IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# EXPLORATION OF SIMULATED AUTONOMOUS VEHICULAR NAVIGATION VIA REINFORCEMENT LEARNING

*AN ANALYTICAL EXAMINATION OF THE UNITY ML-AGENTS FRAMEWORK*

**[1]Aditi Patil, [2]Parag Patel, [3]Dr. Gopal Pardesi**

Department of Information Technology Engineering
Thadomal Shahani Engineering College, Mumbai, India.

***Abstract***: Artificial intelligence developments are assisting researchers in finding solutions to challenging issues, such programming video game automobiles to drive themselves. In this work, computer characters were taught to race on a virtual track using the learning method known as reinforcement learning utilizing specialized software called Unity ML-Agents. We experimented with various training techniques for these computer characters and discovered that Proximal Policy Optimization was the most effective. With relatively little error, our computer-generated figures were able to race around the course using this technique. Then, we added barriers to the track, such as barrels and cones, and trained our computer-generated characters to avoid these barriers using a distinct technique called behavioral cloning. This strategy actually worked.

**Index Terms - reinforcement learning; autonomous driving;**

## I. INTRODUCTION

Intelligent agents can be trained for a variety of purposes using reinforcement learning (RL), which is a powerful technology. This adaptable method is useful in robotics, where it teaches robots to be excellent at activities like manipulating objects and traversing physical surroundings. This strategy holds great promise for enterprises making the transition to Industry 4.0. When it comes to video games, RL gives agents the ability to perform at superhuman levels in chess, go, and poker. By training them to make crucial judgments like lane changes and adherence to traffic restrictions, RL also plays a crucial role in the development of autonomous cars. In the field of healthcare, RL improves treatment strategies, advancing personalized medicine. factories, water treatment facilities, and power grids are examples of industrial systems that benefit from RL's ability to enhance control and efficiency. RL is used in the energy industry to manage renewable energy sources, storage options, and energy efficiency. Additionally, recommendation systems take advantage of RL to offer customers tailored recommendations for things to buy or movies to watch. Finally, RL is crucial in strengthening cybersecurity because it teaches agents how to recognize and efficiently counteract online threats. Recommendation systems, robotics, gaming, autonomous vehicles, healthcare, industrial control, energy management, and cybersecurity are just a few of the fields where RL is used as a flexible and essential technology.

The creation of autonomous racing agents, which are capable of navigating race circuits and making decisions in real-time to avoid opponents and obstacles, is a promising application of Reinforcement Learning (RL). When it comes to things like self-driving cars and finding the best route, it can be difficult and expensive to collect real-world data. So these computer programs are often trained and tested in a virtual world that is like a video game. This way, they can get a lot of practice and learn without spending a lot of money on real-world experiments. Additionally, simulation settings give researchers more control over the training environment, enabling them to experiment with different track configurations and weather patterns.

The development of self-driving race cars is regarded as a key issue in the realm of mobile robotics since it necessitates mastery in a variety of skills, including vision, speed control, racecourse planning, and decision-making. It's basically a full test of all of these abilities. High-speed navigation across dynamic, unstructured settings while dodging objects, navigating tough terrain, and racing against other agents are all aspects of autonomous racing. To make mobile robots work properly, we need extremely intelligent computer programs capable of seeing things, making judgments, and controlling the robot's actions. These programs must also be able to process data quickly. One critical aspect of developing mobile robots is testing them in hazardous environments with numerous obstacles or items the robot is unfamiliar with, and doing so safely. Additionally, advancements in autonomous racing may result in mobile robots that are more capable and useful in a variety of contexts, such as delivery and search and rescue.

Furthermore, because the difficulties in self-driving are similar to those in other autonomous robotic system, breakthroughs in self-driving racing car may spur innovations in the larger area of robotics. It also acts as a testing ground for many technologies that are useful for mobile robotics, such as sensors, cameras, and lidar. This study investigates how to train ML agents for racing track navigation in a realistic simulation environment. Finding the most efficient method for training self-driving racing bots is the goal.

