Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

UDC 50.10

Marat Urmanov

Development of Artificial Intelligence by using Unity Machine Learning Agents

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computing Systems and Software
(degree code: 6M070400)
Department of Computer Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Assist. Prof. PhD Madina Alimanova**

Kaskelen, 2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Marat Urmanov

Abstract

This study aims to provide a research of Unity plugin that helps to develop Machine Learning Agents within Unity engine environment. This work introduces training a single Machine Learning Agent using both Reinforcement Learning and Imitation Learning methods, comparing the results and effectiveness..

Аңдатпа

Бұл жұмыс Unity үшін плагинді зерттеу ұсынады, ол қозғалтқыш ортасының ішінде машина оқыту агенттерін дамытуға көмектеседі. Бұл жұмыс екі әдісті пайдалана отырып, машиналық оқытудың жеке агентін оқыту, Нығайту және Имитацияны оқыту, нәтижелер мен тиімділікті салыстыруды ұсынады.

Аннотация

Эта работа предоставляет исследование плагина для Unity, который помогает разрабатывать Агентов Машинного Обучения внутри среды движка Unity. Эта работа предоставляет тренировку единичного Агента Машинного Обучения, используя оба метода, Обучения с Подкреплением и Обучения Имитацией, сравнение результатов и эффективность.

Acknowledgements

Firstly, I want to thank my supervisor, Madina Alimanova, who is one of major reasons that I entered to this academic life, who encouraged me to begin this work, motivated me to keep going on throughout these two years. Without her I wouldn't have finished this thesis. Also I want to acknowledge my group mates who guided me on how to do a thesis properly, Azamat Serek and Anuar Tastembekov. I would like to thank Unity community and especially youtube channel 'Immersive Limit', which helped a lot in learning how ML agents work.

To my family

It would never be achievable where I am existing right now if it weren't for my family. Anywhere I would end up in this world any success I can achieve. There will always be in the memory of a little boy from a little city seeking knowledge and his family that helps him to get what he seeks.

Contents

Nomenclature

AI Artificial Intelligence

API Application Programming Interface

JSON JavaScript Object Notation

ML Machine Learning

NPC Non-Player Character

RL Reinforcement Learning

RNN Recurrent Neural Network

RTS Real-Time Strategy

SDK Software Development Kit

SDU Suleyman Demirel University

1. Introduction

1.1 Motivation

Since the making of computer games, the artificial intelligence development improvement issue, which would make games all the more fascinating to play, has consistently been important. Regularly artificial intelligence was sub-par compared to players in abilities, so computer games turned out to be unreasonably straightforward for them. For this situation, the game developers, as a rule, turned to different stunts to change the powers of a human and a computer rival. In cases when we are discussing a racing simulation, at that point, the alleged "catchup" word is used. The attributes of a car controlled by a computer are artificially exaggerated. Subsequently, paying little heed to how handy the player is, the computer can play with him as an equivalent. Another case - RTS genre games, where the computer is helped by additional resources. But such solutions seem to be a cheat and reject the player. That is the reason online games are so mainstream, since playing with a genuine individual is significantly more interesting. In this manner, the issue is the amazingly low or absurd degree of insight of computer rivals, which is the second rate compared to the human. With the help of Machine Learning, it will take care of this issue and significantly grow the abilities of AI in games. One of the most popular researches on this subject is the paper of DeepMind Technologies workers. Using Q-Learning [1], they figured out how to actualize a calculation fit for playing straightforward Atari 2600 computer games without knowing anything about them, aside from the pixels on the screen [[2]]. In the event that more data is given to the neural system (such as, the directions of game objects), the use of AI grows. Getting data about the condition of the game world is a genuinely basic task if the neural network is designed for a game with AI, which suggests the presence of game sources, rather than the

work with Atari games. In this work, the development and training ML agent, which is able to control character in a three-dimensional Unity environment, will be considered. The agent's tasks include maneuvering to avoid enemy traditional AI and getting to the aim area.

