https://doi.org/10.33472/AFJBS.6.6.2024.1859-1872

African Journal of Biological Sciences

Journal homepage: http://www.afjbs.com

Research Paper Open Access

Design and Assessment of an Autonomous Parking System Using Reinforcement Learning Agents in Unity3D

Yogendra Narayan Prajapati[1], Dr. Nandita Goyal[2], Dr. Shivani Agarwal[3], Dr. Upasana Pandey[4], Dr.Anu Chaudhary[5]

[1]Assistant Professor, CSE Department, Ajay Kmar Garg Engineering College Ghaziabad, U.P.India
[2]Assistant Professor, IT Department, Ajay Kmar Garg Engineering College Ghaziabad, U.P.India
[3]Associate Professor, IT Department, Ajay Kmar Garg Engineering College Ghaziabad, U.P.India
[4]Professor, Department of Computer science and engineering (Artificial Intelligence) ABES Institute of Technology
[5]Professor, CSE Department, Ajay Kmar Garg Engineering College Ghaziabad, U.P.India

Email: Ynp1581@gmail.com,hodcse@akgec.ac.in, goyalnandita@akgec.ac.in , kasishivani@gmail.com , coe.upasana@gmail.com

Article Info

Volume 6, Issue 6, June 2024

Received: 06 April 2024

Accepted: 11 May 2024

Published: 03 June 2024

doi: 10.33472/AFJBS.6.6.2024.1859-1872

ABSTRACT:

This paper describes how RL agents in the Unity Environment can perform parking. The goal of the study is to propose method that makes use of reinforcement learning techniques offered by the Unity ML- Agents framework within Unity's realistic 3D simulation in order to solve the requirement for autonomous parking solutions. The suggested solution's design, execution, and assessment are highlighted in the paper. In complex situations, the system offers an adaptive and realistic frame-work for autonomous parking. The outcomes of thorough performance testing and comparative analysis highlight the use fullness and promise of the suggested approach in the area of autonomous car parking. The discussion of the results, difficulties faced, and prospects for additional study and advancement in autonomous car parking technology round up the report.

Keywords: Unity ML-Agents, Unity Game Engine, Autonomous Park- ing System, Reinforcement Learning

1. Introduction:

The everyday evolution of Artificial Intelligence and Virtual Simulations is leading the way to new technological advancements. Within this paradigm, the task of autonomous parking requires great precision and adaptability for intricate scenarios. The problem can be addressed by the approaches of Machine Learning (ML) and Reinforcement Learning (RL). This paper focuses on utilising the capabilities of RL provided by the Unity ML-Agents framework, within the Unity3D simulation environment to solve the problem of autonomous vehicle parking.The question arises, what is Reinforcement learning? Let's take the example of Volleyball (Fig1). Initially, the age nts do not have any information on how to play the game. They'll start by taking random actions and through trial and error, they'll learn that:[Zha]

If they hit the ball and it goes over the net to the other side of the court, they score points (positive feedback),

If they let the ball hit the floor on their side of the court, they lose a point (negative feedback).

Doing the things that lead to positive outcomes will teach the agents to hit the ballover the net whenever it's on their side of the court. Technically, Reinforcement learning is a sub domain of machine learning which involves training an 'agent' (here the volleyball player) to learn the correct sequences of actions to take (hitting the ball overhunt)in given state of it environment(the volley ball game)to maximize its reward (scoring points) [Zha][SB18].The RL training process includes 2 key steps/phases: Exploration and Exploitation. The developer's training algorithm will decide when the agent should explore the environment and when to exploit the gainedinformation.Wewilltakeadeeperlookintothisinfurthersections.

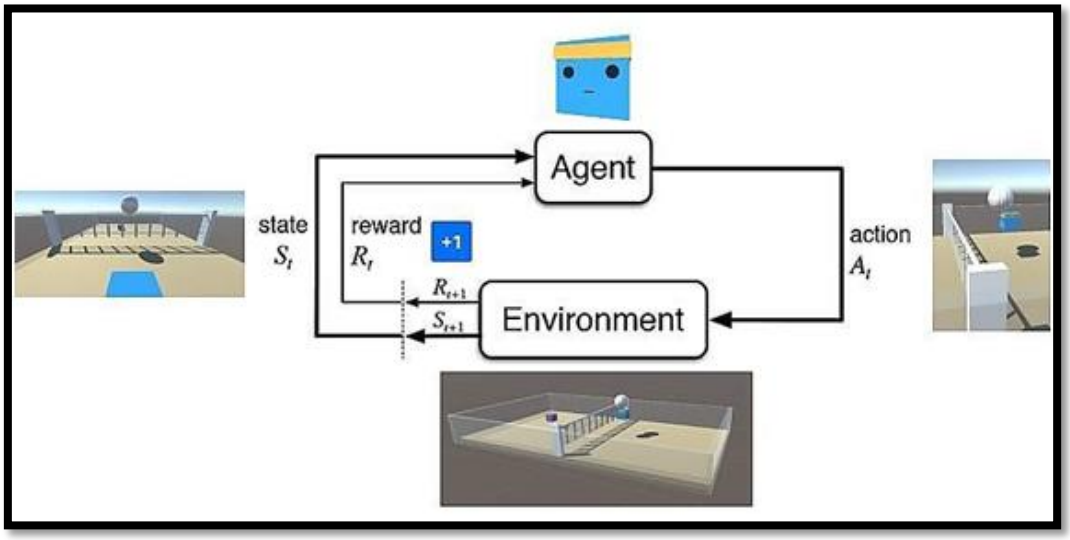


Figure1:Re enforcement Learning[Source]

The robust physics engine and realistic rendering provided by the Unity are ideal for creating simulated environments, they closely mimic the real-world challenges which further rstrengthens the training for our RL agent. Asautonomous vehicle technology progresses, the development of intelligent parking systems becomes crucial. These types of systems can aid in transforming current automobiles into partially autonomous vehicles as they can be provided as third-party modules. This proposal leverages Unity's capabilities to create a simulated environment where a virtual car equipped with an RL agent can learn to navigate

diverse parking scenarios, hence addressing the pressing need for an automatic parking system.

