Self-Driving JET

A SUMMER PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

MASTER OF TECHNOLOGY
IN
INTEGRATED SOFTWARE ENGINEERING

by

Aitha Tarun (18MIS7007)
Reddy Charan Kumar (18MIS7001)
Sai Eswar (18MIS7051)

Under the Guidance of

DR. Hussain Syed

SCHOOL OF COMPUTERS ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

July 2022

CERTIFICATE

This is to certify that the Summer Project work titled "**Self-Driving JET**" that is being submitted by **18MIS7007** is in partial fulfillment of the requirements for the award of Master of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Hussain Syed
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR **DEAN**
M. Tech. SE School Of Computer Science and Engineering

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Hussain Syed, Assistant Professor (Sr. Grade-1), SCOPE, VIT-AP University, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavour. My association with him in not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Web Technologies and Computer Networks.

I would like to express my gratitude to Dr. G. Viswanathan, Dr. Sekar Viswanathan, Kadhambari S. Viswanathan, Dr. S. V. Kota Reddy, and Dr. Sudha S V, SCOPE, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Reeja S R. Professor SCOPE, all teaching staff and members working as limbs of our university for their not-self-centred enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Andhra Pradesh

Date: 24th July, 2022 **Aitha Tarun**

ABSTRACT

Reinforcement learning is a machine learning algorithm supported reward desired behavior's and/or penalty behavior. In general, a reinforcement learning agent is in a position to understand and interpret its setting, take actions and learn through trial and error.

In this project we are going to develop Jet ML Agent and a snowy and mountain environment to train the Jet to self-drive by a reward system and by using proximal policy optimization trainer neural network.

Proximal Policy Optimization is a policy gradient methodology that is used for environments with either distinct or continuous action areas. It trains a random policy in an on-policy approach. Also, it utilizes the actor critic methodology. The actor maps the observation to an action and therefore the critic offers an expectation of the rewards of the agent for the observation given.

These self-driving models are employed in drones, Drones, generally said as "Unmanned Aerial Vehicles" (UAVs) are meant to hold out tasks that vary from the mundane to the ultra-dangerous. These robot-like vehicles may be found aiding the rescue of avalanche victims within the Swiss Alps, at your front doorsill dropping off your groceries and virtually all over in between. Originally developed for the military and part industries, drones have found their approach into the thought attributable to the improved levels of safety and potency they convey. These robotic UAVs operate while not a pilot on board and with totally different levels of autonomy.

TABLE OF CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

Mention self-driving vehicles, and the general public forthwith consider autonomous cars. The concept of an automobile that drives itself has captured the popular imagination. Immense efforts area unit being created to get the groundwork for the age of totally autonomous vehicles and few days goes by while not automakers, school corporations, policymakers, insurance brokers, or infrastructure corporations unveiling the most recent new development.

The idea of self-driving vehicles, or SDVs, conjointly stretches back a protracted time, 1st showing in historical records around 1478. Within the six centuries since then, the concept of autonomous vehicles has been frequently fired as absurd by some, and cherished as a dream by others. However currently the momentum for SDVs is building, following the same flight to it of spacefaring within the late 1950s. Things that after appeared not possible area unit returning reachable at astonishing speed.

Describe a vehicle as self-driving, autonomous or driverless, and completely different individuals can image terribly various things. Some might interpret it as a vehicle that travels utterly severally, while not requiring a driver in any respect. Others might imagine a vehicle that travels on its own creating autonomous selections, however that also needs somebody's driver sitting behind the wheel able to act within the event of associate emergency. The total idea of an automobile driving on its own could appear terribly advanced, however the concept behind it's quite straightforward and lies well at intervals the remit of current technology.

Consider, for a flash, the processes involved in driving a vehicle. to start with, you decide on your destination and decide a way to get there from your start line. Then you begin driving, keeping a watch on your surroundings the least bit times. These embrace static objects like buildings, trees, road signs, and place cars, as well as dynamic objects, like pedestrians, moving cars, and animals. Every now then, one amongst these objects could block the road, which is able to need you to react in a method or another. Within the in the meantime, you, because the driver, are mistreatment all the on the market mechanisms at intervals the vehicle to propel it in the required direction, whereas obeying the foundations of the road.

1.1 Objectives

- Self-Driving Flight/Jet is an agent + environment training process in which flight agent is placed in an environment to train itself such that agent maximizes the reward by passing through the checkpoints in the environment.
- Rewards will be given to the agent if it passes through a checkpoint and a small amount of penalty will be given while passing through the environment to encourage the agent to make take actions.
- Other penalties include like taking more amount of time to complete the path, or when hitting the environment objects like rocks, boundaries, or other agents.
- Also, developing a player with heuristic interface to control the player and allowing user to experience the game with other trained agents.
- Developing environment objects like rocks, checkpoints and finish point along with flight agent.
- Training the agent in the environmental with Proximal Policy Optimization (PPO) neural network architecture.

<u>Six levels of autonomous vehicles ranging from no automation to full automation</u>

- It starts with Level 0, wherever the driving force performs and monitors all aspects of the driving task and therefore the vehicle ‘s systems solely intervene to supply warnings.
- Level 1 adds driver help functions, sanctionative the system to require over either lateral (steering) or longitudinal (acceleration/deceleration) management. the motive force still decides once this could happen and is liable for perpetually watching the vehicle and manually preponderating the system where necessary. This can be the amount that the majority of nowadays ‘s vehicles have reached.
- Level 2, or Partial Automation, is that the 1st level at that the vehicle is capable of moving on its own, with the system taking on each lateral and longitudinal management in outlined use cases. Again, the motive force should perpetually monitor what's happening and be able to manually override the system in the slightest degree times.
- Level 3, or Conditional Automation, could be a vital intensify. Similarly, as absorbing each lateral and longitudinal management, the system is additionally

capable of recognizing its limits and notifying the motive force. The motive force is not any longer needed to watch the drive, however should be able to manually override among a given timeframe if the system requests it.

- Level 4 is High Automation. The system will take over all driving operations among an outlined use case, and therefore the driver has no half to play in either observation or back-up.
- Finally, Level 5, or Full Automation, sees the system take over the whole dynamic driving task altogether use cases. This entails a totally practical vehicle that's capable of driving from a place to begin to the destination with none intervention on a part of its occupants.

Automation Level	Short Name	Longitudinal/ Lateral Control	Driving Environment Monitoring	Fallback Situation Control	System Capability/ Driving Modes
0	No Automation	Human	Human	Human	None
1	Driver Assistance	Human & System	Human	Human	Some
2	Partial Automation	System	Human	Human	Some
3	Conditional Automation	System	System	Human	Some
4	High Automation	System	System	System	Some
5	Full Automation	System	System	System	All

1.2 Background and Literature Survey

The thought of a self-driving vehicle could seem sort of a strictly trendy invention, however a sketch created by sculptor Leonardo da Vinci over five hundred years agone suggests otherwise. The drawing shows a self-propelled cart supercharged by coiled springs and that includes programmable steering supported a briefing of picket pegs. In 2004, Paolo Galluzzi, director of the Institute and repository of the History of Science in Florence, oversaw a project to build an operating model supported the planning created by Leonardo around 1478. A video shows their fastidiously crafted machine in action. Often cited because the initial example of a wheeled vehicle and programmable machine, the planning might even be thought to be the world ‘s first self-driving vehicle, since it had no driver.