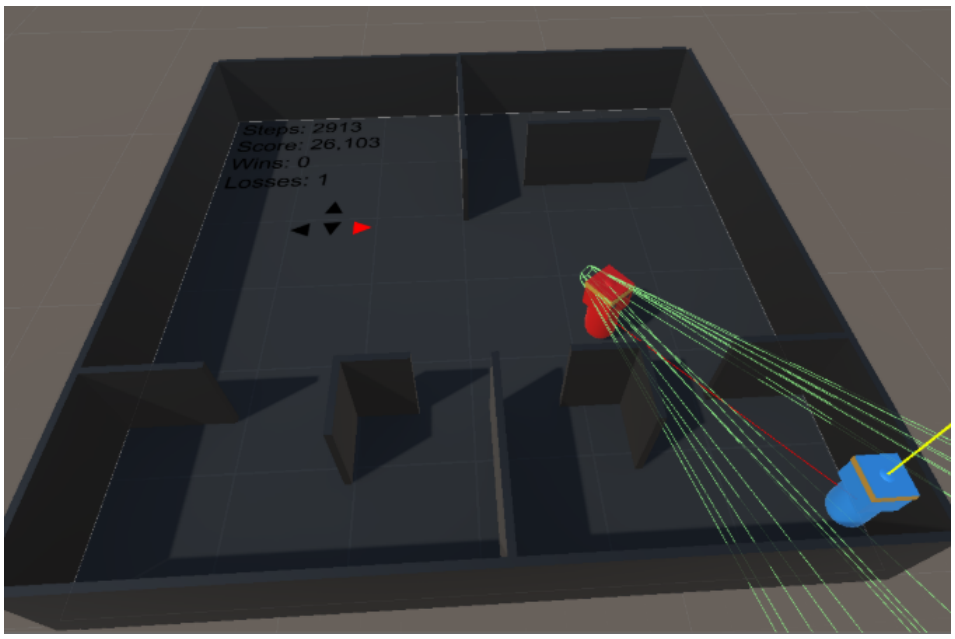

Figure 1.1: The ML Agent is in the field of view and line of sight of the AI, and is being pursued.

1.2 Aims and Objectives

The main aim of this thesis is to provide a research on a Machine learning tool developed by Unity team, define whether or not it is possible to develop a competitive Artificial Intelligence in a Unity engine driven environment. Also, to learn its features and various methods of approaching the training agents.

2. Preliminaries

The Unity game engine was chosen as a tool for the development of a learning environment. To train agents The Unity Machine Learning Agents Toolkit (Machine Learning-Agents) [[3]] was used. For learning, two methods will be used one after another: Reinforcement Learning and Imitation Learning.

2.1 The Q-Learning Algorithm

2.1.1 Contextual Bandit Recap

The objective while doing Reinforcement Learning is to prepare an agent which can figure out how to act in manners that boosts future anticipated rewards inside a given situation. In the last post in this arrangement, that condition was generally static. The condition of the earth was just which of the three potential rooms the operator was in, and the activities were picking which chest inside that space to open. Our calculation took in the Q-work for every one of these state-activity sets: Q(s, a). This Q-work compared to the normal potential compensation that would be obtained by making that move inside that state after some time. We considered this issue the "Contextual Bandit."

2.1.2 The Reinforcement Learning Problem

The absence of two things shielded that Contextual Bandit model from being an appropriate Reinforcement Learning issue: inadequate rewards, and state changes. By inadequate prizes, we allude to the way that the operator doesn't get a compensation for each move it makes. At times these prizes are "deferred," in that specific activities which may in certainty be ideal, may not give a payout until a progression of ideal moves have been made. To utilize an increasingly solid model,

an operator might be following the right way, however it will just get an award toward the finish of the way, not for each progression en route. Every one of those activities may have been basic to getting the last prize, regardless of whether they didn't give an award at that point. We need an approach to perform credit task, that is, permitting the specialist to discover that prior activities were significant, regardless of whether just in a roundabout way.

The second missing component is that in reinforcement learning issues there are advances between states. Thusly, our activities not just produce compensations as per a prize capacity: R(s, a) $\rightarrow$ r, yet in addition produce new states, as indicated by a state change work: P(s, a) $\rightarrow$ s'. A solid model here is that each progression taken while strolling along a way carries the operator to another spot in that way, thus another state. In this way we need our specialist not exclusively to figure out how to act to enhance the present conceivable prize, however act to advance toward states we know give significantly bigger prizes.

