BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Project report on

Generating Game Character Behaviours Using Machine Learning

Submitted

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Name	USN
Kunal Jadhav	01FE17BCS099
Anusha K M	01FE17BCS040
Mohammed Zeeshan	01FE17BCS109
Jyotiba Mane	01FE17BCS088

Under the guidance of

Prof. Namrata Hiremath

Professor

School of Computer Science and Engineering

KLE Technological University, Hubballi

2020-2021

BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2020-21

CERTIFICATE

This is to certify that project entitled "Generating Game Character Behaviours Using Machine Learning" is a bonafied work carried out by the student team Mr. Kunal Jadhav USN:01FE17BCS099, Ms. Anusha K M USN:01FE17BCS040, Mr. Mohammed Zeeshan USN:01FE17BCS109, and Mr. Jyotiba Mane USN:01FE18BCS088, in partial fulfillment of the completion of 8th semester B. E. course during the year 2020 – 2021. The project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said course.

Guide Name
Prof. Namrata Hiremath

SoCSE Head
Dr. Meena S. M.

External Viva-Voce

Name of the examiners Signature with date

1 _____ _____

2 _____ _____

ABSTRACT

Model-free deep reinforcement learning (RL) algorithms have been successfully applied to a range of challenging sequential decision making and control tasks. However, these methods typically suffer from two major challenges: high sample complexity and brittleness to hyper-parameters. Both of these challenges limit the applicability of such methods to real-world domains.In this project we have proposed a framework which involves Imitation learning and Curiosity-driven learning. This framework involves GAIL(Generative Adversarial Imitation Learning), Behavioral Cloning, PPO(Proximal Policy Optimization) and Curiosity-driven learning. We have tested our framework on various environment and the results are very promising than the existing frameworks.

Keywords : *GAIL,PPO,Behavioral Cloning, Curiosity-driven learning.*

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of a number of individuals whose professional guidance and encouragement helped me in the successful completion of this report work.

We take this opportunity to thank Dr. Ashok Shettar, Vice-chancellor, KLE Tech and to Dr. Prakash Tewari, Principal, B V Bhoomaraddi College of Engineering And Technology, Hubli.

We also take this opportunity to thank Dr.Meena S M, Professor and Head of Department, Department of Computer Science and Engineering for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We sincerely thank our guide Ms Namrata Hiremath,Professor, Department of Computer Science and Engineering for his guidance, inspiration and wholehearted co-operation during the course of completion.

We sincerely thank Dr. P G Sunita Hiremath, Department of Computer Science and Engineering for her support, inspiration and wholehearted co-operation during the course of completion.

Our gratitude will not be complete without thanking the Almighty God, our beloved parents, our seniors and our friends who have been a constant source of blessings and aspirations.

Kunal Jadhav - 01FE17BCS099
Anusha K M - 01FE17BCS040
Mohammed Zeeshan - 01FE17BCS109
Jyotiba Mane - 01FE17BCS088

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

The key element of realism in video games is the behavior of computer controlled characters.Gameplay becomes unique, when artificial intelligence (AI) is a part of the immersive, virtual environment. In video games, every snippet of code that targets simulating "intelligent" behavior of virtual players is considered AI.Such software is most commonly based on a mere illusion of intelligence produced by a skillful usage of game design techniques such as NPCs, which harness simple controlling algorithms, realistic graphics, convincing character animations and voices borrowed from famous film actors.A non-player character (NPC) is a character that is not directly controlled by a player in a game. These characters make up part of the game world in games. A game world in a game is essentially the setting of the game.

1.1 Motivation

- Modern AAA games do not have scalable AI methodologies with machine learning.
- In recent games like Cyberpunk 2077, the AI is not responsive and this makes the game experience less immersive.
- The programming techniques are static and this causes unnatural behavior in NPCs.
- It is an emerging area of research with game development industries striving to make advancements in using machine learning for programming game characters.
- There is a need for better QA methods that can bring out bugs in games with little to no human intervention.
- There is complexity in the current system of game character programming which makes the development of better player behaviors difficult.

1.2 Objectives

- To make a series of gaming environments to train and test our model in conditions of varied complexities as seen in applications in the industry.
- To maximize reward function to achieve optimal behavior to train game characters.
- The model should improve training time as compared to existing machine learning techniques.
- To visualize results and showcase the learning process in addition to human-like performance in the games.

1.3 Literature Survey

[1]Giovanna Mart´ınez-Arellano, Richard Cant, and David Woods, "Creating AI Characters for Fighting Games using Genetic Programming",20 December 2016

- This paper devises a character generation approach for the M.U.G.E.N. fighting game that can create engaging AI characters
- M.U.G.E.N. game engine is used here which is an open-source 2D game-making platform.
- The approach uses a Genetic Programming algorithm that refines randomly generated character strategies into better ones using tournament selection.
- The procedure is capable of creating a wide diversity of players with different strategic skills, which could be potentially used as a starting point to a further adaptive process.
- The approach is suitable for low computation capacity and technical expertise.
- This project also shows various approaches to take observations from the game's state to contribute to effective training.

[2]Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine, "Soft Actor Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", 8 August 2018.

- An off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework is developed here.
- In this methodology, the actor aims to maximize the expected reward while also maximizing entropy. That is, to succeed at the task while acting as randomly as possible
- By combining off-policy updates with a stable stochastic actor-critic formulation, this method achieves state-of-the-art performance
- Demonstrated that, in contrast to other off-policy algorithms, this approach is very stable.

[3]Jonathan Ho, Stefano Ermon, "Generative Adversarial Imitation Learning", 10 June 2016

- In this paper, they have proposed a new general framework for directly extracting a policy from data following inverse reinforcement learning
- They have evaluated GAIL against baselines on 9 physics-based control tasks. Like the cart pole, acrobat, mountain car, HalfCheetah, Hopper, Walker, Ant, 3D humanoid locomotion, and Reacher.
- GAIL was tested against 3 baselines - Behavioural Cloning, FEM, GTAL.
- In the classic control tasks (cart pole, acrobat, and mountain car), behavioral cloning generally suffered in expert data efficiency compared to FEM and GTA.
- On these tasks, GAIL consistently produced policies performing better than behavioral cloning, FEM, and GTA.
- GAIL almost always achieved at least 70% of the expert performance for all dataset sizes tested and reached it exactly with the larger datasets, with very little variance among random seeds.