The contribution of the research is the comparative investigation of different RL algorithms for training ML agents in a racing track environment, including Vanilla policy gradient (VPG), Proximal Policy Optimization (PPO) and POCA (Parallel Online Continuous Arcing). In order to speed up model learning and improve agent performance in the racing task, it also suggests using cloning as a training condition. The research provides insights to enhance the training of intelligent agents in simulated settings by shedding light on the advantages and disadvantages of various RL techniques. The findings add to the expanding corpus of work on RL algorithms for educating intelligent agents in virtual environments.

## II. LITERATURE SURVEY

The application of multi-agent reinforcement learning algorithms is a significant development in the realm of reinforcement learning for simulated settings. These methods will simplify the process to teach a group of agents to engage with and acquire knowledge from each other in a common environment. This advancement holds the potential to accelerate both the training of agents for collaborative task execution and the development of more convoluted and plausible simulations. Additionally, interest in the field has experienced a notable increase about the utilization of reinforcement learning algorithms in the development of self-governing racing agents in virtual track environments. Incorporating actor-critic algorithms is a common component of these techniques. These algorithms acquire the ability to anticipate the usefulness of various actions within a given state and leverage this knowledge to influence their decision-making. In a different study, the authors expand the Generative Adversarial Imitation Learning (GAIL) framework, which normally integrates inputs provided by humans, to address limitations during the training of a large number of agents. They present a GAIL variation that shares parameters and exhibits improved stability in multi-agent interactions. In yet another article, the author describes a reinforcement learning-based method for coordinating cooperative activity among several agents in a traffic simulation scenario. The author trains these agents using deep reinforcement learning, honing and optimizing their actions in response to those of other agents in the environment. The performance of both individual agents and the system as a whole is taken into account through a novel reward function that is proposed. The author also offers a system that allows agents to converse and share information about their behaviours in particular settings while navigating by using temporal data and historical records. Convolutional Neural Networks (CNN) are used by the authors of a different study to extract relevant information from the route. They then use a Long Short-Term Memory (LSTM) network to make decisions based on past data that details various responses to different extracted attributes. This method is evaluated in an open racing automobile simulator and shows a noteworthy level of accuracy in its replication of human decision-making.

For the training and improvement of autonomous driving models, certain techniques blend real-world and virtual settings. Initially, simulated agents can train both deep and traditional Reinforcement Learning-based models and then adjusted in actual environments to match desired performance. In their paper, the scientists suggest a thorough method for self-driving systems that makes use of neural networks to roughly determine the right motor commands from sensory inputs. The authors gather recovery information based on the departure from a chosen track during road tests carried out in a simulator in order to handle the problem

of bringing a vehicle back to its allocated lane after deviating off-course. Three steps make up their method: data collection through a path-following module over the course of 100 iterations of driving; training a neural steering module using this dataset to produce driving behaviour; and the neural driving module will be retrained using the data from the path-tracking module over the course of another 100 laps of driving. Among datasets with no data restoration, random data recovery, and the recommended data recovery method, the average distance from the next waypoint link and the average mileage each lap are compared. The outcomes illustrate the effectiveness of the model following the suggested methodology, demonstrating a higher emphasis on the road than on unrelated things across both courses that have undergone training and those that have not be trained as well as under diverse weather situations.

The landscape of RL algorithms is constantly changing in simulated environments, pushing the limits of efficacy and efficiency in training smart agents to accomplish a diverse range of duties. Notably, deep RL, which combines RL approaches with deep neural networks, has become a powerful tool for handling challenging, issues with complex sensory data. As an example, autonomous vehicles can be trained in simulated surroundings like the Unity engine for navigating roadways and traffic. Multi-agent Reinforcement Learning (RL) techniques have also made it easier for numerous agents to learn cooperatively in a shared environment, enabling the advancement of more complicated, plausible simulations and the cooperative completion of tasks. Actor-critic algorithms are used in the context of autonomous racing to direct navigation, and multi-agent collaboration is established for simulated traffic problems. In order to extract road features and past data, Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) are being used. This aids in the decision-making process for autonomous driving. It's significant that some methods integrate simulated and real-world environments to improve model training and fine-tuning in autonomous driving scenarios.