2.1.3 Bellman Updates

While these two included components of multifaceted nature may from the start appear to be random, they are in actuality straightforwardly associated. Both suggest a connection between future expresses our operator may wind up in, and potential compensations our specialist may get. We can exploit this relationship to figure out how to take ideal activities under these conditions with a basic understanding. To be specific, that under a "valid" ideal Q-work (a hypothetical one which we might ever arrive at ourselves) the estimation of a present state and activity can be deteriorated into to the quick prize r in addition to the limited greatest future expected compensation from the following state the operator will wind up in for making that move: Here (gamma) is a markdown term, which

$$Q^*(s,a) = r + \gamma max_{a'} Q^*(s',a')$$

Figure 2.1: This is called the Bellman equation

identifies with the amount we need our specialist to think about future potential prizes. On the off chance that we set gamma to 1.0, our operator would esteem all conceivable potential compensations similarly, and in preparing scenes which never end, the worth gauge may increment to interminability. Consequently, we

set gamma to an option that could be more noteworthy than 0 and under 1. Run of the mill esteems are somewhere in the range of 0.7 and 0.99.

The Bellman condition is valuable since it gives an approach to us to consider refreshing our Q-work by bootstrapping from the Q-work itself. Q*(s, an) alludes to an ideal Q-work, yet even our current, problematic Q esteem assessments of the following state can help push our evaluations of the present state an increasingly precise way. Since we are depending essentially on the genuine prizes at each progression, we can believe that the Q-esteem gauges themselves will gradually improve. We can utilize the Bellman condition to advise the accompanying new Q-learning update: This appears to be like our past relevant scoundrel update

$$Q(s,a) = Q(s,a) + \alpha(\, r + \gamma max_{a'} Q(s',a') - Q(s,a))$$

Figure 2.2: Improved Bellman equation

calculation, then again, actually our Q-target currently incorporates the limited future anticipated prize at the subsequent stage.

2.1.4 Exploration

So as to guarantee that our operator appropriately investigates the state space, we will use a type of investigation called epsilon-ravenous. To utilize epsilon-avaricious, we basically set an epsilon esteem ϵ to 1.0, and decline it just barely every time the operator makes a move. At the point when the operator picks an activity, it either picks argmax(Q(s, a)), the voracious move, or makes an irregular move with likelihood . The instinct is that toward the start of preparing our specialist's Q-esteem gauges are probably going to be extremely poor, yet as we find out about the world, and ϵ diminishes, our Q-capacity will gradually compare more to the genuine Q-capacity of the earth, and the moves we make utilizing it will be progressively exact. To exhibit a Q-learning agent, I have constructed a basic GridWorld condition utilizing Unity. The earth comprises of the accompanying: 1-a specialist set haphazardly inside the world, 2-an arbitrarily positioned objective area that we need our operator to figure out how to push toward, 3-and arbitrarily positioned hindrances that we need our operator to figure out how to stay away from. The state (s) of the earth will be a whole number which compares to the situation on the lattice. The four activities (a) will

comprise of (Up, Down, Left, and Right), and the prizes (r) will be: +1 for moving to the state with the objective, - 1 for moving to the state with a deterrent, and - 0.05 for each progression, to urge fast development to the objective with respect to the operator. Every scene will end after 100 stages, or when the specialist moves to a state with either an objective or impediment. Like in the past instructional exercise, the specialist's Q esteems will be put away utilizing a table, where the lines relate to the state, and the sections to the potential activities. You can play with this condition and operator inside your Web program here, and download the Unity undertaking to change for use in your own games here. As the operator investigates the earth, hued spheres will show up in every one of the GridWorld states. These compare to the operator's normal Q-esteem gauge for that state. When the specialist learns an ideal strategy, it will be obvious as an immediate worth inclination from the beginning situation to the objective.

2.1.5 Going Forward

The agent and condition introduced here speak to a great plain detailing of the Q-learning issue. In the event that you are feeling that maybe there isn't much in the same way as this essential condition and the ones you find in contemporary games, don't stress. In the years since the calculation's presentation during the 90s, there have been various significant improvements to permit Q-figuring out how to be utilized in progressively changed and dynamic circumstances. One prime model is DeepMind's Deep Q-Network which was utilized to figure out how to play many diverse ATARI games straightforwardly from pixels, an accomplishment inconceivable utilizing just a query table like the one here. So as to achieve this, they used a specialist which was constrained by a Deep Neural Network (DNN). By utilizing a neural system it is conceivable to gain proficiency with a summed up Q-work which can be applied to totally inconspicuous states, for example, novel mixes of pixels on a screen.