[4]Marek Kopel and Tomasz Hajas, "Implementing AI for non-player characters in 3D Video Games", January 2018.

- This paper proposes a solution for implementing the intelligent behavior of NPC agents (non-player characters) in video games.
- Four approaches to NPC AI implementation are compared: decision tree, genetic algorithm, Q-learning, and a hybrid method.
- Best results can be achieved using a hybrid method consisting of Multilayer Perceptron Neural Network with Q-Learning(QL) rule and Genetic Algorithm(GA).
- These methods are helpful for us as a frame of reference to understand the performance of other existing solutions.

1.4 Problem Statement

To implement a machine learning methodology to generate character behaviors in variable game environments.

Chapter 2

REQUIREMENT ANALYSIS

The Software Requirements Specifications (SRS) is a description of a software system to be developed. It lays out functional and non-functional and may include a set of use cases that describe user interactions that the software must provide.
Software Requirements Specifications establish the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It also provides a realistic basis for estimating product costs, risks, and schedules.
Used appropriately, software requirements specifications can help prevent software project failure.

2.1 Functional Requirements

A functional requirement defines a function of a system or its components. Functional Requirements can be calculation, technical details, data manipulation and processing, and other specific functionality that defines what a system is supposed to accomplish.

- **User:**
  - User shall be able to start the gaming environment and simulate agent learning.
  - User shall be able to train and draw an inference based on various configuration parameters.
- **System:**
  - The system shall generate a game environment and capture observations for training.
  - System shall compute state action pairs according to a particular policy for the agent to take in the next time step

2.2 Non Functional Requirements

A non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. The plan for implementing non-functional requirements is detailed in the system architecture because they are usually Architecturally Significant Requirements.

- **Speed:** Response time should be optimum.
- **Adaptability:** Suitable for changes in The system environment
- **Reliability:** The learning algorithm should be reliable.

2.3 Software and Hardware Requirement Specification:

It specifies what all the software and hardware required for smooth functioning of web-application.

2.3.1 Software Requirements

- **OS:** Windows XP and above
- **Software:** Unity, Google Chrome.
- **Packages:** Pip, mlagents, tensorboard

2.3.2 Hardware Requirements

- **Processor:** Intel Pentium III
- **Graphics:** NVIDIA GeForce FX 5200.
- **System Memory:** 256 MB RAM.

Chapter 3

SYSTEM DESIGN

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture, and systems engineering. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

3.1 Architecture Design

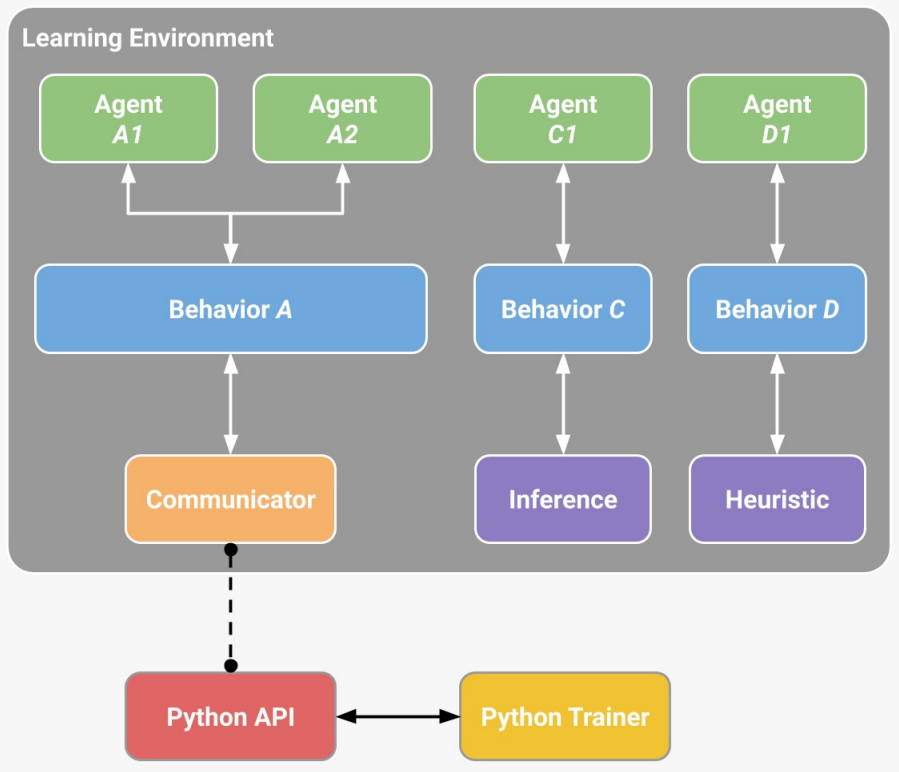

Figure 3.1: Architecture diagram

3.2 Level 0 DFD

A data-flow diagram (DFD) is a way of representing the flow of data of a process or a system (usually an information system).
The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