The idea of SDVs resurfaced in the 1939 at the New York world fair in the Futurama exhibit sponsored by the overall Motors Corporation. The installation captured the public ‘s imagination with its vision of the globe 20 years into the longer term, showing self-driving cars in operation on an automatic highway system. Sixteen years later, General Motors swollen on the themes of good roads and driverless cars in its musical short titled, "Key to the longer term." The film of an unrelentingly cheerful family enjoying the wonders of autopilot was exhibited at its 1956 Motorama auto show, that was attended by over a pair of 2 million guests in numerous locations around the America.

Fast forward another 50 years, and therefore the development of autonomous vehicles gained vital momentum with the DARPA Grand Challenge competition command by the U.S. Department of Defence in 2005, and therefore the DARPA Urban Challenge in 2007. In every case, the collaborating teams were needed to create a driverless vehicle, and complete a course among a such timeframe. This provided a serious boost to technologies such as vehicle code and AI development and marked a tipping purpose for technological progress within the field of SDVs. Since then, BMW, Audi, Daimler, Google, Tesla, Uber, Baidu, and lots of alternative corporations have continued to advance autonomous vehicle technology in varied ways.

1.3 Target Segment / Market Size

- This trained model can be used for next coming AI cars in the future.
- Also, can be used also used in autonomous drones which perform various operations like delivery and military operations.

1.4 Problem Identification

- Pilot Error: Pilots are involved at every stage of the flight, and pilot errors can occur at any one of them. Pilot error refers to an action or decision on the part of the pilot that leads to the accident. Poor training, a lack of experience, fatigue, and intoxication are all factors that can contribute to pilot error.
- Weather: Flying becomes more dangerous in bad weather. In fact, a National Transportation Safety Board study shows that more than two thirds of all weather-related aviation crashes have been fatal. Snow, fog, heavy rainstorms, and other natural elements can all make flying more difficult, which is a big reason planes often get delayed because of bad weather.

- Other Human Error and Traffic: Sabotages, terrorist attacks, unexplained disappearances, and other events have also played a role in aviation accidents over time. Land traffic causes delays and more accidents in delivery system due to congestion of vehicles and surrounding environment which can be solved by training drones to deliver goods.

1.5 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 discusses the results obtained after the project was implemented.
- Chapter 4 concludes the report.
- Chapter 5 consists of codes.
- Chapter 6 gives references.

CHAPTER 2

Self-Driving JET

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

- The trained model will be able to transport in the real world / environment by sensing the various environment objects through its sensors and take movement decisions through them.
- Jet/Plane/Agent will pass through some checkpoints to maximize its reward so that it could propagate the desired path without any collision's objects, environment boundaries or other agents.

2.2 Working Methodology

Reinforcement Learning

- Here we won't have any target label which tells the model what to perform in the environment.

- In reinforcement learning the network that transforms inputs to output actions is called policy network.

- Here policy networks are trained with policy gradients, here we start out with a completely random network and we feed that network a frame from the game engine and produces a random output action and send the action back to the game engine and game engine will produce the next frame.

- Since here we enable out agent to learn entirely by itself the only feedback, we will give the agent is the penalty/reward so, whenever the agent passes through a checkpoint it receives a rewards and penalty (negative reward) will be given when it collides with environment boundaries, other agents or environment objects such has rocks and also small penalty while propagating through the environment to encourage agent to take further actions.

- So, the entire goal of the agent is to optimize its policy to receive as much reward as possible (reward maximization).

- First thing here we do is to collect the experience by running the game multiple time through our network which selects random actions and feed them back into the engine and engine creates a whole bunch of random games (episodes).

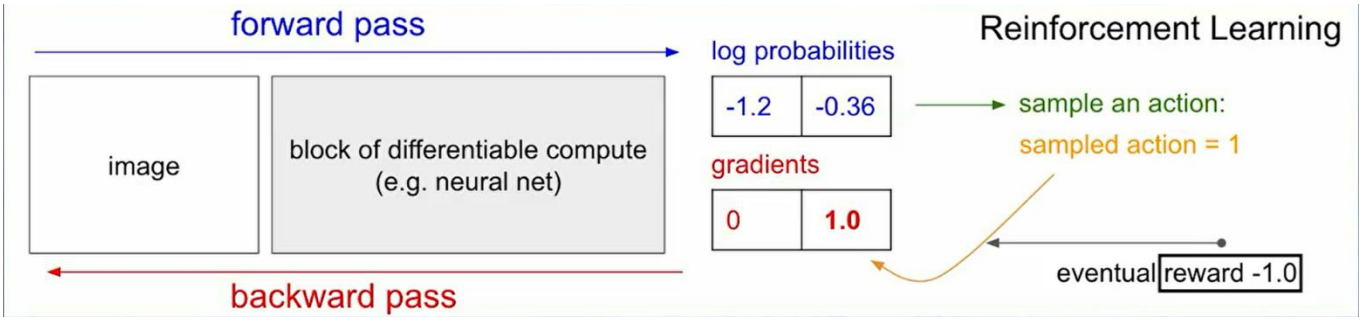
- At first, since the agent hasn't learned anything useful it will lose most of those games and gets penalty and some time the agent gets lucky and wins the game and gets rewards.

- For every episode regardless of whether we won a positive reward or negative reward we can already compute the gradients that would make the actions that our agents have chosen more likely in the future.

- Policy gradients: For every episode where we got a positive reward, we use the normal gradients to increase the probability of those actions in the

future. But when we get a negative reward, we apply the same gradient and multiply it will -1, this negative will make sure in the future all the actions that we took in a bad episode are going to b less likely in the future.

- So, results while training the policy network the actions that lead to negative rewards are slowly going to be filtered out and actions that lead to positive reward are going to become more and more likely.



Proximal Policy Optimization

- It won't use a replay buffer to store the past experiences (Deep Q-Learning) but instead it learns directly from whatever its agents encounter in the environment.

- Once a batch of experience has been used to do a gradient update the experience are then discarded and the policy moves on.

- The agent here will care more about the reward that its going to get very quickly compared to which it gets after some episodes which is called discounted sum of rewards.

- Here state will be passed to the neural network value function to get the estimation of discounted sum of reward from the respective point onwards.

- During the training the neural network which represents the value function will get frequently updated using the experience that agent collects in the environment which works like a supervised learning this is called baseline estimate.

- By subtracting the discounted rewards and baseline estimate we get advantage estimate which answer the question how much better the action which is taken based on the expectation that would happen in the state that the agent is in.

- This advantage estimate is then multiplied with $\log \pi_\theta$ which give the objective function.

- When advantage estimate is positive meaning that the actions that the agent took in sample trajectory resulted in better than the average return which leads to increasing the probability of selecting them again in the future when we encounter the same state.

- If the advantage function is negative then we will reduce the likelihood of the selected actions.

- But here we will lose the policy if we keep on updating it with the gradients.

- So, the updated policy must not move much far from the base policy.

- This done by adding a KL constraint to the optimization objective but this adds additional overhead to the optimization process which led to undesired training behaviours.

- Here PPO will directly include this extra constraint directly to the optimization objective.