This paper is structured as follows:[JBT+20]

We begin with an introduction of the problem at hand, its significance, and the scope of this paper. Then, we describe the current and previous works on the problem and propose a series of solutions, including our primary RL agent using Unity ML Agent in the Unity3D Environment,

We then describe the Unity engine and Unity ML-Agents Toolkit, a general platform and discuss its ability to enable research and how we can achieve the proposed solution using them,

Wenextoutlinethearchitecture,functionalityandtoolsprovidedbytheUnity ML-Agents Toolkit which enables the deployment of RL Agent within Unity environments on example Parking Scenarios,Then, we assess the outcome and conclude by proposing future avenues of our findings.

The Problem:

With the increase in the popularity of self-driving cars, the world's roads are projected to be dominated by such cars in the next decade. But it also introduces Various challenges, including but not limited to lane detection, lane following, signal detection, and obstacle detection and avoidance, make autonomous driving a complex task. For this paper, we will focus on the task of parking. Car parking is a complex task that requires the driver to handle:

Spatial awareness

Trajectory planning

Real-time decision-making, and more (Fig. 2)

These challenges combined are so significant that a traditional algorithm cannot overcome them. This paper identifies this problem and aims to train an RL agent to navigate and park a car autonomously, recognizing the dynamic and complex nature of the parking situation. The main reason to opt for the RL technique over other sorts of algorithms, and the Unity environment, is due to the facilities provided. We will take a closer look at the Unity Engine and the Unity ML-Agents framework (including RL tools provided) in further sections.

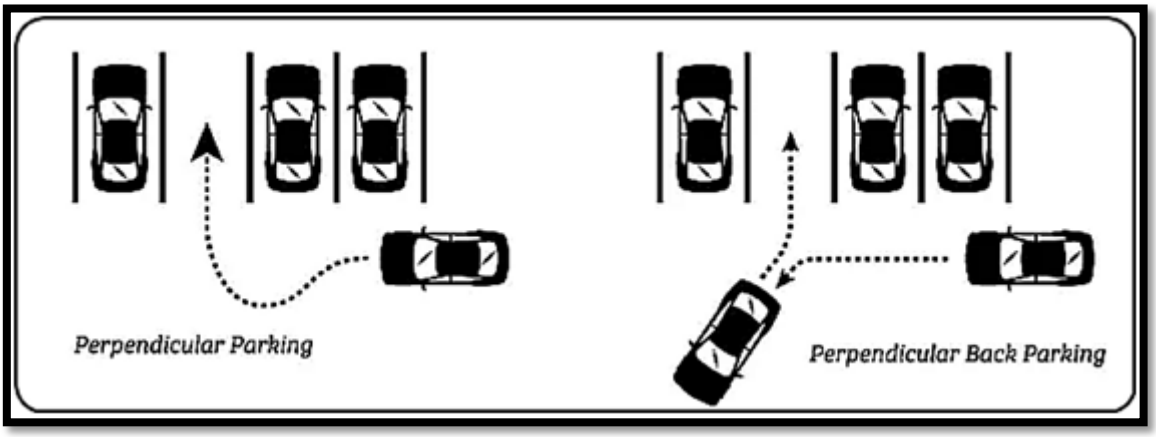


Figure 2: Perpendicular Parking Scenarios

The proposal seeks to address fundamental questions such as:

How can we construct a system capable of parking a vehicle on its own?

What factors contribute to successful navigation in various parking scenarios?

How efficient will the system be? By honing in on these challenges, the paper aims to contribute to the development of robust and adaptive RL models capable of handling the

problems associated with automated car parking. Moreover, it will inform about the challenges of the field and whether the ML and RL approach is feasible and effective or not.

Significance:

The paper is directly aimed at the advancement of autonomous vehicles and how to execute the car parking task autonomously, highlighting its significance in the field. Automated parking systems are an integral part of broader self-driving technology and demand intelligent agents capable of making swift and accurate decisions in real time.

By employing RL techniques, this project endeavors to create a model that not only learns optimal parking strategies but also adapts to diverse scenarios, show casing the adaptability necessary forreal-world applications. Moreover, the RL model will be evaluated in both training and testing phases which further outlines its efficiency and relevancy as a proposed solution. The paper also includes the prospect of the proposed solution, which provides a peek into the future of autonomous parking, it will also lay out the disadvantages of our approach which will further aid any research in the field. The paper, as stated earlier, makes use of Unity3D Engine and Unity ML-Agents frame work. This shows the effectiveness of the software and the framework in the field of machine learning-oriented research. The above-mentioned points promoted the need and proved the significance of this paper, also to conduct research andproposenewmoreeffectiveandfeasible solutions by other fellow researchers.