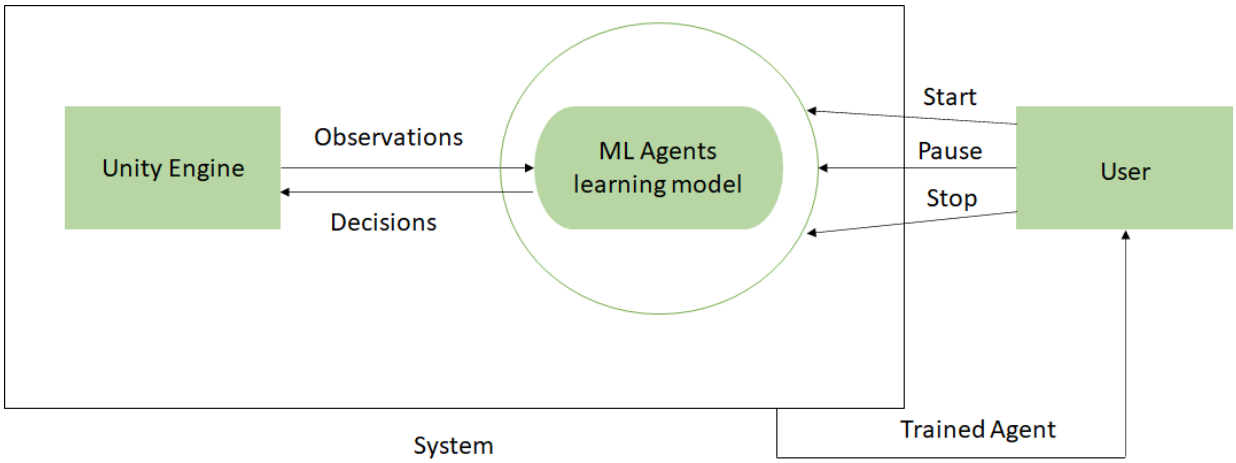

Figure 3.2: Level 0 DFD diagram

3.3 Level 1 DFD

A data flow diagram (DFD) graphically describes the flow of data within a System. It is used to document existing systems and to plan and design new ones. Generally, there is no ideal way to develop a DFD. However it is composed of the following four basic elements: data sources and destinations, data flows, transformation processes, and data stores. In this study, the DFD shows how information is delivered by the Agent from various states, Rewards and Actions.
It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

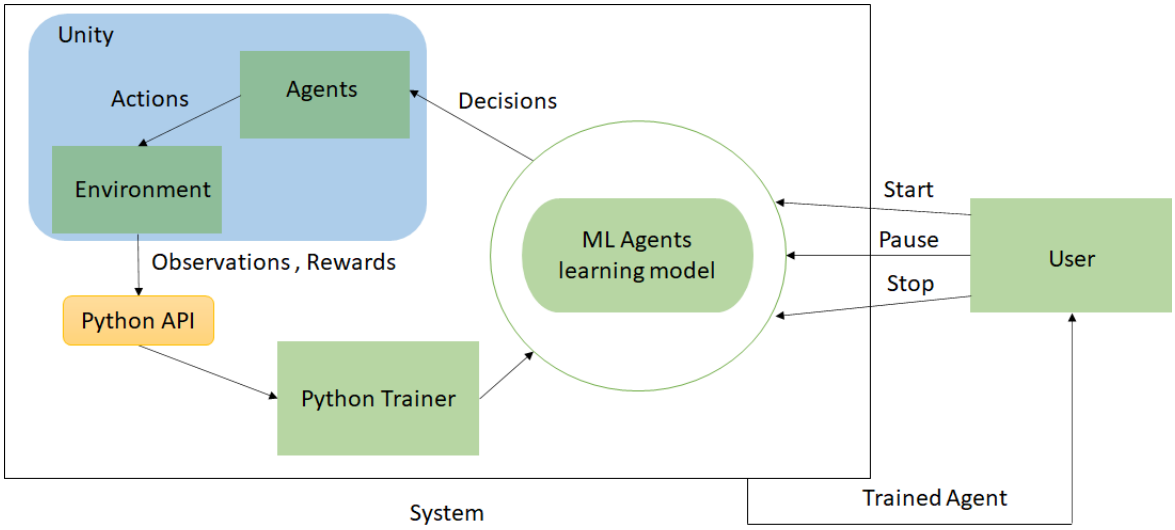

Figure 3.3: Level 1 DFD diagram

Figure 3.3 represents the Data Flow Diagram Level 1 for the System. When the User starts the Game, Agent gets Trained and Actions are performed. Every action performed will have rewards and depending on rewards Optimal Actions will change its States which can also be called as State-Action Pair, These Actions will be analyzed with an Artificial Neural Network which will create Desired Output. The Outputs will Train the Agent using the O-Learning Model This will create the Trained Agent with maximum Rewards.

Chapter 4

IMPLEMENTATION

This chapter gives a brief description about implementation details of the system by describing each environment.

4.1 Shooter

Figure 4.1 represents the training environment of the Shooter game. In this game the agent ie. the shooter is expected to aim at the enemy and shoot it. The action space includes three discrete actions, which are rotate left, rotate right and shoot. In one episode three enemies will enter one by one, So the shooter has to kill all three enemies in one episode in order to gain the maximum cumulative reward. Here +1 reward is given for each successful shot and -1 for each hit from the enemy.

