately, compared to the DQN, in the same time frame, it trains on a fraction of the number of episodes and fails to reach the height of both implementations of the DQN. Moreover, it was found that the reproducibility of results in the DDPG is poor, with multiple training instances showing vastly varying results, which is in line with the claims of incoherent result reproduction reported [34].

The implementations of the DQN show varying performance. While it was expected for the implementation with no ε-decay to perform worse, it can be observed that after 2000 episodes, there is a noticeable difference in the ten Moving Average Reward as the vanilla DQN scores a lower moving average of 415.8 at the 2000th episode against 613.44 scored by the DQN with no ε-decay.

Moreover, the DQN with ε-decay takes more episodes to begin, showing strong, consistent growth in rewards compared to the DQN with no ε-decay. The DQN with ε-decay also maintains an upward trend, while the DQN with no ε-decay begins to plateau out over the whole after the 1200th episode. It can be concluded that although the DQN with no ε-decay begins to score higher rewards quicker than the DQN with it, over a longer training period, it is not as stable. Moreover, the trends of our plots suggest that the DQN with ε-decay shows trends for better growth in rewards against the DQN without ε-decay, which, by the 2000th episode, begins slowing down. Based on our evaluation, the authors found that for shorter timeframes, the DQN showed the best performance, followed by the DDPG, while PPO had the lowest performance.

6. Conclusions

According to the findings in this paper, in an environment limited by computational resources and where time is a limiting factor, a DQN produces the best results amongst selected deep reinforcement algorithms. In the DQN without ε-decay, it was found that in ultra-short training time frames, it performs better. But, with the proper training of a DQN with ε-decay, it produces results that are not erratic and produce upward trends, suggesting potential for significantly improved performance over longer training periods. Thus, it can be concluded that it is the better model out of the two. The PPO algorithm produced the worst results, cementing the need for further investigation with significantly longer training periods to evaluate it fairly. The DDPG algorithm had a performance that was better than PPO but still not as good as the DQN. However, it produced erratic results that limit its reproducibility. In the future, the authors would further investigate how these selected models perform over longer training periods and investigate the impact of other hyperparameters of the DQN when trained in shorter time frames. Thus, it can be safely concluded that for our particular problem statement, the Deep Q-learning algorithm is the most suited and DRL is a step in the right direction. With more research on the hyperparameters used and fine-tuning them, authors can successfully implement our model race car simulation and integrate it into car simulation games in the future.

Author Contributions: Conceptualization, A.T., N.C., B.S., D.V. (Divya Venkatesh), G.D. and D.V. (Deepali Vora); methodology, D.V. (Deepali Vora), A.T., N.C., B.S., D.V. (Divya Venkatesh) and G.D.; validation, D.V. (Deepali Vora), S.P., S.A. and K.K.; analysis, S.P., K.K. and S.A.; writing—original draft preparation, N.C., B.S., A.T., D.V. (Divya Venkatesh) and G.D.; writing—review and editing, N.C., B.S., A.T., D.V. (Divya Venkatesh), G.D. and D.V. (Deepali Vora); supervision, D.V. (Deepali Vora) and S.P.; project administration, D.V. (Deepali Vora), S.P., K.K. and S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Researchers Supporting Project Number (RSPD2023R890), King Saud University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors extend their appreciation to King Saud University for funding this research through the Researchers Supporting Project Number (RSPD2023R890), King Saud University, Riyadh, Saudi Arabia.

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

References

1. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement Learning: A Survey. *arXiv* **1996**. [CrossRef]
2. Gosavi, A. Reinforcement Learning: A Tutorial Survey and Recent Advances. *Inf. J. Comput.* **2009**, *21*, 178–192. [CrossRef]
3. Balhara, S.; Gupta, N.; Alkhayyat, A.; Bharti, I.; Malik, R.Q.; Mahmood, S.N.; Abedi, F. A Survey on Deep Reinforcement Learning Architectures, Applications and Emerging Trends. *IET Commun.* **2022**, 1–16. [CrossRef]
4. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym 2016. *arXiv* **2016**, arXiv:1606.01540.
5. Risi, S.; Stanley, K.O. Deep Neuroevolution of Recurrent and Discrete World Models. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019.
6. World Health Organization. Number of Road Traffic Deaths. 2013. Available online: https://www.who.int/data/gho/data/themes/road-safety (accessed on 1 September 2022).
7. Elallid, B.B.; Benamar, N.; Hafid, A.S.; Rachidi, T.; Mrani, N. A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 7366–7390. [CrossRef]
8. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2020**, arXiv:1509.02971.
9. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
10. Van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30. [CrossRef]
11. Padakandla, S. A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments. *ACM Comput. Surv.* **2022**, *54*, 1–25. [CrossRef]
12. Karunakaran, D. Deep Deterministic Policy Gradient (DDPG)—An off-Policy Reinforcement Learning Algorithm. 2020. Available online: https://medium.com/intro-to-artificial-intelligence/deep-deterministic-policy-gradient-ddpg-an-off-policy-reinforcement-learning-algorithm-38ca8698131b (accessed on 1 September 2022).
13. Zhu, M.; Wang, Y.; Pu, Z.; Hu, J.; Wang, X.; Ke, R. Safe, Efficient, and Comfortable Velocity Control Based on Reinforcement Learning for Autonomous Driving. *Transp. Res. Part C Emerg. Technol.* **2020**, *117*, 102662. [CrossRef]
14. Chen, Y.; Dong, C.; Palanisamy, P.; Mudalige, P.; Muelling, K.; Dolan, J.M. Attention-Based Hierarchical Deep Reinforcement Learning for Lane Change Behaviors in Autonomous Driving. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 3697–3703.
15. Cai, Y.; Yang, S.; Wang, H.; Teng, C.; Chen, L. A Decision Control Method for Autonomous Driving Based on Multi-Task Reinforcement Learning. *IEEE Access* **2021**, *9*, 154553–154562. [CrossRef]
16. Wolf, P.; Hubschneider, C.; Weber, M.; Bauer, A.; Hartl, J.; Durr, F.; Zollner, J.M. Learning How to Drive in a Real World Simulation with Deep Q-Networks. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 244–250.
17. Ghimire, M.; Choudhury, M.R.; Lagudu, G.S.S.H. Lane Change Decision-Making through Deep Reinforcement Learning. *arXiv* **2021**, arXiv:2112.14705.
18. Arvind, C.S.; Senthilnath, J. Autonomous Vehicle for Obstacle Detection and Avoidance Using Reinforcement Learning. In *Soft Computing for Problem Solving*; Das, K.N., Bansal, J.C., Deep, K., Nagar, A.K., Pathipooranam, P., Naidu, R.C., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2020; Volume 1048, pp. 55–66. ISBN 9789811500343.