Objectives:

Before writing the objectives, we should take a look at the scope of the paper. This Increasing understanding of objectives and expectations is crucial for better planning and decision-making. The scope of this paper encompasses the development of an autonomous car parking system using Reinforcement Learning (RL) within the Unity simulation environment. The primary focus is on training a virtual agent to autonomously navigate and park a car in diverse scenarios, emulating real-world challenges. The system will address various aspects of automated parking, including spatial awareness trajectory planning, and real-time decision-making. We will also evaluate the resulting model in the training and testing phase, and conclude with the results and future uses or alterations. We will also briefly discuss the impact of this approach on the field and what areas should future researchers pay utmost attention to. Now, various objectives of this paper include:

Develop an RL model tailored for car parking in Unity, integrating state-of-the-art algorithms to enable effective learning and decision-making.

Design and implement a simulation environment within Unity that encompasses a range of parking scenarios, capturing the complexities of real-world parking challenges.

Train the RL agent to navigate and park a virtual car autonomously,

- Emphasizing adaptability to different parking space configurations and dynamic environments.
- Evaluate the performance of the trained RL agent based on key metrics,
- Including success rate, parking accuracy, and computational efficiency.
- Contribute insights to the broader field of ML applications in simulated
- Environments, offering solutions and methodologies for training RL agents in complex tasks, specifically in the context of automated car parking.

2. The Related Work:

Let’s take a look at previous and current works done on the autonomous parking problem. Most of the work is done utilising single and multi-agent Reinforcement Learning, and Machine Learning techniques. However, few researchers have also implemented Unity3D and Unity ML-Agents framework for the task. Some of the previous works are:

• Clara Barbu and Stefan Alexandru Mocanu: On the development of Autonomous Agents using Deep Reinforcement Learning.[BM21] : The pa- per presents a general study of autonomous agents with their development powered by deep reinforcement learning. This is combined with autonomous vehicles viaan example of a vehicle agent parking autonomously in the virtual parkingenvironmentprovidedbytheUnity3DEngine.TheagentisutilisingDeepQ-Learning, Double DeepQ-Learning ,and Experience Replay.

Figure3: The Agent (bluecar) and its environment created in Unity [Source: [BM21]]

The paper resulted in a model (Fig 3.1) able to park a car using Deep Q- Learning techniques, but the model took more than 72 hours to train. The results for a more general application (Ball-Cube) were more promising and quick utilising the Double Deep Q-Learning.

• Mohamed Fethi Dellali and Mohamed El Mahdi Bouzegzeg: Autonomous Parking Simulation using Unity Game Engine and Reinforcement Learning.[DB22] : The report implemented an autonomous parking

• Simulation using Unity3D Game Engine, Unity ML-Agents framework, and Reinforcement Learning. They started with a discussion of various artificial intelligence (AI) subsets and their methods, followed by a detailed discussion of reinforcement learning, Unity game engine, and ML-Agents.

• The report resulted in a model which can seek out the empty parking lot in a parking area and execute the parking task correctly. The model trained for over 12 million steps in 12 hours with a theoretical success rate of 97% during the training phase.Omar Tanner: "Multi-Agent Car Parking using Reinforcement Learning" [Tan22]: The paper aimed to train a model able to perform in a multi-agent system (Fig. 3.2) where other cars can also communicate and aid in parking tasks.

Figure4:Multi-Agent System for Autonomous Parking [Source:[Tan22]]

The fixed goals and obstacles environment yielded a model using up to 7 agents with a success rate of 98.1%. If the model only implemented 2 agents instead of 7, the rate bumped up to 99.3%. This proved the effectiveness of the multi-agent system in the autonomous parking scenario.

•Yusef Savid, Reza Mahmoudi,RytisMaskeliu¯nas,and Robertas Damaˇseviˇcius: Simulated Autonomous Driving Using Reinforcement Learning: A Comparative Study on Unity's ML-Agents Framework" [SMMD23]: The paper compares the performance of several different RL algorithms and configurations on the task of training kart agents to successfully traverse a racing track (Fig. 3.3) and identifies the most effective approach for training.Tain agents to navigate a racing track and avoid obstacles on that track.

Figure5: The Race Track environment in Unity [Source:[SMMD23]]

The paper also explored the effectiveness of behavioral cloning; a technique of copying human skills or inputs and training the model to closely mimic them, in the area of racing simulators. The results when compared to the Proximal Policy Optimization algorithm, notice donlya deviation of 23.07%invalue loss and only a 10.64% deviation in cumulative reward, hence confirming the usefulness of behavioral cloning for improving the performance of intelligent agents for racing tasks. Nowlet' sproposeour solution for Autonomous Parking:

We address the situation through the employment of Reinforcement Learning, specifically leveraging the Proximal Policy Optimization (PPO) algorithm. The utilization of Unity ML-Agents will be integral in executing this RL model using the PPO. Within the Unity Engine, the model will undergo training, testing, and evaluation. The agent will rely on the "Ray Perception Sensors" component provided by Unity to perceive the environment, mimicking the functionality of real-life Lidar sensors. Access to a Car Controller script will be granted to the agent, enabling it to execute actions such as driving, steering, and braking. Positioned within a dynamic simulation environment, the agent will face ever-changing scenarios in each episode to enhance the adaptability of the RL model. The reward system designed for the agent will incentivize "reverse parking" as opposed to traditional front parking, aiming to instill good parking etiquette. Moreover, penalties will be imposed on the agent in case of collisions to reinforce task completion. Evaluation of the agent will be conducted in two manners: assessing parking success and inspecting model parameters. The former evaluation involves analyzing the "Efficiency Percentage" of successful parking instances within a set timeframe encompassing both training and testing phases. The latter evaluation refers to utilizing Tensor board to monitor model parameters like extrinsic reward, episode length, policy loss, etc., throughout the training process. The ultimate results will amalgamate insights from both evaluations, leading to a conclusive statement. This academic discourse is influenced by references from prestigious Scopus indexed IEEE & Springer conference papers.