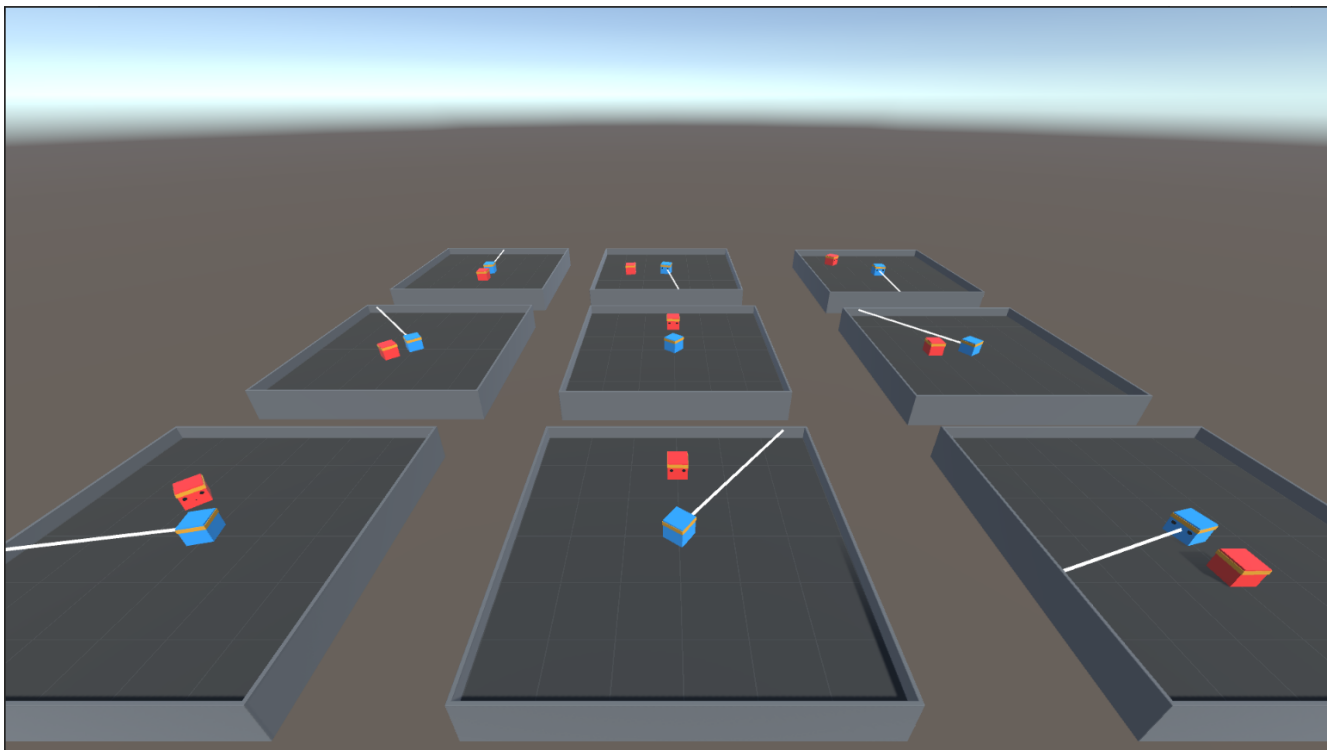

Figure 4.1: Shooter training environment.

We have trained the environment with five different algorithms. These include Proximal

Policy Optimization(PPO) and Soft Actor Critic(SAC) which are Reinforcement Learning algorithms. We used both of these in conjunction with Generative Adversarial Imitation Learning(GAIL). Based on the results of these algorithms we propose a framework that makes use of Proximal Policy Optimization(PPO), Generative Adversarial Imitation Learning(GAIL), Behavioural Cloning(BC) and Curiosity.

4.2 GemRush

Figure 4.2 represents the training environment of GemRush application. In this game the player has to match three gems of same category. Observation space includes state of each cell in the grid and action space is handled by actuators. For each move that results in a match we have a +1 reward and for each move wasted a penalty of -1. In the application we have made an addition that includes target score upon achieving which the agent wins the game.

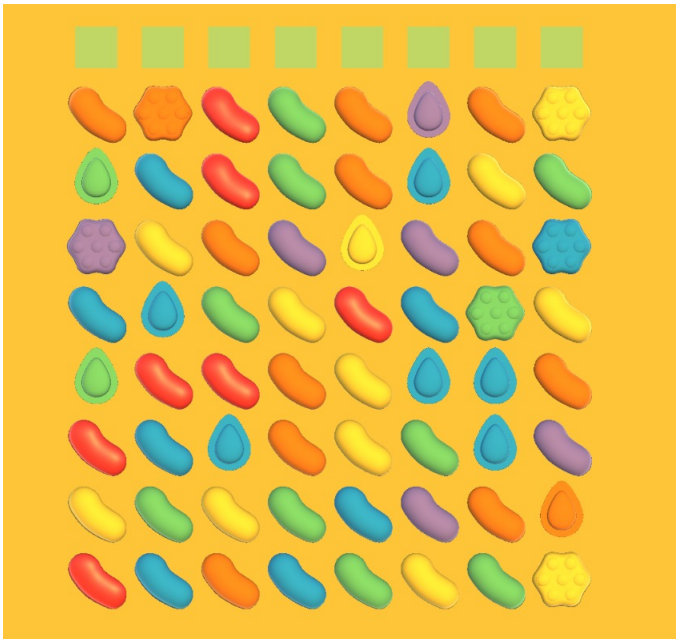

Figure 4.2: GemRush training environment.

We have trained the environment with five different algorithms. These include Proximal Policy Optimization(PPO) and Soft Actor Critic(SAC) which are Reinforcement Learning algorithms. We used both of these in conjunction with Generative Adversarial Imitation Learning(GAIL). Based on the results of these algorithms we propose a framework that makes use of Proximal Policy Optimization(PPO), Generative Adversarial Imitation Learning(GAIL), Behavioural Cloning(BC) and Curiosity.

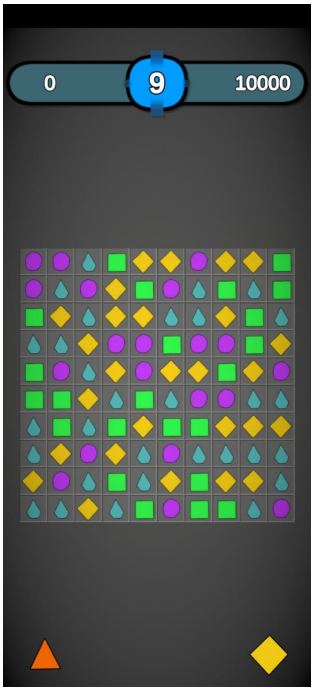

Figure 4.3: GemRush game application.

4.3 City Racer

Figure 4.4 represents the environment of City Racer application. In this game the goal of training is to navigate through the set path without any collisions. The action space includes two continuous actions which are turn and accelerate. Here we have positive rewards based on distance travelled without hitting a collider. The application has a trained agent as a rival against a player character and the goal is to race to the finishing line.