2.3 System Details

This section describes the software and hardware details of the system:

2.3.1 Software Details

- Blender: A free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D-printed models, motion

graphics, interactive 3D applications, virtual reality, and, formerly, video games

- Unity game engine: used to develop video games and simulations for computers, consoles and mobile devices.

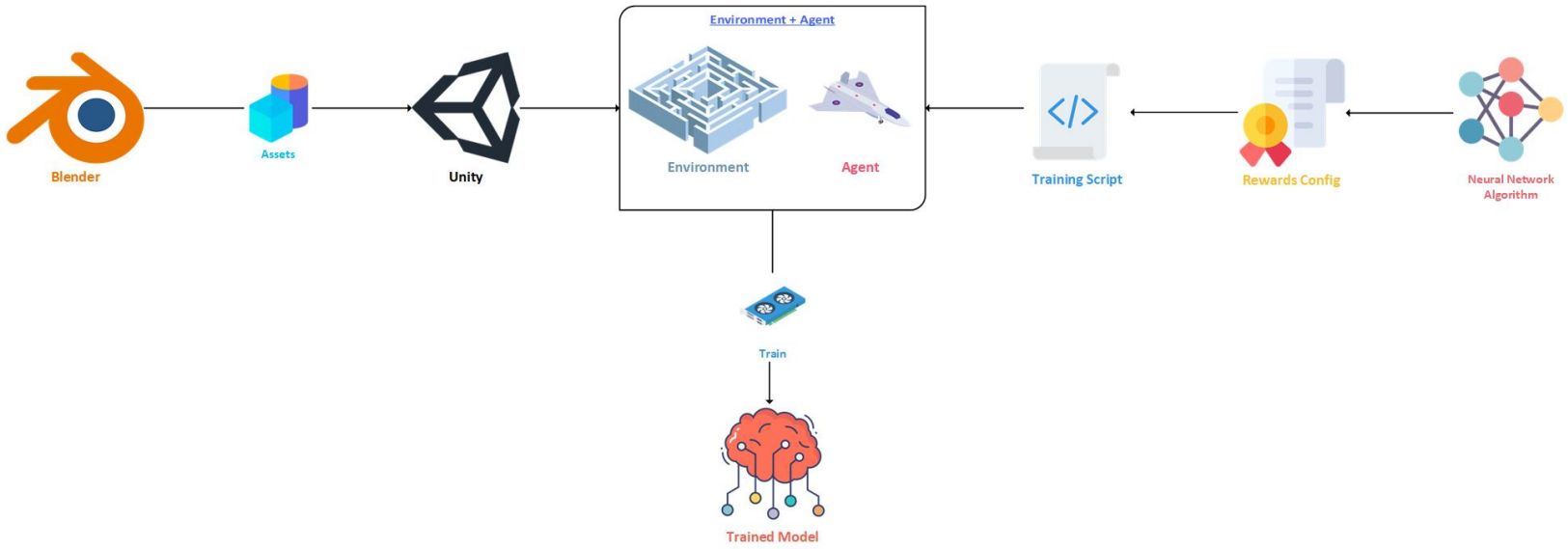
2.3.2 Hardware Details

This application requires a minimum of 8GB ram, minimum of 4 core with 2.30 base speed processor, and a GPU with minimum of 4GB memory and for training the agent in the environment.