## III. METHODOLOGY

### 3.1 Autonomous Cart Racing employing Reinforcement Learning

A learner comprehends how to make decisions through engaging with their surroundings in a learning framework referred to as reinforcement learning (RL). Enhancing the anticipated long-term collective reward is the agent's objective. The task of autonomous cart racing pits various actors, represented by autonomous carts, against one another, which involves navigating through a track as quickly as feasible. For this purpose, RL can be characterized mathematically as follows:

Consider a group of autonomous agents designated by "A," which consists of $a_1$, $a_2$,..., $a_n$, where 'n' represents the total number of agents. Each agent, namely ai, interacts with the racetrack environment via autonomous vehicles over a discrete time horizon denoted by t = 1, 2,..., T. Agents are entrusted with making decisions throughout their interactions, choosing from a repertoire of actions, including movements such as steering, acceleration, and deceleration, at each discrete time point, 't'. Following that, the environment goes through a state transition, moving to a new state, $s_{t+1}$, and rewarding each agent with a scalar reward, $r_{i,t}$. Each agent's primary goal is to develop an operational strategy, indicated here as $\pi_i(a_{i,t},s_t)$, that embodies the distribution of probabilities guiding their action options depending on the current state. In effect, this strategy acts as their blueprint for optimizing the cumulative incentives gained throughout the whole time horizon.

$$J(\pi_i) = E\pi_i\left[\sum t = 0^{T-1}\gamma^t r_{i,t}\right] \quad \text{Eqn (1)}$$

where [0, 1] is a discount factor that controls the importance of future benefits. $r_{i,t}$ also refers to the award for finishing the race fast, avoiding crashes, and avoiding fines. This may be described further by examining the state value function $V_{\pi_i}(s_t)$, which indicates the estimated time to reach the target line beginning from state $s_t$ and following policy $\pi_i$:

$$V_{\pi_i}(s_t) = E\pi_i\left[\sum k = 0^{T-t-1}\gamma^k r_{i,t+k}|s_t\right] \quad \text{Eqn (2)}$$

The below equation indicates the estimated time to reach the target line beginning with state st, executing action $a_{i,t}$, and according to policy $\pi_i$:

$$Q_{\pi_i}(s_t, a_{i,t}) = E\pi_i\left[\sum k = 0^{T-t-1}\gamma^k r_{i,t+k}|s_t a_{i,t}\right] \quad \text{Eqn (3)}$$

### 3.2 Algorithms

3.2.1 MA-PPO Algorithm

MAPPO is a reinforcement learning system that trains numerous agents to make cooperative or competitive decisions in a shared environment. It is an extension of the Proximal Policy Optimization (PPO) method designed for scenarios with numerous agents interacting with one another, such as in self-driving automobiles.

MAPPO can be used to train many autonomous vehicles that must negotiate a complicated metropolitan environment in the context of self-driving automobiles. These cars must make judgments that take into account not only their own sensor data and objectives, but also the actions and behaviours of other vehicles on the road. Because of the cooperative or competitive interaction amongst autonomous agents, MAPPO is a good choice for training self-driving automobiles.

MAPPO works by optimizing each agent's policies to maximize their individual or collective benefits while ensuring that the policies do not diverge too far from earlier versions. This stability is critical in training multi-agent systems to avoid undesired behaviours. MAPPO does this by imposing a "proximal" restriction on policy updates, which limits the amount by which policies can change at each training phase.

This implies that, in the context of self-driving cars, each vehicle, as an agent, learns to adopt behaviours that are not only safe and efficient on their own, but also take into account the behaviour of other vehicles in the surroundings. For example, a self-driving automobile employing MAPPO may learn to blend into traffic smoothly, avoid crashes, and collaborate.

Overall, Multi-Agent Proximal Policy Optimization is a useful technique for teaching self-driving automobiles how to operate successfully and safely in complicated traffic settings by allowing them to learn both individual and cooperative behaviours using a reinforcement learning framework. This enables them to navigate real-world situations and make driving judgments that take other cars' presence and behaviour into account, ultimately leading to safer and more efficient autonomous transportation systems.