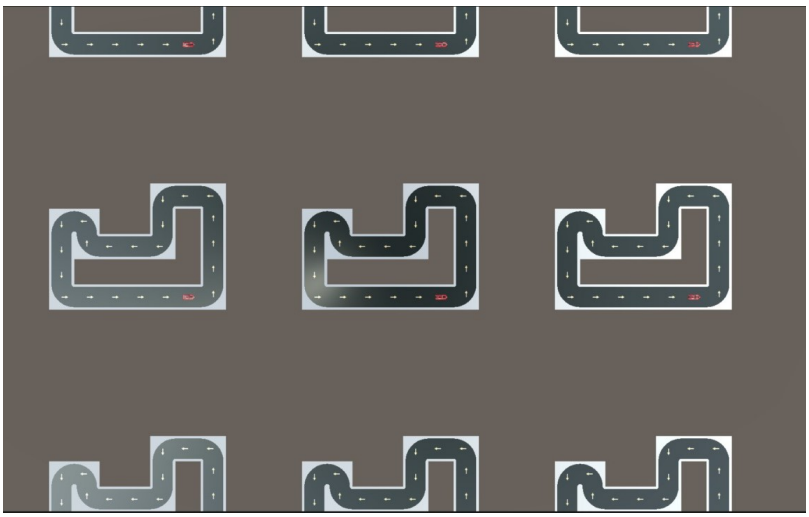

Figure 4.4: City Racer training environment.

Figure 4.5: City Racer game application.

We have trained the environment with four different algorithms. These include Proximal Policy Optimization(PPO). Based on the results of these algorithms we propose a framework that makes use of Proximal Policy Optimization(PPO), Generative Adversarial Imitation Learning(GAIL), Behavioural Cloning(BC) and Curiosity.

4.4 Fighter Mania

Figure 4.6 represents the training environment of Fighter Mania application. In this we have adversarial agents fighting against each other where the goal is to bring the opponents health to zero. The action space includes move, punch, kick and block which are all discrete actions. Reward of +1 is given to the winner and -1 to the loser when the match ends and -1 for both if there is no winner.

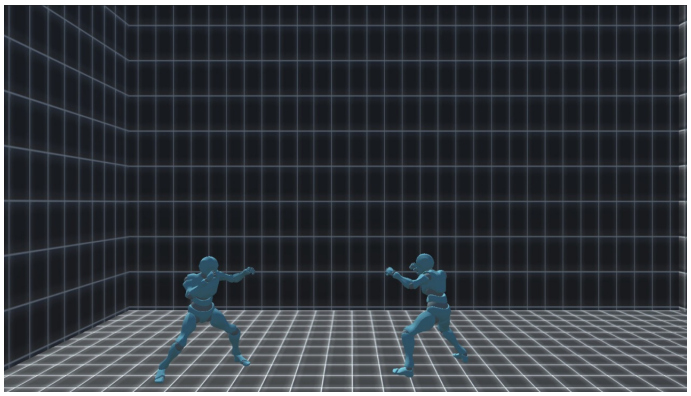

Figure 4.6: Fighter Mania training environment.

Figure 4.7: Fighter Mania game application.

We have trained the environment with four different algorithms. These include Proximal Policy Optimization(PPO) and self play mechanism . Based on the results of these algorithms we propose a framework that makes use of Proximal Policy Optimization(PPO), Generative Adversarial Imitation Learning(GAIL), Behavioural Cloning(BC), Curiosity and self play mechanism.

4.5 Proposed Methodology

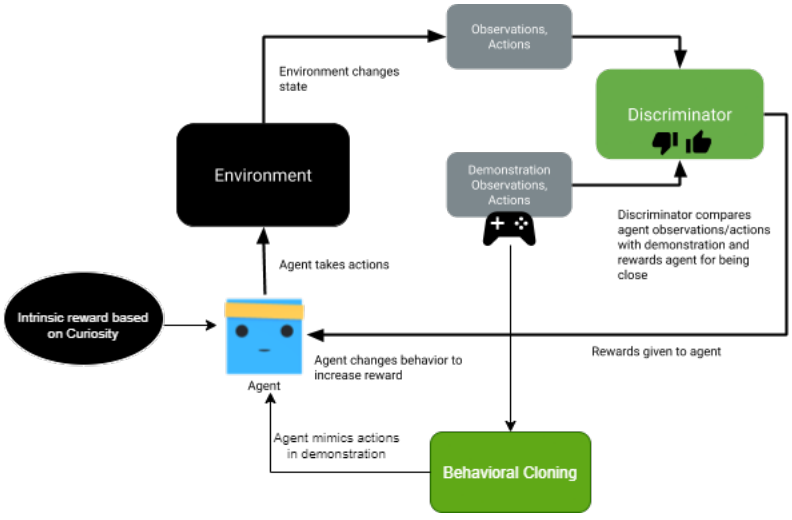

Figure 4.8: Proposed Methodology.

Chapter 5

RESULTS AND DISCUSSION

This chapter gives a brief description about the input used and the expected output of the system.

5.1 Module 1 : Cumulative Reward

The Figure 5.1 represents Cumulative reward of the different algorithms used to train the Shooter game and GemRush game environment. The maximum cumulative value doesn't meet the expectations in the PPO algorithm. GAIL SAC takes the maximum training time but has a sufficient cumulative reward value. SAC achieves best performance in terms of cumulative value only but requires high computation power and performs poorly with other algorithms. GAIL has stable performance and achieves high cumulative reward but lacks in exploration. Proposed model performs similar to GAIL but overcomes the lack of exploration by using curiosity.