CHAPTER 3

Diagrams

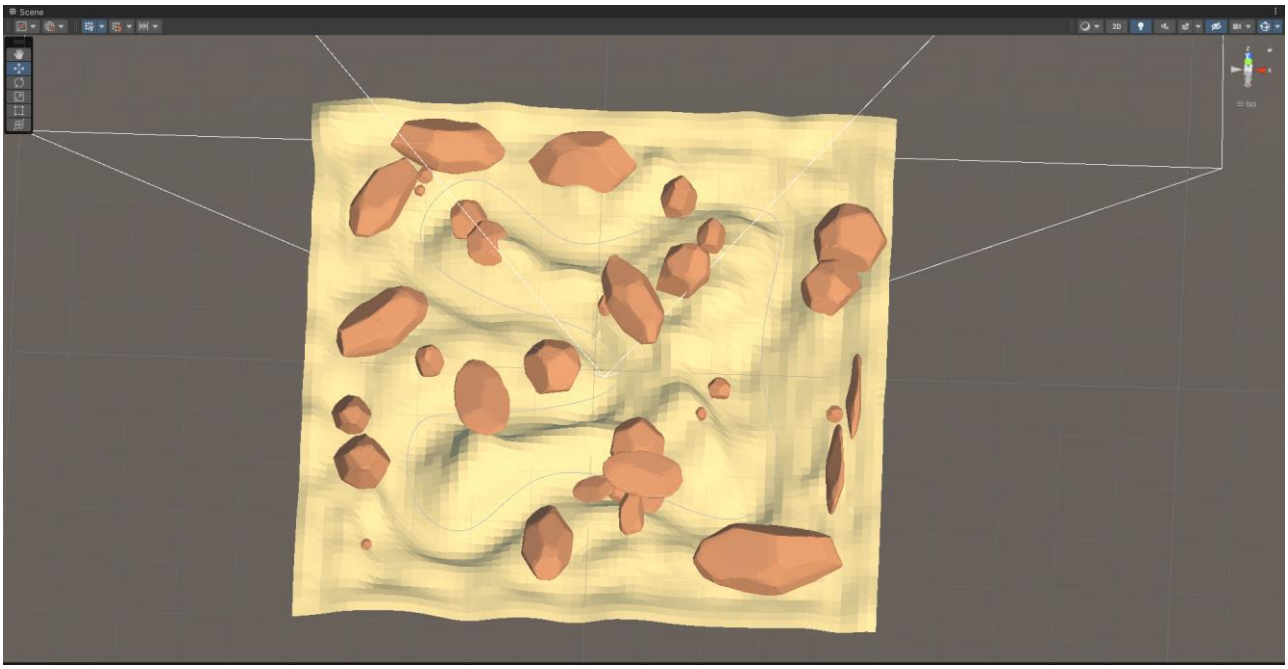
3.1 System Architecture



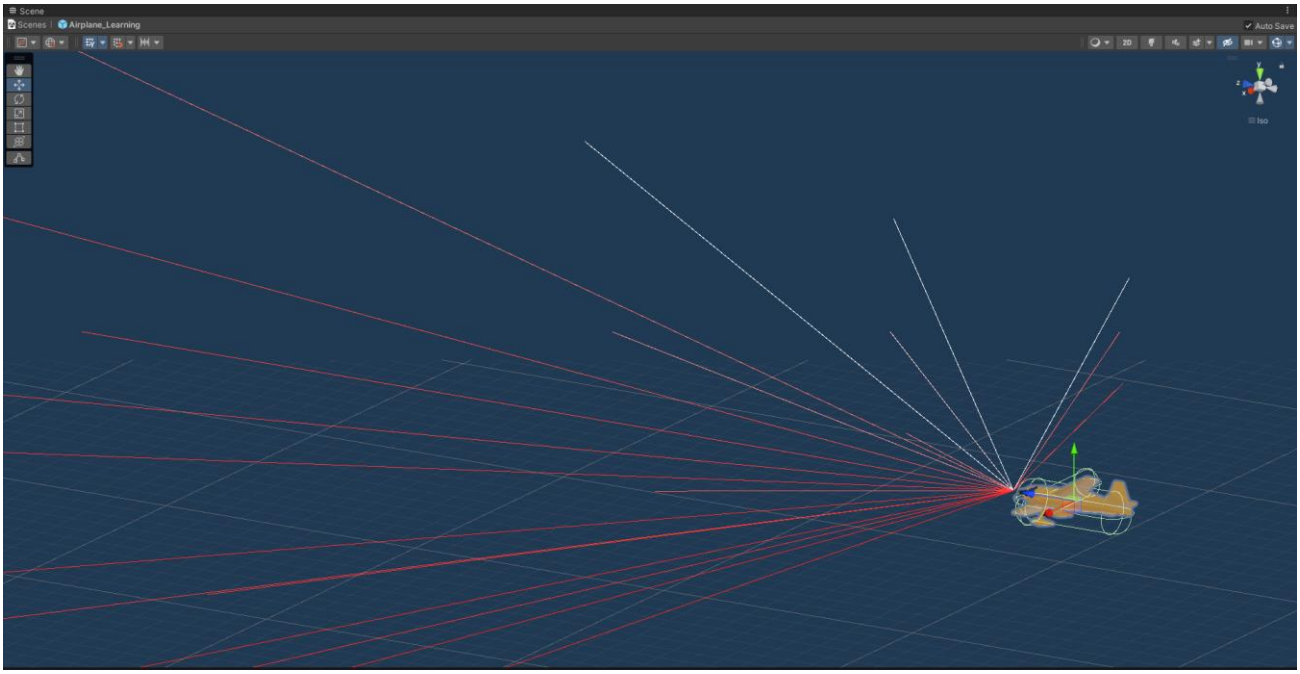
CHAPTER 4

Results

Desert Scene (Environment)

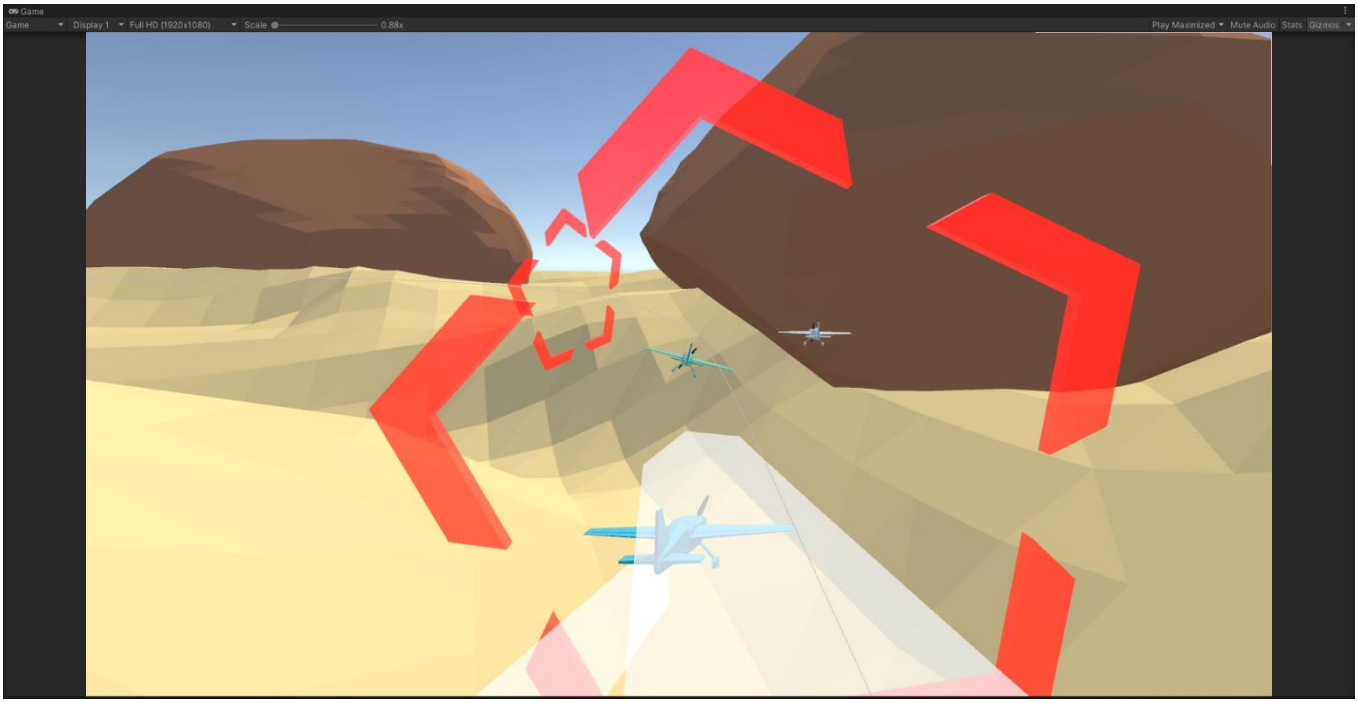


Plane/Jet (Agent)