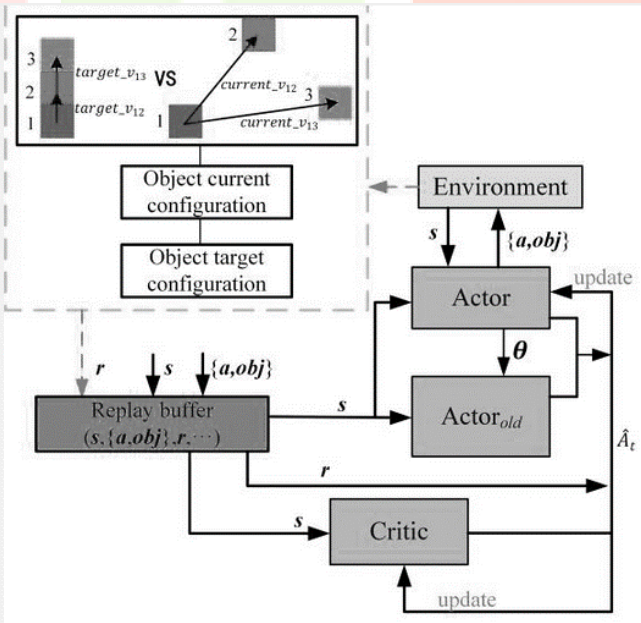



Fig 1. Flowchart of MA-PPO Algorithm

3.2.2 POCA Algorithm

POCA (Parallel Online Continuous Arcing) is a sophisticated prediction approach that differs from previous methods such as Adaboost. Instead of guessing one item at a time, it's like having a whole team of guessers who are constantly making guesses and learning and modifying on the fly. This allows individuals to adjust fast to changing scenarios and eliminates the need for them to remember what they've previously assumed, making it ideal for continual learning and swiftly responding to new knowledge. In Figure 2, a simplified view of the way POCA algorithm works can be seen.

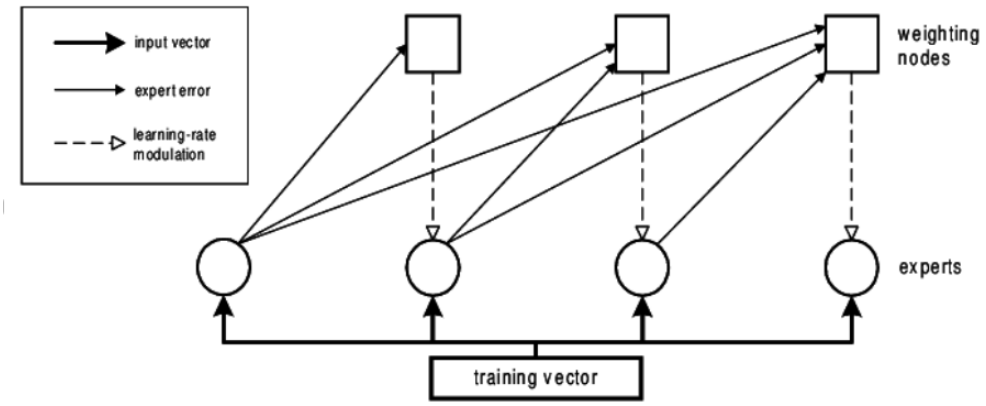



Fig 2. Flowchart of POCA Algorithm

3.2.3 VPG Algorithm

A fundamental reinforcement learning method called Vanilla Policy Gradient (VPG), commonly referred to as the Reinforce algorithm, tries to train an agent's policy by maximizing the expected cumulative rewards it receives in an environment. VPG directly focuses on optimizing the policy itself, in contrast to standard value-based methods that assess the worth of states or state-action combinations. To do this, it employs stochastic rules, which are frequently represented by neural networks, and modifies their parameters using gradient ascent to raise the likelihood that a given activity would result in bigger rewards.