2.2 Reinforcement Learning

The principle thought of Reinforcement Learning is that the t-agent exists in a specific S-environment. Whenever the agent may process an action (or more than one action) from the arrangement of A-actions. Because of this action, the

environment changes its state and the agent gets the r-reward[[4]]. In light of this cooperation with the environment, the agent must pick the ideal solution that boosts its reward. Reinforcement Learning is particularly useful for taking care of issues related to a decision between long-term and short-term profits. It has been effectively applied in different fields, for example, robotics, media communications, lift management. Likewise, Reinforcement Learning is a decent method to create AI in games. On account of games, the game character goes about as an agent and his general surroundings and his enemies go about as an environment. Each time the character plays out an action that approximates him to win, he gets a fortifying reward. For instance, the car agent on a dashing track gets a reward after some time if the separation to the finish line is decreased. This works in the same logic otherwise - when performing ineffective actions, the agent gets a punishment. All together for the agent to perform effective actions, it is fundamental for him to get a variety of data describing the condition of the environment. The measure of this data ought to be sufficient to guarantee that the operator gets all the vital data about the environment, yet not be unreasonably huge for the agent to train all the more effectively. Additionally, it is important to normalize the data, so the estimations of the signals showing up to the agent were inside the range of [0; 1] or [-1; 1]. There are examples of the input signal for a car like speed and position on the track. A list of action signals would be resulted out of the agent's work. Just as input signals, they require normalization. For example, the input signal for a car could be [0; 1] for the gas pedal and [-1; 1] for the steering.

Figure 2.3: The Reinforcement Learning cycle

2.3 Imitation Learning

Unlike Reinforcement Learning, which processes with a reward/punishment system, Imitation Learning uses a framework dependent on the cooperation between a Teacher agent that executes the task and a Student agent that imitates the teacher. This is helpful in cases where you don't need your AI to have machine-like flawlessness, yet you need your agent to act like a real human being [**5**]. Imitation Learning Support was introduced in Machine Learning-Agents v0.3 Beta. This tool component is a ground-breaking feature that facilitates the development of a complex AI using fewer resources. All things considered, the procedure of AI development proceeds like this: there are two agents, one is a Teacher and another is a Student. Another neural network, a real person or a deterministic algorithm may go as a Teacher. The most effective outcomes are accomplished if the Teacher is a human being. Next, the learning procedure starts. The Teacher plays for some time. The planning shifts relying upon the task difficulty. For easier tasks, it takes around 4-6 minutes. For complex tasks, it is required about 2 hours. The learning is that while the Teacher agent plays, and the Student watches his activities and tries to imitate its Teacher.

There are an assortment of conceivable Imitation Learning calculations that can be utilized, and for v0.3 we are beginning with the least complex one: Behavioral Cloning. This works by gathering preparing information from an educator operator, and afterward just utilizing it to straightforwardly become familiar with a conduct, similarly that Supervised Learning for picture characterization or other customary Machine Learning undertakings work. Multi-Brain training State for instance you have a soccer match where various players, for instance hostile and cautious, should be controlled in an unexpected way. Utilizing Multi-Brain Training, you can give each "position" on the field a different mind, with its own perception and activity space, and train it nearby different cerebrums.

Toward the finish of preparing, you will get one parallel (.bytes) document, which contains one neural system model for each cerebrum. This takes into consideration blending and coordinating distinctive hyperparameters, just as utilizing our Curriculum Learning highlight to logically change how various arrangements of minds and specialists associate inside the earth after some time. On-Demand Decision-Making There are numerous types of games, for example, games, con-

stant technique games, pretending games, table games, and so on, all of which depend on specialists having the option to settle on choices after factor measures of time. We are glad to be supporting this in Machine Learning-Agents toolbox. You would now be able to empower and impair On-Demand Decision-Making for every operator freely with the snap of a catch! Basically empower it on your specialist, and make a basic capacity approach an operator to request a choice from its cerebrum. Learning under partial observability Some portion of the inspiration for changing semantics from state to perception is that, in many situations, the specialists are never really presented to the full condition of the earth. Rather, they get fractional perceptions which regularly comprise of neighborhood or fragmented data. It is frequently too costly to even think about providing the specialist with the full state, or it is indistinct how to try and speak to that state. So as to conquer this, we are including two techniques for managing fractional recognizability inside learning conditions through Memory-Enhanced Agents.

The primary memory upgrade is Observation-Stacking. This permits a specialist to monitor up to the previous ten past perceptions inside a scene, and to take care of all to the mind for dynamic. The second type of memory is the incorporation of a discretionary intermittent layer for the neural system being prepared. These Recurrent Neural Networks (RNNs) can figure out how to monitor significant data after some time in a concealed state. You can think about this as the memory of the operator.