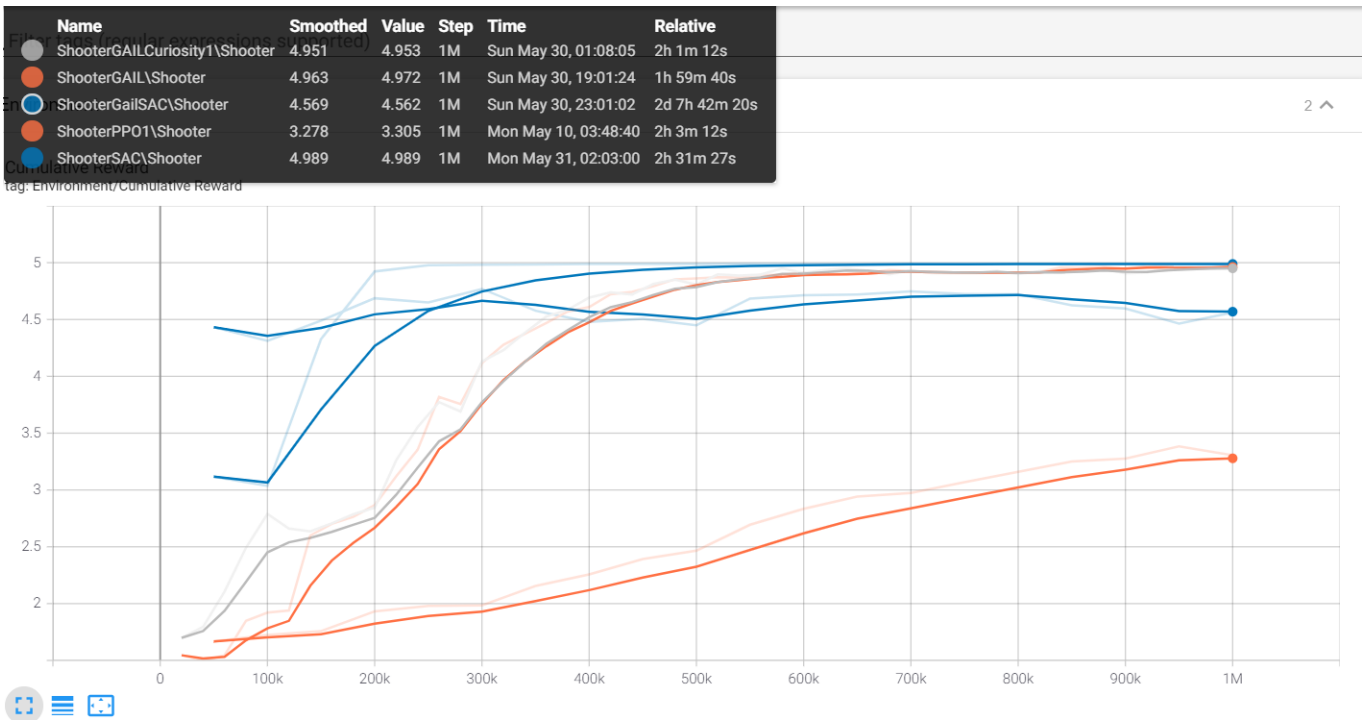

Figure 5.1: Cumulative Reward for Shooter Game.

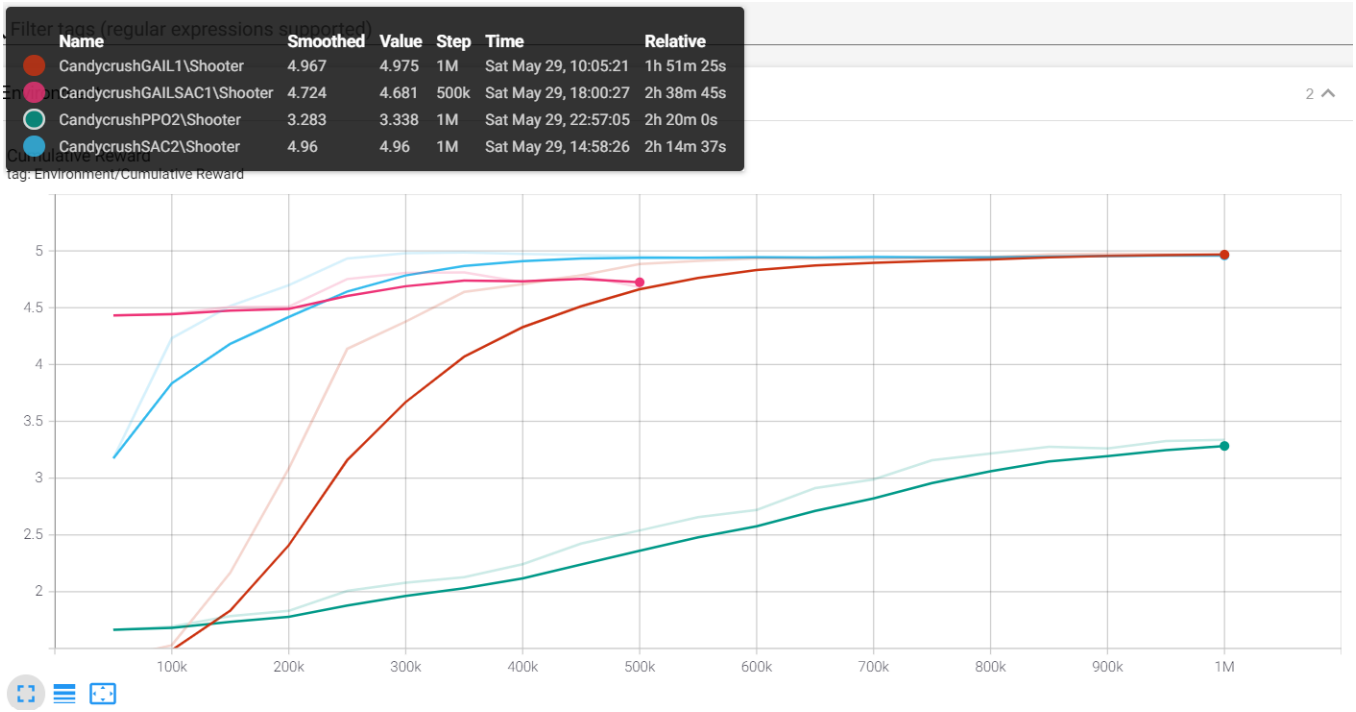

Figure 5.2: Cumulative Reward for GemRush Game.

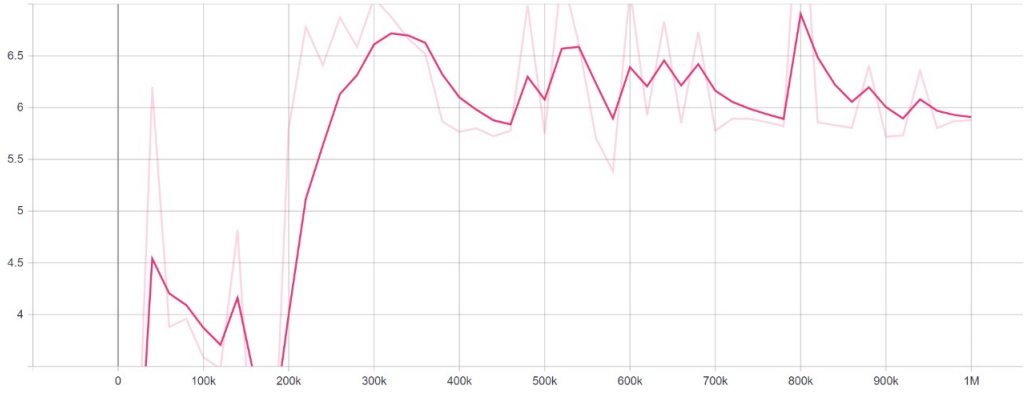

Figure 5.3: Cumulative Reward for City Racer Game.