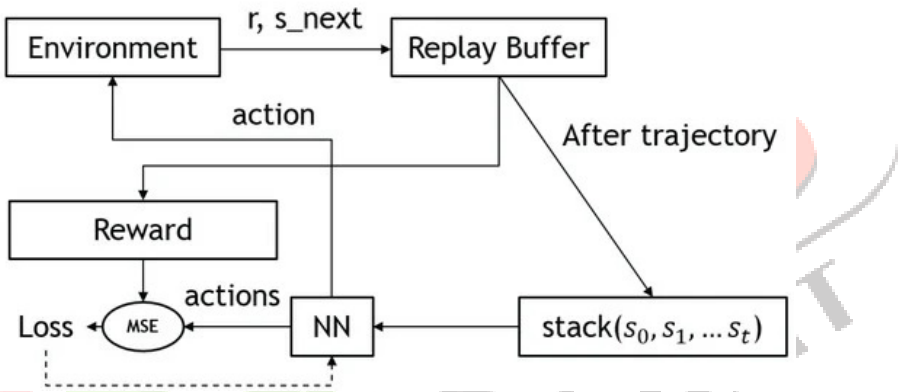



Fig 3. Flowchart of VPG Algorithm.

## IV. EXPERIMENTAL EVALUATION AND RESULTS

### 4.1 Settings

In the final stage, the mean reward for all vehicles is compared. It is not recommended to compare the rewards of experiments with and without obstacles directly since obstacles slow down the learner. So we check all the algorithm in surrounding of obstacles and with surrounding without the obstacles, to get the optimised algorithm. Below is algorithms we checked in no obstacle surroundings and with obstacle surroundings.

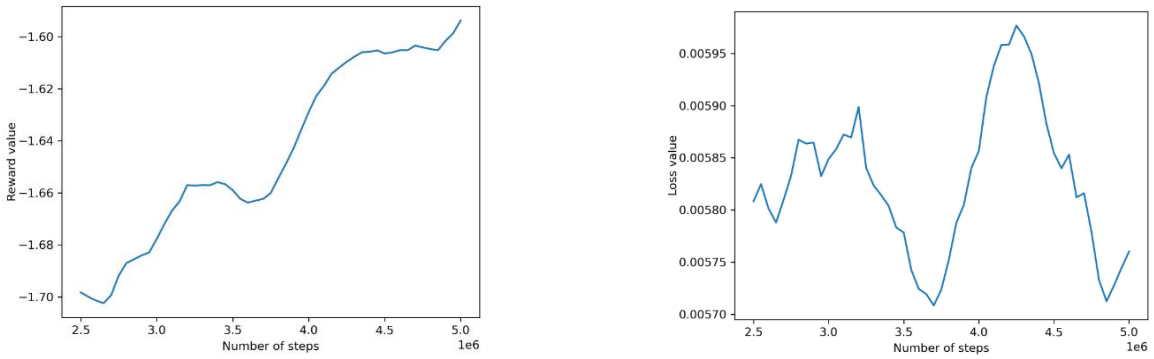
- No Obstacle.
  - VPG algorithm.
  - PPO algorithm.
  - POCA algorithm.
  - Including Recurrent neural network to the optimised model.
- Obstacle.
  - PPO algorithm.
  - Behavioural cloning with the PPO algorithm.

## 4.2 Comparative Analysis

4.2.1 Environment without Obstacles

- VPG model

  First, experiments were conducted using the default VPG model. As a result of the agents' difficulty navigating the track and failure to provide any recent rewards or value loss, the results were not satisfactory. The model produced a value of loss function of 0.005 and a mean cumulative return function of 1.583. The model's plots are displayed in Figure 4**.**