2.4 Unity Machine Learning Agents Toolkit

2.4.1 Training Intelligent Agents

AI is evolving the manner in which we hope to get insightful conduct out of self-sufficient operators. Though earlier the conduct was programmed by a human, it is progressively instructed to the specialist (either virtual symbol or a robot) through communication in a preparation situation. This strategy is utilized to take in conduct for everything from mechanical robots, flying internet of things devices, and self-ruling cars, to characters in the games and rivals. The nature of this preparation condition is basic to the sorts of practices that can be scholarly, and there are frequently exchange offs of some sort that should be made. The

commonplace situation for preparing operators in virtual conditions is to have a solitary domain and specialist which are firmly coupled. The activities of the agent change the condition of the earth, and furnish the operator with remunerations. At Unity, we needed to structure a framework that give more prominent adaptability and usability to the developing gatherings keen on applying AI to creating clever operators. Also, we needed to do this while exploiting the top notch material science and illustrations, and straightforward yet incredible designer control gave by the Unity Engine and Editor. We believe that this mix can profit the accompanying gatherings in manners that different arrangements may not:

- Scholastic specialists keen on contemplating complex multi-operator conduct in reasonable serious and helpful situations.
- Industry scientists keen for huge scope equal preparing systems for mechanical technology, self-ruling vehicle, and other modern applications.
- Game designers keen on filling virtual universes with smart specialists each acting with dynamic and connecting with conduct.

2.4.2 Learning Environments

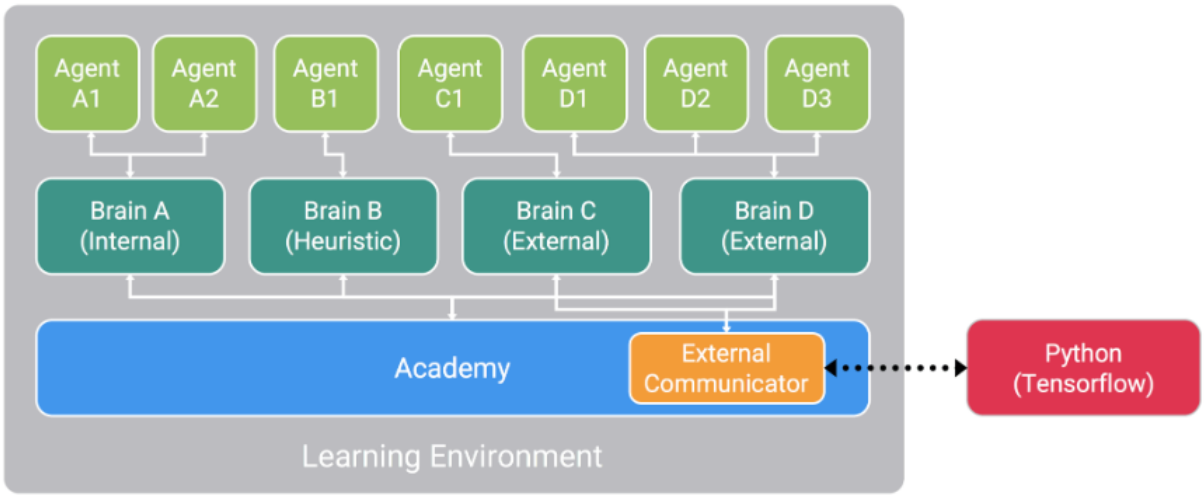

Figure 2.4: A visual representation of a possible configuration of a Learning Environment within Unity Machine Learning-Agents Toolkit.

The three fundamental sorts of items inside any Learning Environment are:

- Agent – Every Agent might have a one of a kind arrangement of states and perceptions, take one of a kind activities inside the earth, and get one of a kind compensations for occasions inside nature. An operator's activities are chosen by the mind it is connected to.

- Brain – Every Brain characterizes a particular state and activity space, and is answerable for choosing which activities every one of its connected specialists will take. The present discharge bolsters Brains can be set to one of these 4 modes:

  - External – Action choices are made utilizing TensorFlow (or your Machine Learning library of decision) through correspondence over an open attachment with our Python API.
  - Internal (Experimental) – Actions choices are made utilizing a prepared model installed into the venture by means of TensorFlowSharp.
  - Player – Action choices are made utilizing player input.
  - Heuristic – Action choices are made utilizing hand-coded conduct.