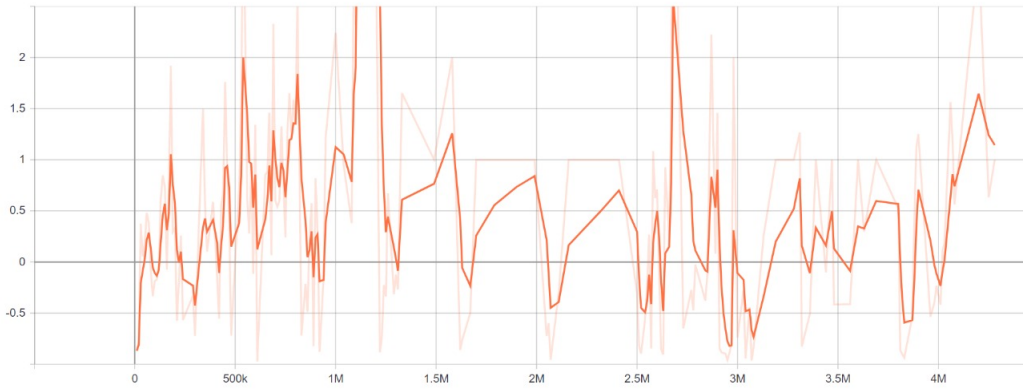

Figure 5.4: Cumulative Reward for Fighter Mania Game.

5.2 Module 2 : Episode Length

The Figure 5.1 represents Episode Length of the different algorithms used to train the Shooter game and GemRush game environment. We expect a gradual decrease in the episode length which signifies stable training. The decrease in Episode length doesn't meet the expectations in the PPO algorithm. GAIL SAC doesn't have enough change in Episode length over the course of training. SAC has a sharp decline in Episode Length which signifies instability in training. GAIL has stable performance and achieves gradual decline in Episode length but lacks in exploration. Proposed model performs similar to GAIL but overcomes the lack of exploration by using curiosity.

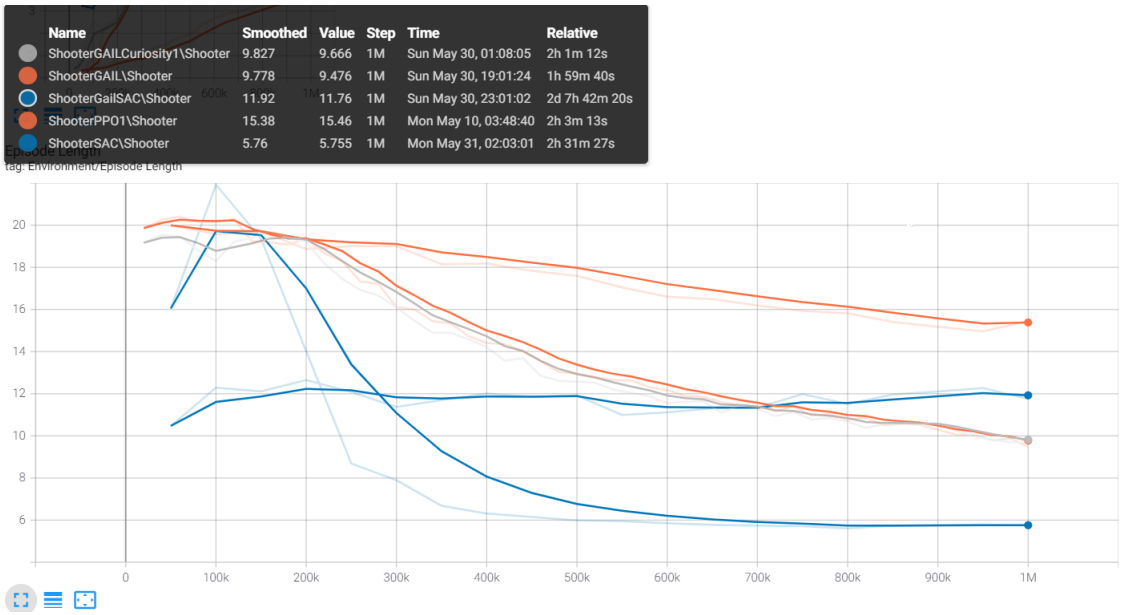

Figure 5.5: Episode Length for Shooter Game.