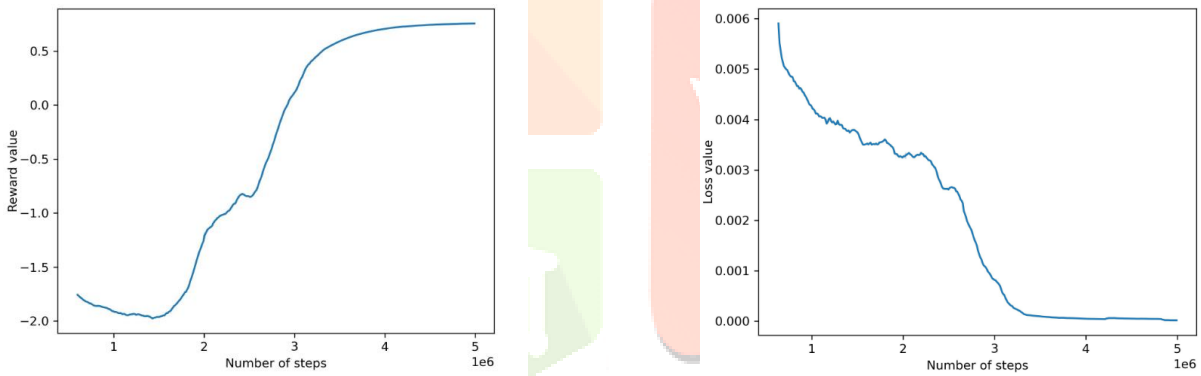



(i) (ii)

Fig 4. VPG model results: (i) Reward function, (ii) Loss function

- PPO model

  The PPO model was. Showed the results with a reward function value of 0.760 and a loss function value of 0.0012. Figure 5 displays the graphs generated by the model.

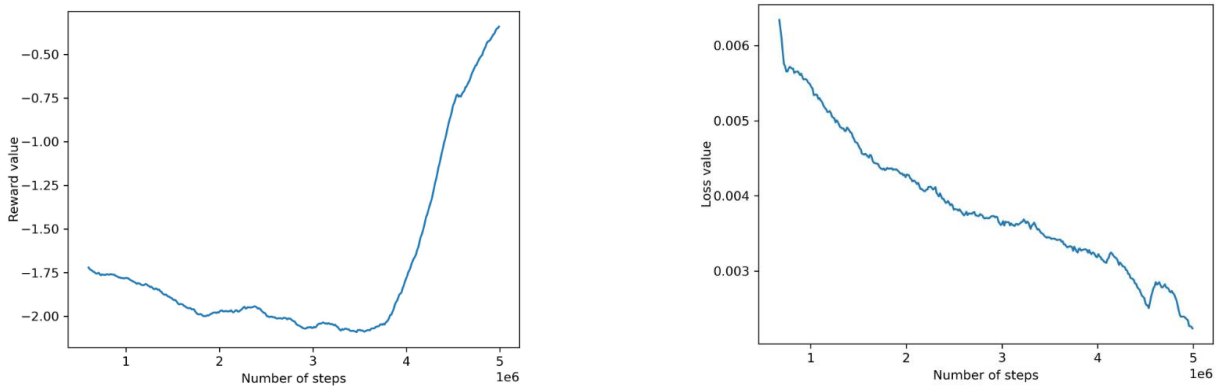



(i) (ii)

Fig 5. PPO algorithm results: (i) Reward function, (ii) Loss function

- POCA Model

  The cumulative value of reward function and value of loss function for the POCA model were decent, coming in at 0.38 and 0.003, respectively. In Figure 6, you can view the model charts.