19. Gopika Gopinath, T.G.; Anitha Kumari, S. Deep Reinforcement Learning Framework for Navigation in Autonomous Driving. *Int. J. Eng. Res. Technol.* **2019**, *8*, 1461–1464.
20. Yildirim, M.; Mozaffari, S.; McCutcheon, L.; Dianati, M.; Tamaddoni-Nezhad, A.; Fallah, S. Prediction Based Decision Making for Autonomous Highway Driving. In Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, China, 8–12 October 2022; pp. 138–145.
21. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. Deep Reinforcement Learning Framework for Autonomous Driving. *arXiv* **2017**, arXiv:1704.02532. [CrossRef]
22. Osinski, B.; Jakubowski, A.; Ziecina, P.; Milos, P.; Galias, C.; Homoceanu, S.; Michalewski, H. Simulation-Based Reinforcement Learning for Real-World Autonomous Driving. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 6411–6418.
23. Candela, E.; Parada, L.; Marques, L.; Georgescu, T.-A.; Demiris, Y.; Angeloudis, P. Transferring Multi-Agent Reinforcement Learning Policies for Autonomous Driving Using Sim-to-Real. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022.
24. Song, S.; Saunders, K.; Yue, Y.; Liu, J. Smooth Trajectory Collision Avoidance through Deep Reinforcement Learning. In Proceedings of the 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 12–14 December 2022.
25. Xiong, X.; Wang, J.; Zhang, F.; Li, K. Combining Deep Reinforcement Learning and Safety Based Control for Autonomous Driving. *arXiv* **2016**, arXiv:1612.00147.
26. Dagdanov, R.; Durmus, H.; Ure, N.K. Self-Improving Safety Performance of Reinforcement Learning Based Driving with Black-Box Verification Algorithms. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 5631–5637.
27. Dagdanov, R.; Eksen, F.; Durmus, H.; Yurdakul, F.; Ure, N.K. DeFIX: Detecting and Fixing Failure Scenarios with Reinforcement Learning in Imitation Learning Based Autonomous Driving. *arXiv* **2022**, arXiv:2210.16567. [CrossRef]
28. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
29. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Riedmiller, M.; Wierstra, D. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; Volume 32, pp. 387–395.
30. Williams, R.J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
31. Wang, P.; Chan, C.-Y.; de La Fortelle, A. A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018.
32. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *arXiv* **2016**, arXiv:1610.03295.
33. Pan, X.; You, Y.; Wang, Z.; Lu, C. Virtual to Real Reinforcement Learning for Autonomous Driving. *arXiv* **2017**, arXiv:1704.03952.
34. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep Reinforcement Learning That Matters. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32. [CrossRef]
35. Asperti, A.; Del Brutto, M. MicroRacer: A Didactic Environment for Deep Reinforcement Learning. In Proceedings of the In International Conference on Machine Learning, Optimization, and Data Science, Certosa di Pontignano, Italy, 19–22 September 2022.
36. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
37. Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; de Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016.
38. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 1889–1897.
39. Zhu, Z.; Zhao, H. A Survey of Deep RL and IL for Autonomous Driving Policy Learning. *IEEE Trans. Intell. Transport. Syst.* **2022**, *23*, 14043–14065. [CrossRef]
40. Liu, G.; Zhang, C.; Zhao, L.; Qin, T.; Zhu, J.; Li, J.; Yu, N.; Liu, T.-Y. Return-Based Contrastive Representation Learning for Reinforcement Learning. *arXiv* **2021**, arXiv:2102.10960.
41. Srinivas, A.; Laskin, M.; Abbeel, P. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020.
42. Joshi, A.; Kurchania, H.; Patwa, A.; Sagar, T.S.K.; Kshirsagar, U.; Vora, D.R. Video Object Segmentation with Self-Supervised Framework for an Autonomous Vehicle. In Proceedings of the 6th Smart Cities Symposium (SCS 2022), Hybrid Conference, Bahrain, 6–8 December 2022; pp. 20–25.