The Tools & Technologies Used

Let’s look into brief details of specific tools (Unity Engine and Unity ML-Agents)and software technologies or methods (Reinforcement Learning, Artificial Neural Networks, and Proximal Policy Optimization) we have utilised for our work. Unity Engine

Unity is a cross-platform game engine developed by Unity Technologies, first announcedandreleasedinJune2005atAppleWorldwideDevelopersConferenceas a Mac OS X game engine [Wik].Over the years, it has grown into a cross-platform powerhouse, supporting development for a multitude of devices and platforms, from desktop and mobile to consoles and virtual reality.

Key Features:

Key Features of the system encompass a wide range of training scenarios and environmental setups, catering to diverse needs and requirements across different domains. Various advanced Deep Reinforcement Learning algorithms such as Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), Multi-Agent Proximal Policy Optimization with Centralized Critics (MA-POCA), and self-play mechanisms are provided to facilitate the training of agents in single-agent, multi-agent cooperative, and multi-agent competitive settings. Moreover, users are offered support in leveraging two distinct imitation learning algorithms, namely Generative Adversarial Imitation Learning (GAIL) and Behavioral Cloning (BC), to effectively learn from expert demonstrations and improve performance. Additionally, the platform is equipped to handle concurrent training sessions involving multiple instances of the Unity environment, enabling a parallelized training approach that enhances efficiency and throughput without compromising the system's ability to adapt to different scenarios and configurations. This concurrent training capability significantly accelerates the overall training process, allowing users to expedite the learning curve and achieve optimal results in a timely manner.

Reinforcement Learning

Reinforcement Learning can be defined as a technique for problem-solving where an intelligent agent is trained using experiences. The agent will be put in the problem environment at a particular state or situation, where it can perform certain actionsthatwillgeneraterewardsorpenaltiesandtransferitintoanewstate.Astatecan be defined as

a particular scenario in a problem and used by the agent to perform actions and get to a solution.The reward is a positive incentive the agent receives when it comes close to the desired output whereas the penalty is a negative reward which is given when the agent either deviates from the solution or makes a blunder. Penalties are significantly higher than the rewards to make sure the agent never repeats the negative actions.

The reinforcement learning process generally results in a model capable of per-forming the task it was trained for with great efficiency.The model is typically represented with an artificial neural network, a multilayer feed-forward neural net- work in our case, which has node functions and weights calculated according to its learned behaviour from the training phase.

Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of biological neural networks in the human brain. ANNs consist of interconnected nodes, or neurons, organized in layers, allowing them to learn complex pattern sand relationships from data.

Feed forward Neural Networks: Feed forward Neural Networks (FNNs) are a type of artificial neural network where the information moves in only one direction—from the input layer, through any hidden layers, to the output layer. There are no cycles or loops in the network, meaning the information does not flow backward. Here are some key points about FNNs.

Proximal Policy Optimization

Based on policy gradient approaches, proximal policy optimization (PPO) seeks to maximizepredictedcumulativerewardsbyrepeatedlyimprovinganagent'spolicy.

Fundamentally, PPO makes use of a surrogate objective function to direct policy updates while guaranteeing effective and consistent learning dynamics.

The Architecture & Working

Within the Unity Editor, the solution has the capability to function in two distinct modes: Training and Testing. In the Training mode, the agent is devoid of any neural network guidance, starting the training process independently. To initiate the training, a specific command is executed in the Anaconda command prompt, setting the run-id as "modelNameOrId." This run-id serves the purpose of defining the model name and is utilized by tensorboard to display model statistics. The behavior exhibited during this training phase is governed by the trainer config.yaml file, which outlines actions like periodic exports of the Neural Network and the creation of checkpoints. These actions play a critical role in preserving the progress made during the training process.

On the other hand, in the Testing mode within the Unity Editor, the presence of a neural network is indispensable. This neural network, also known as the model, is then subjected to various parking scenarios to evaluate its performance. The evaluation of performance takes place on an individual agent basis, with the assistance of Efficiency Cal.cs, and collectively for all agents, managed by Efficiency Comb.cs. This evaluation process continues indefinitely and can only be interrupted by choosing the "Stop" option within the Unity.