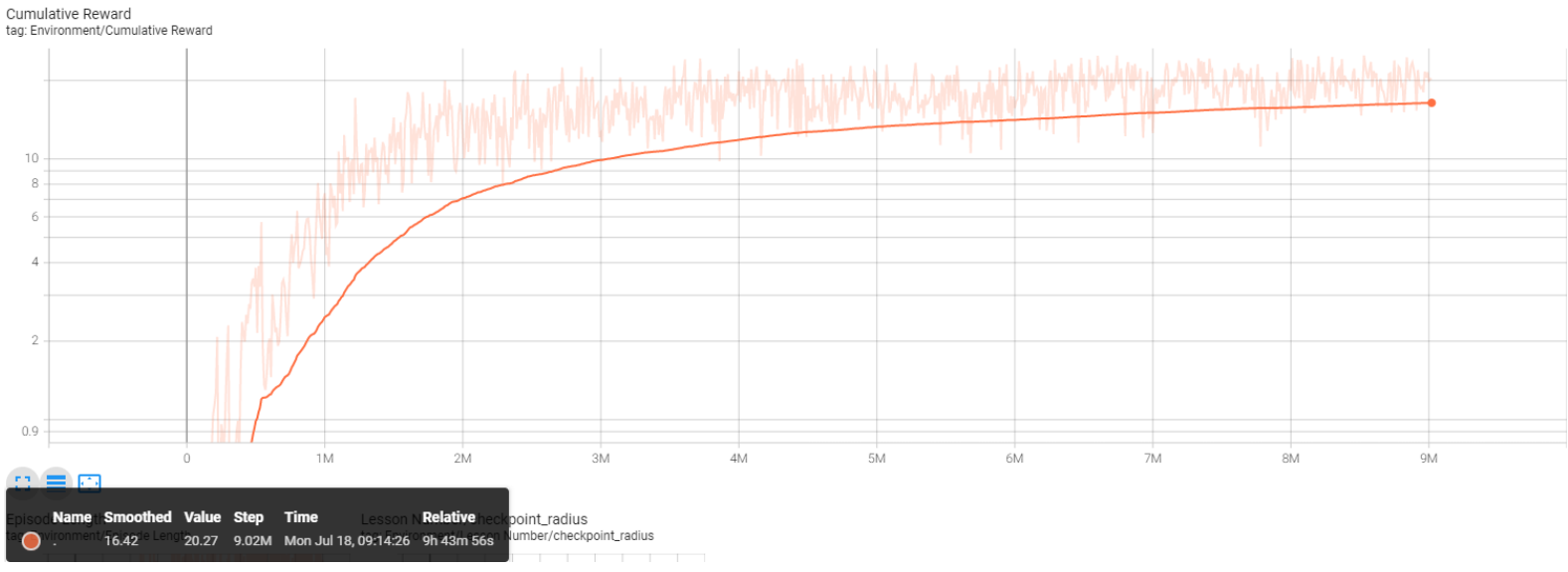
Agents in the Environment

Agents playing in the Environment

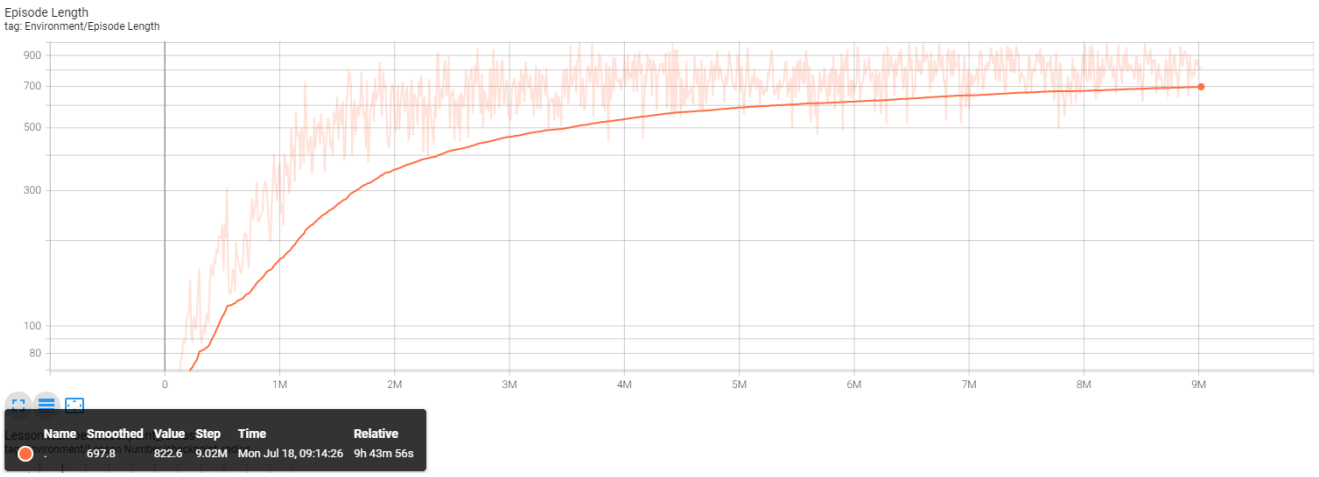


Agents when Collides with Environment

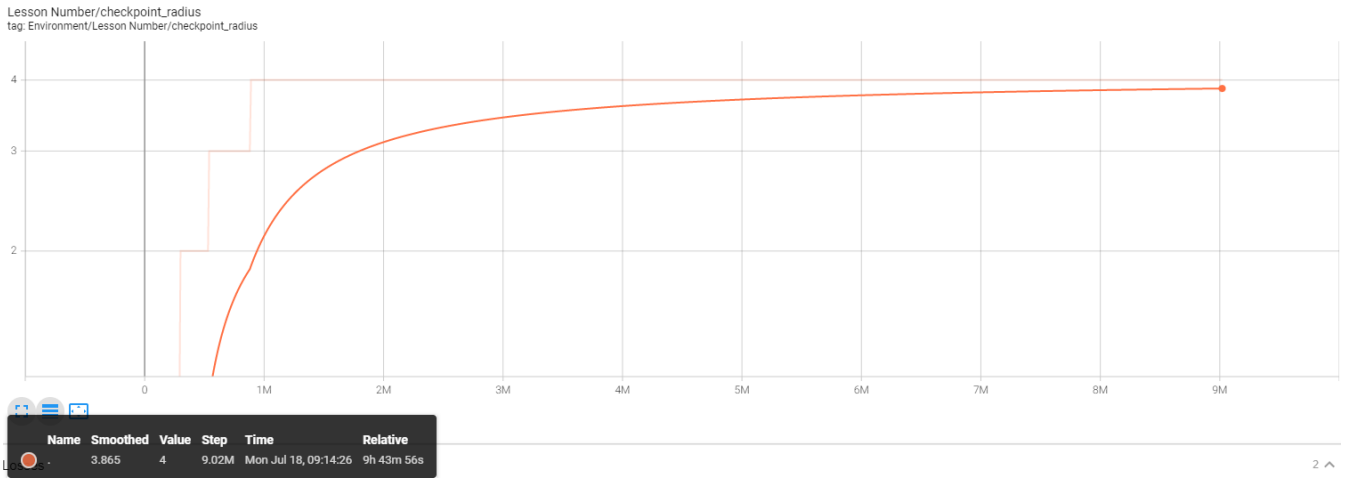
Cumulative Reward



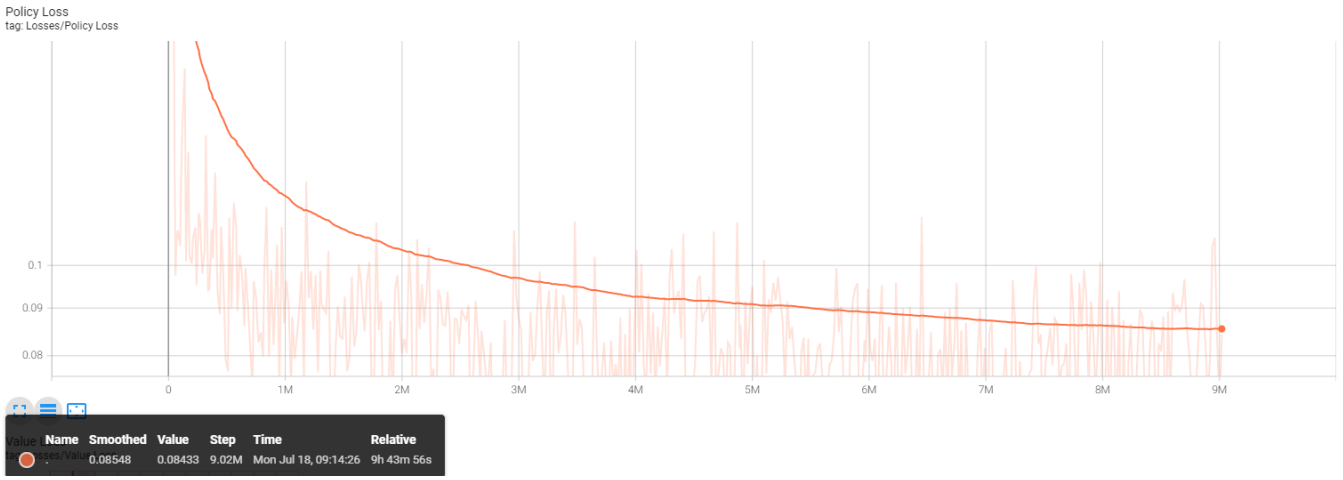
Episode Length



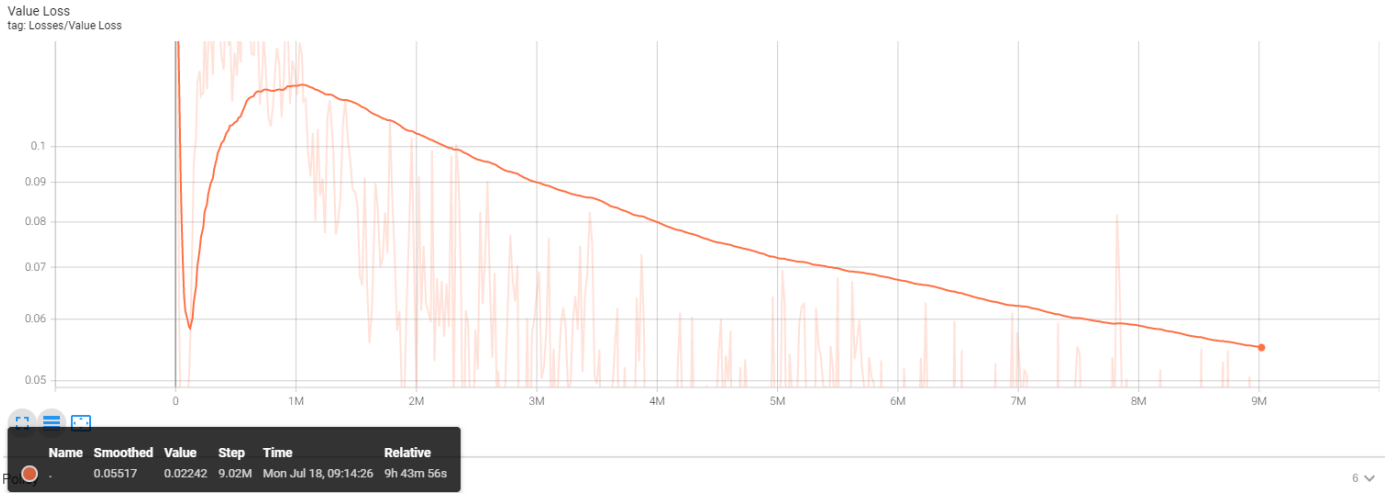
Lessons Number



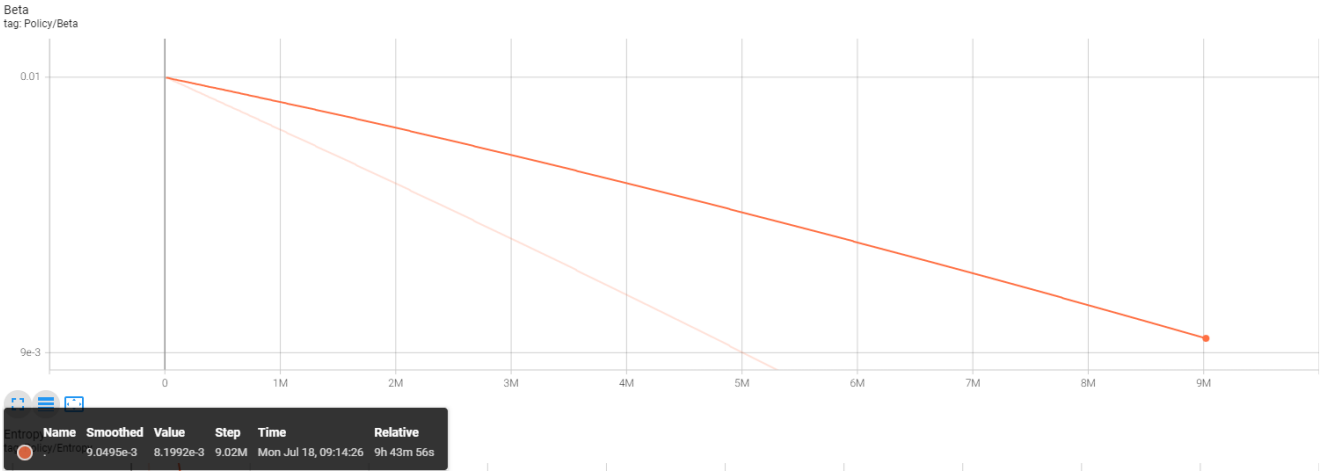
Policy Loss



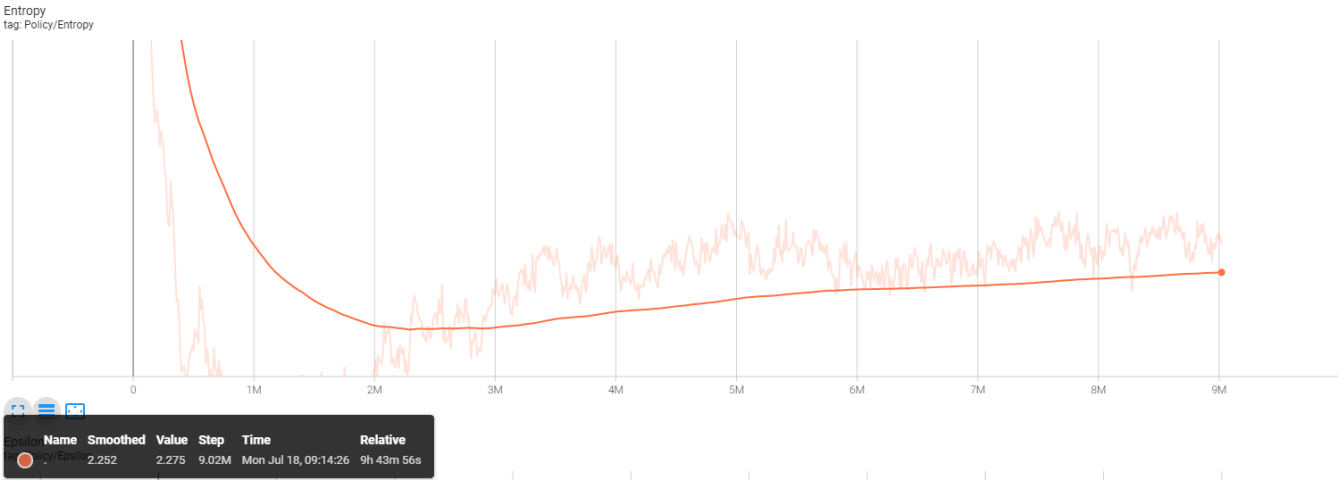
Value Loss



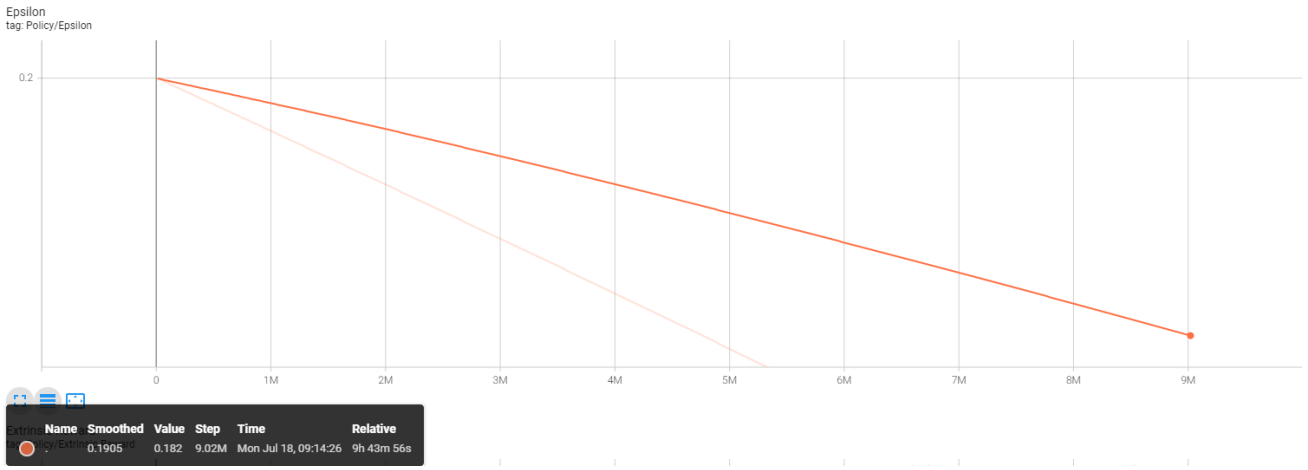
Beta



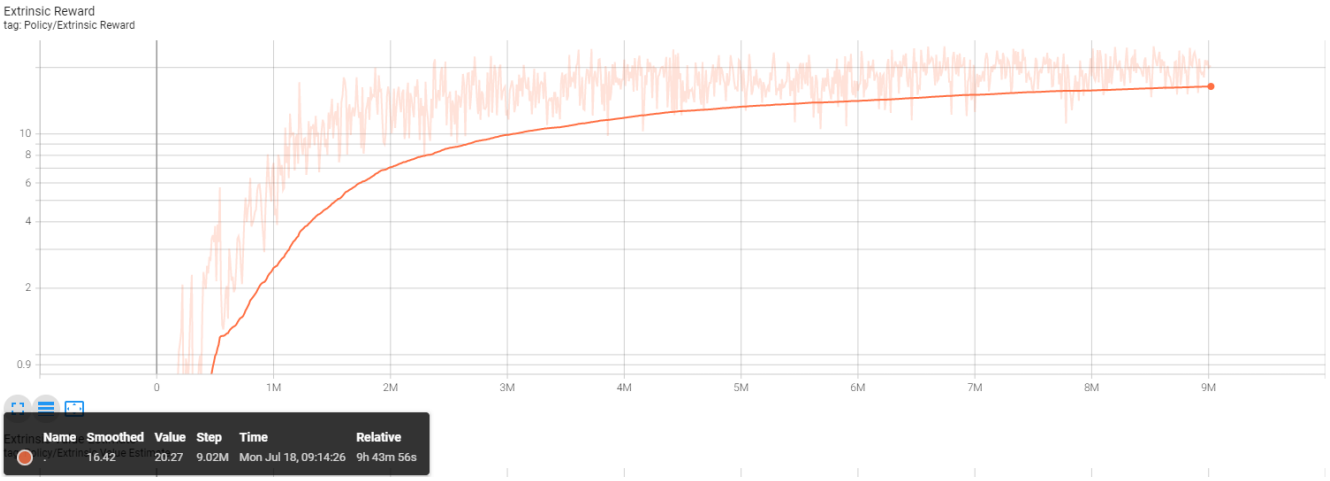
Entropy



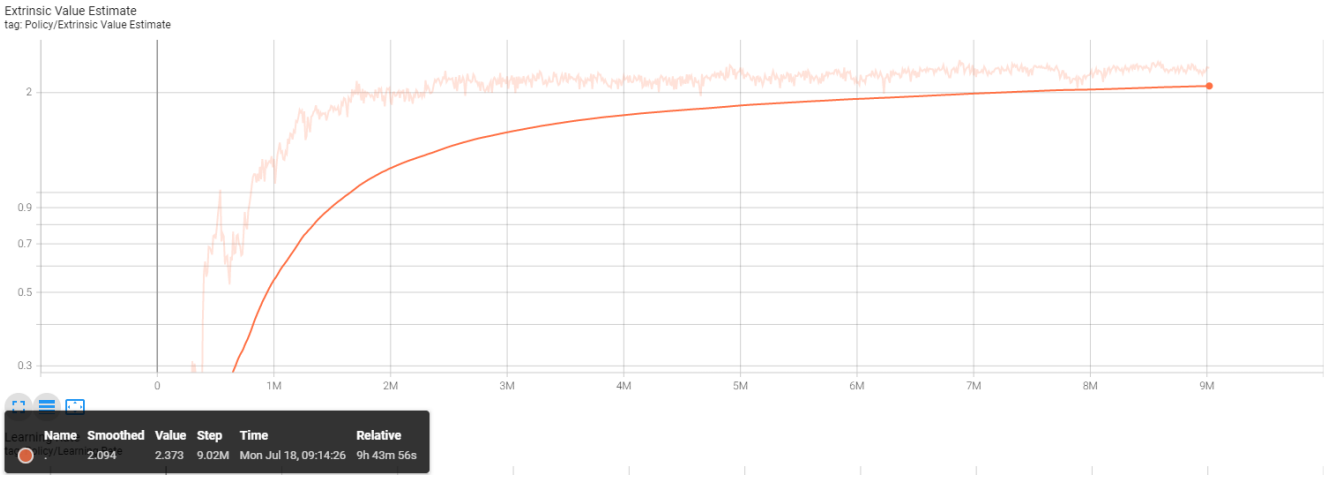
Epsilon



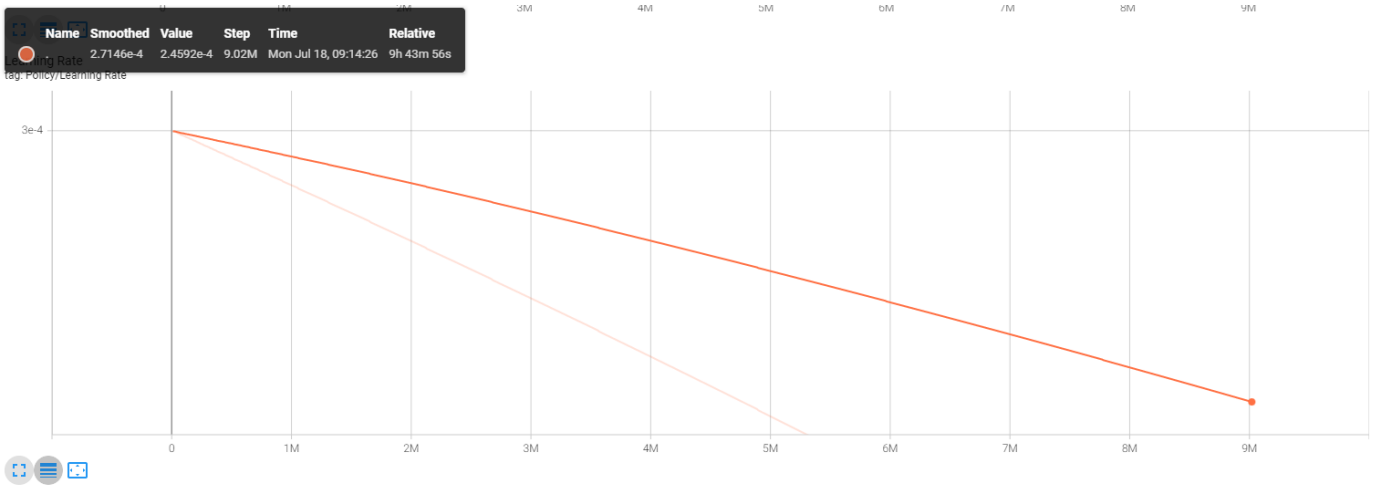
Extrinsic Reward



Extrinsic Value Estimate



Learning Rate



CHAPTER 5

Conclusion and Future Work

- The training results which are acquired are with a training time of 10 hours, with a significant performance of the agents in the environment. It takes more computational resources and time to train the agents to be more correct in the environment to maximize it rewards.

- So, this self-driving jet trained model can be used in many applications like autonomous cars, planes and drone. Which reduces the human error, accidents, and robust to any climate conditions.

- Since we can't afford training the agent in real world, we employed of creating a virtual environment. But in future we could implement this with a drone in the real environment and achieve better results.

Codes

Sample code for Aircraft Area (Environment)

```
using System;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using Cinemachine;
using Random = UnityEngine.Random;

namespace Aircraft
{
    public class AircraftArea : MonoBehaviour
    {
        [Tooltip("The path the race will take")]
        public CinemachineSmoothPath racePath;

        [Tooltip("The prefab to use for checkpoints")]