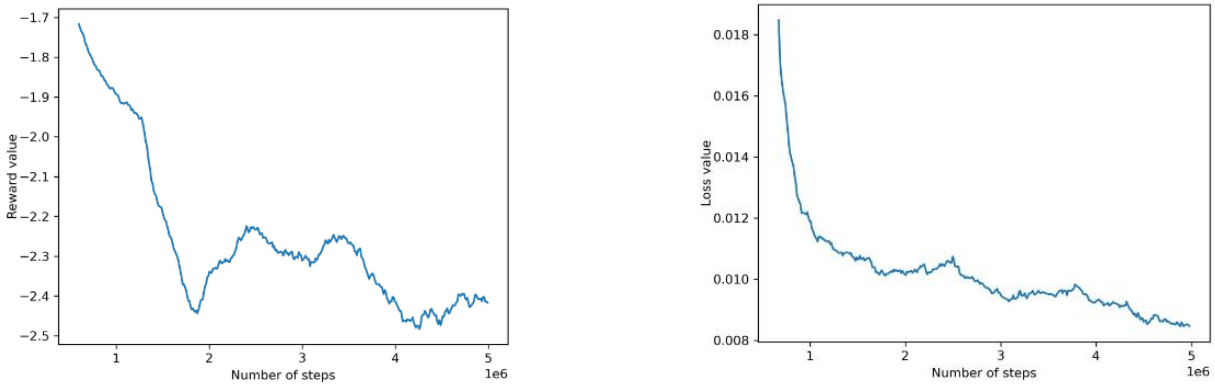



(i) (ii)

Fig 6. POCA model results : (i) Reward function, (ii) Loss function

- Including Recurrent neural network to the optimised model

The cumulative value of reward function and value of loss function for the POCA model were decent, coming in at 2.410 and 0.083, respectively. In Figure 7, you can view the model charts.



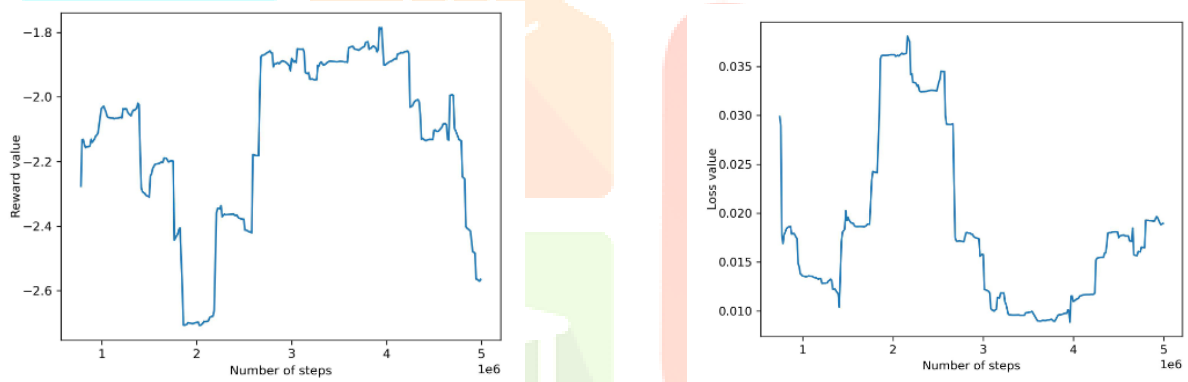

(i) (ii)

Fig 7. Adding RNN to PPO model results : (i) Reward function, (ii) Loss function

4.2.2 Environment with Obstacles

- PPO model

The cumulative value of reward function and value of loss function for the POCA model were decent, coming in at 2.575 and 0.019, respectively. In Figure 8, you can view the model charts.

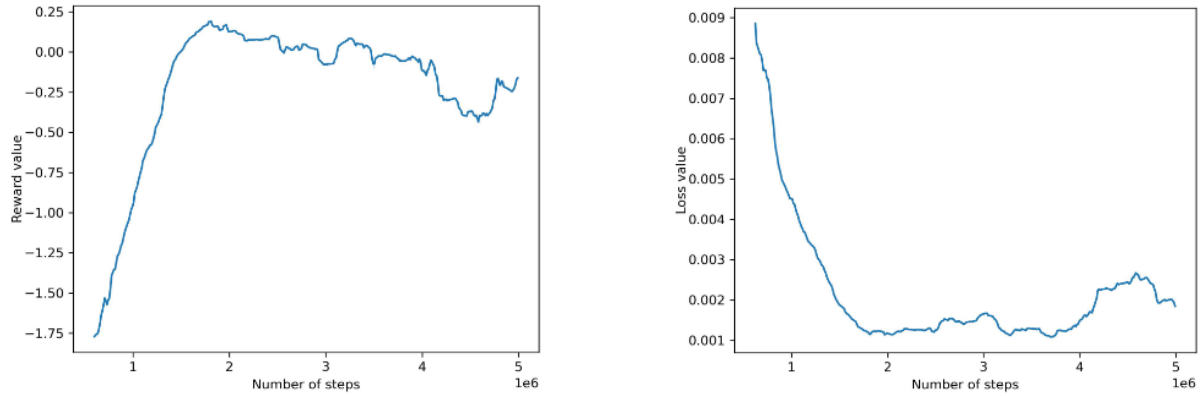



(i) (ii)