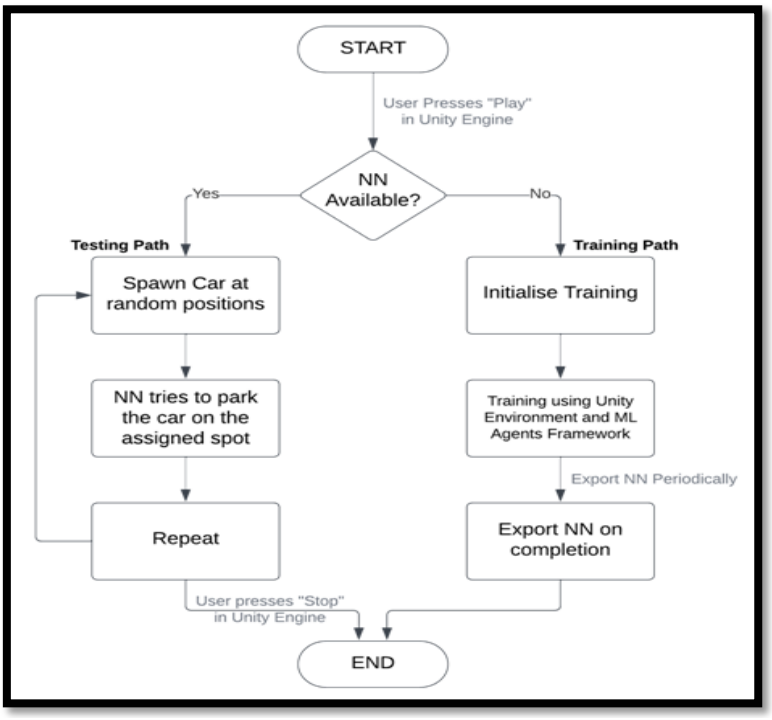


Figure6: Flow chart showing the working in Unity Editor

In the exported application, the functionality of the solution is exclusively limited to testing mode, wherein the initiation of the testing process is triggered by the user's selection of the "Start" option and subsequent selection of a model from the list of available options. The application is equipped with a diverse range of 7 models for the user's consideration, encompassing 4 development models and 3 export models, thereby providing a comprehensive array of choices for testing and evaluation. The inclusion of these various models serves to enhance the user experience by offering a wide selection of options to explore and engage with, thereby maximizing the utility and effectiveness of the testing phase.

Models, all saved on the drive, are available for selection. Following this selection, the "Final Scene" is loaded, functioning akin to "Unity Editor: Testing Mode." Moreover, users can opt to "Reset" the scene, "Go Back" to the main menu to try another model, or "Quit" the application

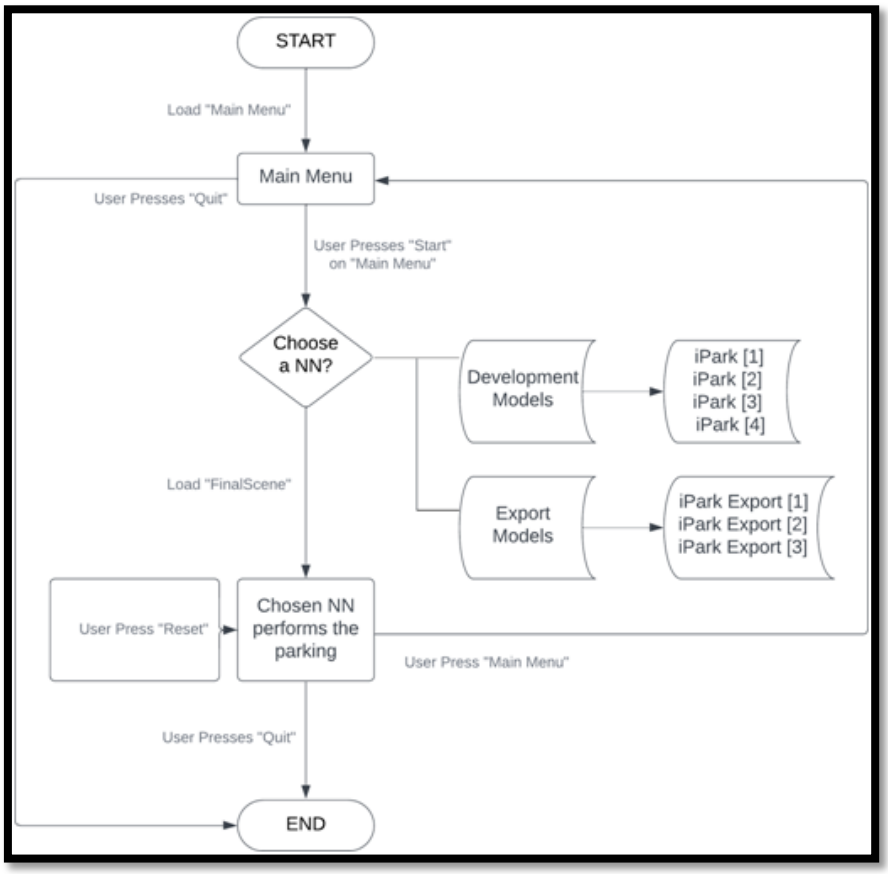


Figure7: Flow chart showing the working in exported Application

3. The Results:

We created a total of 7 models for this purpose and compared training and testing resultsofeach.Thetrainingresultsmainlyconsistedofmodelorpolicyparameters such as the "Cumulative Reward", "Episode Length", "Policy Entropy" etc. Additionally, we calculated the parking efficiency during training. The testing results is the parking efficiency of the model over an evaluation period of 4hrs.Let's define the training specific terms first then take a look at the training and testing results for our most effective model (i-Park Export [02]).At the end, we will discuss the distribution and how the reader can use the application himself.