        public GameObject checkpointPrefab;

        [Tooltip("The prefab to use for start/end checkpoint")]
        public GameObject finishCheckpointPrefab;

        [Tooltip("If true, enable training mode")]
        public bool trainingMode;

        public  List<AircraftAgent> AircraftAgents { get; private set; }

        public  List<GameObject> Checkpoints { get; private set; }

        /// <summary>
        /// Actions to perform when script wakes up
        /// </summary>
        private void Awake()
        {
            if (AircraftAgents == null)
            {
                FindAircraftAgents();
            }
        }

        /// <summary>
        /// Finds Aircraft Agents in the area
        /// </summary>
        private void FindAircraftAgents()
        {
            // Find all aircraft agents in the area
            AircraftAgents = transform.GetComponentsInChildren<AircraftAgent>().ToList();

            Debug.Assert(AircraftAgents.Count > 0, "No Aircraft Agents found");
        }

        /// <summary>
        /// Set up the area
        /// </summary>
        private void Start()
        {
            if (Checkpoints == null)
            {
                CreateCheckpoints();
            }
        }
```

Sample code for Aircraft Agent (Agent)

```
using Unity.MLAgents;
using Unity.MLAgents.Actuators;
using Unity.MLAgents.Sensors;
using UnityEngine;

namespace Aircraft
{
    1 asset usage  4 usages  1 inheritor
    public class AircraftAgent : Agent
    {
        7 usages
        public int NextCheckpointIndex { get; set; }

        [Header("Movement Parameters")]
        private static float normalThrust = 100000f;
        private static float boostThrust = 100000f;
        // private static float boostThrust = 150000f;

        public float thrust = normalThrust; // To push plane forward (z-axis)  Changed in 1+ assets