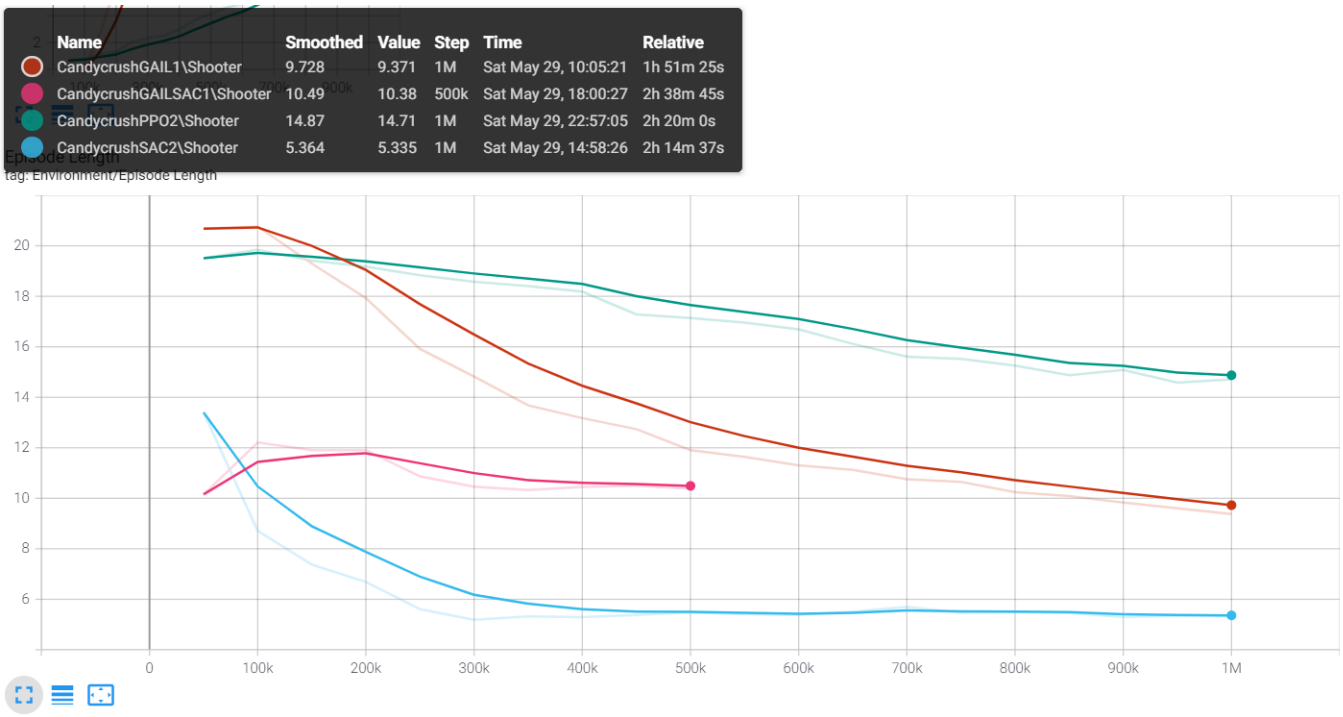

Figure 5.6: Episode Length for GemRush Game.

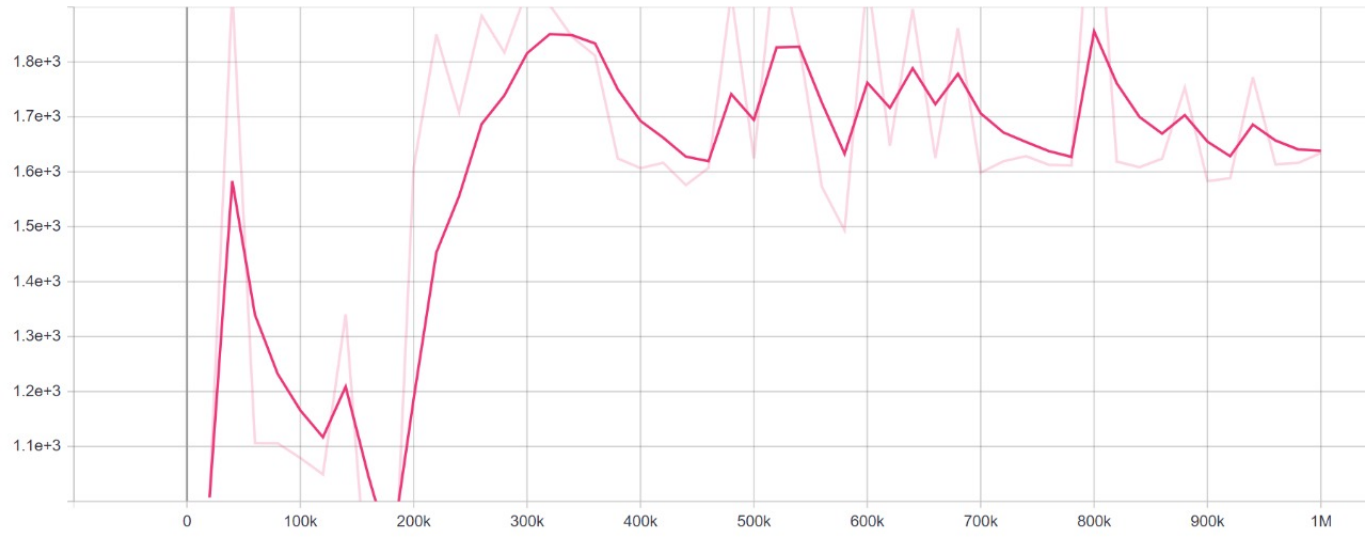

Figure 5.7: Episode Length for City Racer Game.

Chapter 6

CONCLUSION

In our proposed model we have successfully shown improvement in training criteria. Training with this model improved stability while maintaining a trade-off with exploration due to the use of curiosity. The model also reduced training time without the need of complex reward engineering or high computational requirements. The model also showcased adaptability as shown with its use in distinct environments. We also saw that various algorithms improved performance when stacked with regard to all training criteria. Based on the results compared between different algorithms we selected our proposed model and applied it on various real world game applications.

REFERENCES

- Giovanna Mart´ınez-Arellano, Richard Cant, and David Woods, "Creating AI Characters for Fighting Games using Genetic Programming",20 December 2016
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine, "Soft Actor Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", 8 August 2018
- Jonathan Ho, Stefano Ermon, "Generative Adversarial Imitation Learning", 10 June 2016
- Marek Kopel and Tomasz Hajas, "Implementing AI for non-player characters in 3D VideoGames", January 2018

Appendix A

A.1 Glossary

- **GAIL**(Generative Adversarial Imitation Learning) - GAIL is a model-free imitation learning algorithm that obtains significant performance gains over existing model-free methods in imitating complex behaviors in large, high-dimensional environments.
- **PPO**(Proximal Policy Optimization)- It is a policy gradient method for reinforcement learning.
- **Curiosity-driven learning**- It is an intrinsic reward that is equal to the error of our agent to predict the consequence of its own actions given its current state.
- **Behavioural Cloning**- Behavioral cloning learns a policy as a supervised learning problem over state-action pairs from expert trajectories.