Training Results Related Terminology

1. Cumulative Reward: Cumulative Reward in Tensor Board graph stracks the total reward obtained by the agent during training or evaluation. It offers aquick overview of the agent's overall performance and its ability to achieve goals within the environment.
2. Episode Length: Episode Length in Tensor Board graphs represents the du-ration of each episode during training. It indicates how long the agent interacts with the environment before reaching a terminal state or completing a task. Tracking episode length helps monitor the efficiency and effectiveness of the agent's decision-making process over time.
3. Policy Loss: Policy Loss in Tensor Board graphs reflects the discrepancy between the predicted actions of the agent and the optimal actions determined by the policy during training. It measures how well the agent's policy approximates the desired behavior and provides insights into the training progress and stability of the reinforcement learning algorithm.
4. Value Loss: In Tensor Board, the Value Loss metric tracks the error between predicted and observed returns during training. Good performance shows a decreasing trend over time,

indicating improved accuracy in predicting future rewards. Fluctuations may occur, but overall, the curve should converge to a low and stable level, signaling successful learning by the agent.

5. Policy Entropy: Policy Entropy in Tensor Board measures the uncertainty or randomness of the agent's action selection. A good agent should maintain a moderate level of entropy to encourage exploration and prevent premature convergence to suboptimal policies. An ideal scenario shows a decreasing trend in entropy as the agent learns to make more confident and informed decisions over time, but without diminishing too quickly, ensuring a balance between exploration and exploitation.

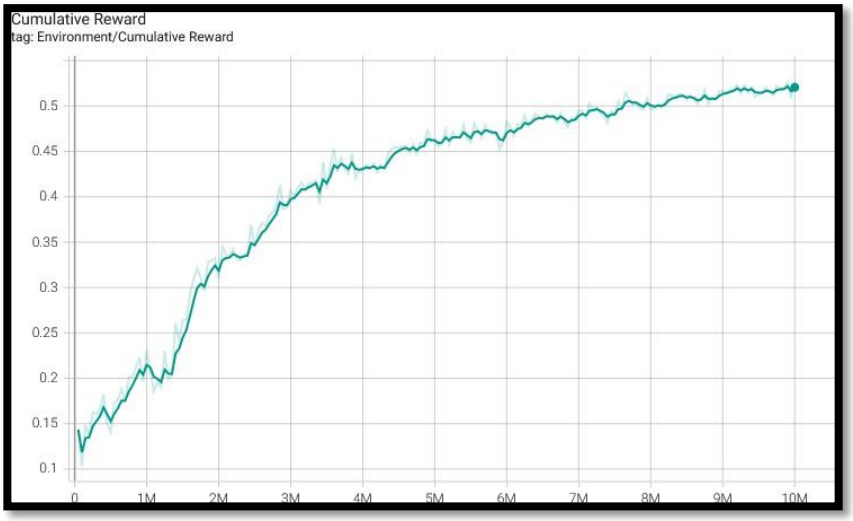


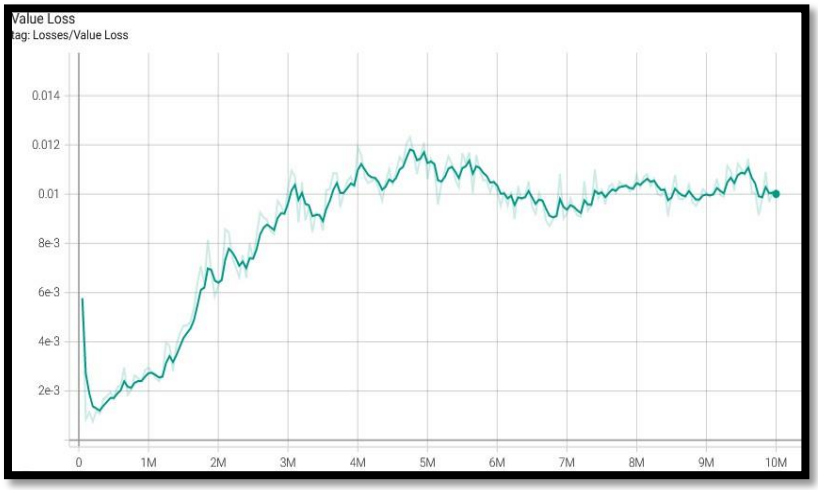
Figure8: Cumulative Reward for I Park Export [02]

The cumulative reward graph starts at 50k steps with a reward value of 0.1434, and endsat10Mstepswithavalueof0.5269in3hours28mins.Thisshowsclearly that the model is learning to park. The graph is increasing steadily throughout the period. This shows that the model was learning new behaviours and did not mature early.

Figure9: Episode Length for i-Park Export[02]

The episode length graph starts at 50k steps with an episode length of 242 and ends at 10M steps with a length value of 32 (32.37). This shows that the model was learning new optimal behaviors and was able to park with fewer steps in each episode. Moreover, the graph was almost flat from 4M steps (36.35), indicating that the model was able to find optimal settings very early.