        public float pitchSpeed = 100f; // How much we rotate around the x-axis  Changed in 0+ as
        public float yawSpeed = 100f; // How much we rotate around the y-axis  Changed in 0+ assets
        public float rollSpeed = 100f; // Rotation  Changed in 0+ assets
        public float boostMultiplier = 2f; // Extra force to add when airplane is boosting

        [Header("Explosion Stuff")]
        [Tooltip("The aircraft mesh hat will disappear on explosion")]
        public GameObject meshObject;  Changed in 1+ assets

        [Tooltip("The game object of the explosion particle effect")]
        public GameObject explosionEffect;  Changed in 1+ assets

        [Header("Training")]
        [Tooltip("Number of steps to time out after in training")]
        public int stepTimeout = 300;  Changed in 0+ assets

        // Components to keep track of
        private AircraftArea area;
        new private Rigidbody rigidbody; // Body of plane
        private TrailRenderer trail; // Smoke when plane is boosting

        // When the next step timeout will be during training
        private float nextStepTimout;

        // Whether the aircraft is frozen (intentionally not flying)
        private bool frozen = false;

        // Controls
        private float pitchChange = 0f;
        private float smoothPitchChange = 0f;
        private float maxPitchAngle = 45f;
        private float yawChange = 0f;
        private float smoothYawChange = 0f;
        private float rollChange = 0f;
        private float smoothRollChange = 0f;
        private float maxRollAngle = 45f;
        private bool boost = false;
```

Training Configuration

```yaml
behaviors:
  AircraftLearning:
    trainer_type: ppo
    max_steps: 5.0e7
    time_horizon: 128
    summary_freq: 10000
    hyperparameters:
      batch_size: 2048
      beta: 1.0e-2