Fig 8. PPO model (Obstacle Environment) results: (i) Reward function, (ii) Loss function

- Behavioural cloning with the default PPO algorithm

Through the use of an environment with obstacles added, the default PPO was retrained. Here, a pre-training condition of behavioural cloning was applied, with a strength hyperparameter of 1.0. With a final mean reward function value of 0.0680 and a loss function value of 0.0012, the results were unsatisfactory. In Figure 9, the model's graphs are displayed.




(i) (ii)

Fig 9. PPO model (Obstacle Environment) results with Behavioural Cloning :
(i) Reward function, (ii) Loss function

## V CONCLUSION

This research delves into the application of reinforcement learning (RL) techniques for navigating kart agents as they traverse a simulated racing track within the Unity ML-Agents toolbox. With the objective of instructing kart agents in efficient race track navigation and obstacle avoidance., we have analysed the performance of a number of various RL algorithms and configurations. As a result, we have determined the most efficient method.

Our findings generally have substantial implications for the development and deployment of intelligent entities within racing simulations. Our observations provide light on the limitations and potential of various RL algorithms and can guide the creation of improved training methods for intelligent agents in virtual environments. In our investigation and findings, we include a number of sources.

1. Various models were trained, and the outcomes were documented. The usual normal setting, which employs the PPO algorithm, showed out to be the most efficient model. The model generates a total reward function's value of 0.760 and a loss function's value of 0.0012 for the last interval.
2. The outcomes of adding obstacles and retraining with the best algorithm discovered were unsatisfactory. AI bots failed to discover a strategy that yields respectable rewards. This model's final phase revealed that the reward and loss were 1.718 and 0.0152, respectively. The model was pre-trained using Behavioural cloning to assist it learn the required behaviour. By physically interacting with the system, the authors were able to record the intended behaviour. The model was successful in producing results that were satisfactory in that the agents were able to manage to dodge and avoid obstacles and finish the track using behavioural cloning. These had rewards of 0.0681 and losses of 0.0011, respectively.

## VI REFERENCES

[1] Liu, H.; Kiumarsi, B.; Kartal, Y.; Taha Koru, A.; Modares, H.; Lewis, F.L Reinforcement Learning Applications in Unmanned Vehicle Control: A Comprehensive Overview.
[2] Cai, P.; Wang, H.; Huang, H.; Liu, Y.; Liu, M. Vision-Based Autonomous Car Racing Using Deep Imitative Reinforcement
Learning. IEEE Robot. Autom.
[3] Wei, W.; Gao, F.; Scherer, R.; Damasevicius, R.; Połap, D. Design and implementation of autonomous path planning for intelligent vehicle. J. Internet Technol.
[4] Yusef Savid, Reza Mahmoudi, Rytis Maskeliunas and Robertas Damasevicius: Simulated Autonomous Driving Using Reinforcement Learning: A Comparative Study on Unity's ML-Agents Framework. In the Proceedings of 2023 Multidisciplinary Digital Publishing Institute (MDPI).
[5] Urrea, C.; Garrido, F.; Kern, J. Design and implementation of intelligent agent training systems for virtual vehicles.
[6] Almasi, P.; Moni, R.; Gyires-Toth, B. Robust Reinforcement Learning-based Autonomous Driving Agent for Simulation and Real World. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN).
[7] Onishi, T.; Motoyoshi, T.; Suga, Y.; Mori, H.; Ogata, T. End-to-end Learning Method for Self-Driving Cars with Trajectory Recovery Using a Path-following Function. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN).
[8] Palanisamy, P. Multi-Agent Connected Autonomous Driving using Deep Reinforcement Learning. In Proceedings of the 2020
International Joint Conference on Neural Networks (IJCNN).