Figure10: Policy Loss for I Park Export[02]



The policy loss graph starts at 50k steps with a value of 0.03426 and ends at 10M steps with a value of 0.03257. This decrease in value shows that the model was able to find an optimal policy function. Moreover, the graph is constantly declining, indicating that the model was improving throughout the training period. The fluctuations show that the agent is learning from the environment

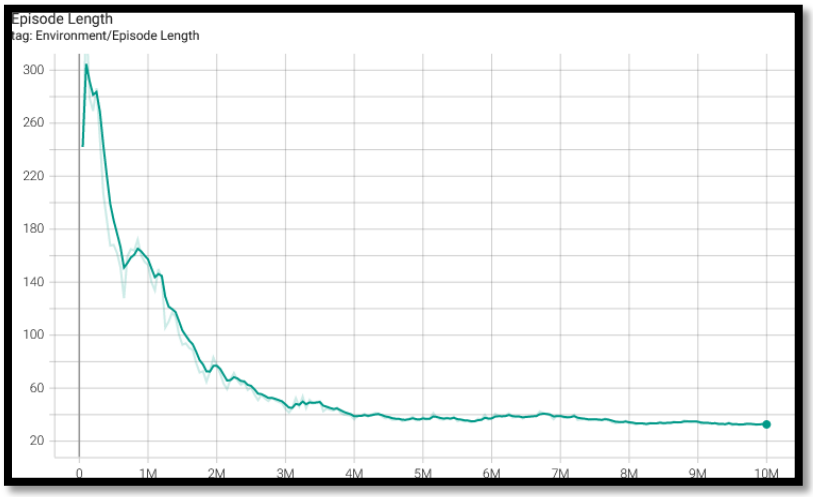


Figure11:Value Loss for i-Park Export[02]

The value loss graph starts at 50k steps with a value of 5.768e-3 (0.005768) and ends at 10M steps with a value of 9.9214e-3 (0.0099214). Value loss shows the difference between the predicted value of state-action pairs by the agent's value function and the actual observed returns received during training. An ideal behavior will be a decreasing or constant graph. The graph initially increased till 4.75M steps,but then it declined and stayed continuous from 7M steps. Morover, the overall increase was very low (0.0041534) which shows the model really performed well in value loss and find an optimal solution.

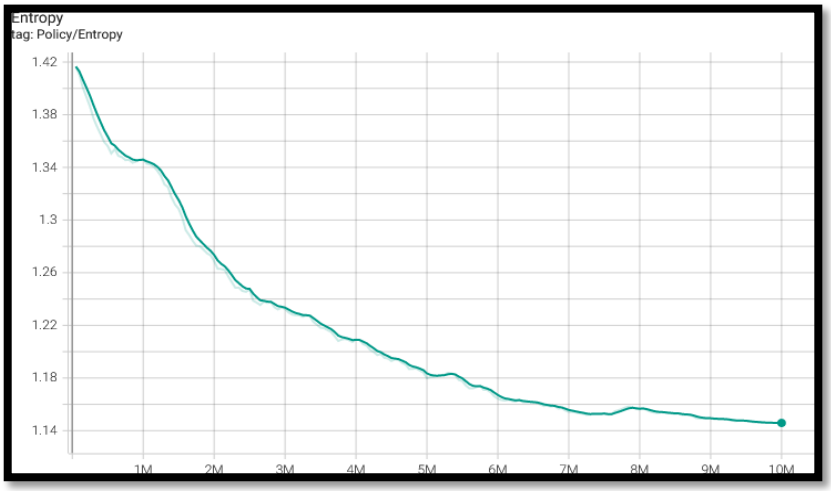


Figure12: Policy Entropy for i-Park Export[02]

The policy entropy graph starts at 50k steps with a value of 1.417 and ends at 10M steps with a value of 1.146. The decreasing trend is favorable here. It shows that the agent was able to balance between exploration and exploitation without converging to a sub-optimal solution.
Training Parking Efficiency

The training efficiency is indeed reported in the "Efficiency.txt" file by the performance metric component
08-05-202415:30:55(Training Model)
Efficiency 79.27053%
Total Park170392
Total Collision 44558
Total Cases 214950
The statement is correct. The Efficiency Cal and Efficiency Combined scripts report the efficiency of the model for the entire training period, and it may differ from real testing scenarios.
Graph Analysis
The cumulative graph demonstrates significant progress, with the final value exceeding the average (0.5269). Notably, the training period was the shortest among all, showcasing efficient learning. However, relying solely on this metric may not accurately predict the model's performance in test scenarios. Similarly, the episode length graph displays an ideal trend, with the final value being among the lowest, indicating the successful optimization of episode length by the model..The policy loss graph also exhibits a consistent decline, reflecting effective policy improvement throughout training. However, the value loss presents a challenge, with a continuous rise for most of the training period, though the model managed to stabilize it towards the end. There are challenges, but the policy entropy is ideal, which indicates the Proximal Policy Optimization (PPO) algorithm is working well.
Training efficiency data looks promising, but it might not directly correlate with how the model performs in testing.
Testing will provide more insights into the model's true performance.
I-Park Export[02]Testing Results
#training efficiency report edinthe
"Efficiency.txt"by the performance metric component
16-05-202414:34:04(Testing Model "i-Park Export[02]
-10000076(Unity. Barracuda.NNModel)")
Efficiency89.37852%
TotalPark31539
TotalCollision3748
TotalCases35287
The 4-hour evaluation test resulted in the agent attempting a total of 35,287 parking scenarios. It successfully parked 31,539 times and collided 3,748 times. This results in an 89.38% success rate, which is 10.11% higher than the training data's success rate.

Data Analysis

The 4-hour evaluation test resulted in the agent attempting a total of 35,287 parking scenarios. It successfully parked 31,539 times and collided 3,748 times, resulting in a success rate of 89.38%. This is 10.11% higher than the training data's success rate, representing the highest improvement recorded so far. The model's exceptional performance across all training parameters suggests it is the best-suited model for the task.