      buffer_size: 20480
      epsilon: 0.2
      lambd: 0.95
      learning_rate: 3.0e-4
      learning_rate_schedule: linear
      num_epoch: 3
    network_settings:
      hidden_units: 256
      normalize: false
      num_layers: 2
      vis_encode_type: simple
      memory:
        memory_size: 256
        sequence_length: 64
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99
environment_parameters:
  checkpoint_radius:
    curriculum:
      - name: Lesson0
        completion_criteria:
          measure: "reward"
          behavior: AircraftLearning
          signal_smoothing: true
          min_lesson_length: 100
          threshold: 2.0
        value: 50.0
      - name: Lesson1
        completion_criteria:
          measure: "reward"
          behavior: AircraftLearning
          signal_smoothing: true
          min_lesson_length: 100
          threshold: 2.0
        value: 30.0
      - name: Lesson3
        completion_criteria:
          measure: "reward"
          behavior: AircraftLearning
          signal_smoothing: true
          min_lesson_length: 100
          threshold: 6.0
        value: 10.0
      - name: Lesson4
        value: 0.0
```

References

- https://learn.unity.com/tutorial/karting-mod-smart-karts-training-guide?uv=2019.3&projectId=5c82b27cedbc2a0e8db0c728
- https://github.com/Unity-Technologies/ml-agents/tree/main/docs
- https://arxiv.org/abs/1707.06347
- https://arxiv.org/abs/1502.05477
- Introduction to Self-Driving Vehicle Technology (Chapman & Hall/CRC Artificial Intelligence and Robotics Series)
- https://www.youtube.com/c/DapperDinoCodingTutorials/search?query=ML-Agents
- https://www.youtube.com/watch?v=8fLyMPPixdU
- https://www.youtube.com/watch?v=k_JNyLoB8vg&list=PLQMQNmwN3FvzY_J21RPTBYvJ-43uRAJKS&index=6
- https://www.youtube.com/watch?v=76WboOELXEQ
- https://www.youtube.com/watch?v=5Vy5Dxu7vDs