All model comparison

Table1: Comparing models on the irrespective training and testing data

Training Efficiency		Testing Efficiency	
Model Name	**Efficiency %**	**Model Name**	**Efficiency %**
iPark[01]21-11-2023	–	iPark[01]21-11-2023	78.56705%

iPark[02]28-03-2024	74.95232%	iPark[02]28-03-2024	84.407%
iPark[03]29-03-2024	84.70142%	iPark[03]29-03-2024	86.45386%
iPark[04]30-03-2024	79.63393%	iPark[04]30-03-2024	88.92231%
iPark Export[01]04-05-2024	75.3954%	iPark Export[01]04-05-2024	85.36852%
iPark Export[02]06-05-2024	79.27053%	iPark Export[02]06-05-2024	89.37852%
iPark Export[03]08-05-2024	79.2570%	iPark Export[03]08-05-2024	88.61481%

Deployment

The deployed application setup can be found here:
iParkSetup.exe [https://github.com/KushagraYashu/iPark/releases/download/setup /iParkSetup.exe]
After downloading, the program can be installed by running and following the in striations in the setup.

4. The Conclusion

In conclusion, this research successfully demonstrates the viability and effectiveness of employing Reinforcement Learning (RL) agents within the Unity3D environment to tackle the complex problem of autonomous parking. By integrating the Unity ML-Agents framework, we have shown that virtual agents can be trained to navigate and park vehicles autonomously in a variety of scenarios that closely mimic real-world parking situations real-world conditions. The RL approach, particularly within the robust and versa- tile Unity simulation, proved to be a powerful method for developing adaptive and intelligent parking systems.

Throughout this study, the RL agents displayed significant capabilities in spatial awareness, trajectory planning, and real-time decision-making. The performance metrics, including success rate and parking accuracy, indicated that the RL agents could consistently and efficiently execute parking maneuvers across different configurations and dynamic environments. These results underscore the potential of RL techniques in advancing autonomous vehicle technologies, particularly in enhancing the functionality and reliability of self-parking systems.

Moreover, this research contributes valuable insights into the broader application of machine learning in simulated environments, offering a blueprint for future studies aiming to train RL agents for complex tasks. The use of Unity3D as a simulation platform not only provided a realistic training ground but also facilitated the exploration of various parking scenarios, thereby enhancing the agent's learning process

Looking ahead, there are several avenues for further research. Future work could explore the integration of additional sensors and real-world data to improve the realism and robustness of the simulation. Additionally, investigating other RL algorithms and hybrid approaches could further enhance the efficiency and effectiveness of the autonomous parking system. The insights gained from this study pave the way for more sophisticated and adaptable autonomous systems, contributing to the ongoing evolution of intelligent transportation solutions.

In summary, the development and evaluation of the autonomous parking system utilizing RL agents within the Unity3D environment demonstrate a promising step forward in the realm of autonomous vehicle technology. The findings from this research highlight the practical

applications of RL and set the stage for future in- novations in automated parking and beyond.

5. References:

1. Clara Barbu and Stefan Alexandru Mocanu. On the development of Autonomous Agents using Deep Reinforcement Learning.*Scientific Bulletin, University Politehnica of Bucharest*, Vol 83(Series C):97–115,2021.
2. Mohamed Fethi Dellali and Mohamed El Mahdi Bouzegzeg.Au-tonomous Parking Simulation using Unity Game Engine and Reinforce- ment Learning.*Univ Blida*, 2022.
3. Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen,Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange.Unity: A General Platform for Intelligent Agents.*arXiv:1809.02627v2*, 2020.
4. Richard S. Sutton and Andrew G. Barto.*Reinforcement Learning: An Introduction*.Second Edition. MIT Press, 2018.
5. Yusef Savid, Reza Mahmoudi, RytisMaskeliu ̄nas,and Robertas Damaševičius.Simulated Autonomous Driving using Reinforcement Learning:A Comparative Study on Unity's ML-Agents Framework.*In- formation*, 14(5):290, 2023.
6. OmarTanner.Multi-AgentCarParkingusingReinforcementLearning. *Book of Abstracts, ICUR 2022, ArXiv 2206.13338*, 2022.
7. Yogendra Narayan Prajapati, U. Sesadri, T. R., M. ., Shreyanth S., Ashish Oberoi, & Khel Prakash Jayant. (2022). Machine Learning Algorithms in Big Data Analytics for Social Media Data Based Sentimental Analysis. *International Journal of Intelligent Systems and Applications in Engineering*, *10*(2s), 264.
8. A REVIEW PAPER ON CAUSE OF HEART DISEASE USING MACHINE LEARNING ALGORITHMS. 2022). *Journal of Pharmaceutical Negative Results*, 9250-9259.
9. Prajapati, Y.N., Sharma, M. (2024). Novel Machine Learning Algorithms for Predicting COVID-19 Clinical Outcomes with Gender Analysis. In: Garg, D., Rodrigues, J.J.P.C., Gupta, S.K., Cheng, X., Sarao, P., Patel, G.S. (eds) Advanced Computing. IACC 2023. Communications in Computer and Information Science, vol 2054
10. Y. N. Prajapati and M. Sharma, "Designing AI to Predict Covid-19 Outcomes by Gender," *2023 Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, Chennai, India, 2023, pp. 1-7, doi: 10.1109/ICDSAAI59313.2023.10452565.
11. Azam, Farooque, et al. "Enhancing COVID-19 Diagnosis and Severity Evaluation through Machine Learning Algorithms Applied to CT Images." *Multidisciplinary Science Journal* Accepted Articles (2024.