- Academy – The Academy object inside a scene likewise contains as youngsters all Brains inside the earth. Every condition contains a solitary Academy which characterizes the extent of nature, regarding:

  - Engine Configuration – The speed and rendering nature of the game engine in both preparing and derivation modes.
  - Frameskip – what number motor strides to skip between every specialist settling on another choice.
  - Global episode length – How long the scene will last. When reached, all operators are set to done.

The states and perceptions of all operators with minds set to External are gathered by the External Communicator, and imparted to our Python API for handling utilizing your Machine Learning library of decision. By setting numerous specialists to a solitary mind, activities can be chosen in a group style, opening the chance of getting the upsides of equal calculation, when upheld.

2.4.3 Flexible Training Scenarios

With Unity Machine Learning-Agents toolkit, an assortment of preparing situations are conceivable, contingent upon how specialists, minds, and rewards are associated. We are eager to perceive what sorts of novel and fun situations the network makes. For those new to preparing keen operators, underneath are a couple of models that can fill in as motivation. Each is a prototypical domain setups with a portrayal of how it very well may be made utilizing the Machine Learning-Agents SDK.

- Single-Agent – A solitary operator connected to a solitary cerebrum. The customary method of preparing an operator. A model is any single-player game, for example, Chicken. (Demo venture included – "GridWorld")

- Simultaneous Single-Agent – Multiple free agents with autonomous prize capacities connected to a solitary mind. A parallelized adaptation of the conventional preparing situation, which can accelerate and balance out the preparation procedure. A model may be preparing twelve robot-arms to each open an entryway all the while. (Demo venture included – "3DBall")

- Adversarial Self-Play – Two associating agents with reverse prize capacities connected to a solitary mind. In two-player games, ill-disposed self-play can permit a specialist to turn out to be progressively increasingly talented, while continually having the completely coordinated rival: itself. This was the procedure utilized when preparing AlphaGo, and all the more as of late utilized by OpenAI to prepare a human-beating 1v1 Dota 2 agent. (Demo venture included – "Tennis")[5]

- Cooperative Multi-Agent – Various interfacing agents with a mutual prize capacity connected to either a solitary or numerous various cerebrums. In this situation, all operators must cooperate to achieve an errand than wasn't possible alone. Models incorporate situations where every specialist just approaches fractional data, which should be partaken so as to achieve the undertaking or cooperatively understand a riddle.

- Competitive Multi-Agent – Multiple interfacing agents with converse prize capacity connected to either a solitary or various cerebrums. In this situa-

tion, operators must contend with each other to either win an opposition, or acquire some constrained arrangement of assets. All group activities would fall into this situation. (Demo venture not far off)

- Ecosystem – Multiple cooperating agents with autonomous prize capacity connected to either a solitary or numerous various minds. This situation can be thought of as making a little world in which creatures with various objectives all associate, such a savanna in which there may be zebras, elephants, and giraffes, or a self-ruling driving reproduction inside a urban domain. (Demo venture just around the corner)

2.4.4 Additional Features

Past the adaptable preparing situations made conceivable by the Academy/Brain/Agent framework, the Unity Machine Learning-Agents toolbox likewise incorporates different highlights which improve the adaptability and interpretability of the preparation procedure.

- Monitoring Agent's Decision Making - From the correspondence in Unity Machine Learning-Agents toolbox is a two-way road, we give an Agent Monitor class in Unity which can show parts of the prepared operator, for example, approach and worth yield inside the Unity condition itself. By giving these yields progressively, specialists and designers can all the more effectively troubleshoot an agent's conduct.

- Curriculum Learning - It is frequently hard for agents to become familiar with an intricate undertaking toward the start of the preparation procedure. Educational program learning is the procedure of step by step expanding the trouble of an undertaking to permit increasingly proficient learning. The Unity Machine Learning-Agents toolbox bolsters setting custom condition parameters each time the earth is reset. This permits components of nature identified with trouble or intricacy to be powerfully balanced dependent on preparing progress.

- Complex Visual Observations - In contrast to different stages, where the agent's perception may be restricted to a solitary vector or picture, the

Unity Machine Learning-Agents toolbox permits various cameras to be utilized for perceptions per specialist. This empowers specialists to figure out how to coordinate data from numerous streams being visual, as might be simply the situation when preparing a driving vehicle which required various cameras with various perspectives, a navigational operator which may need to incorporate aeronautical and first-individual visuals, or an operator which takes both a crude visual contribution, just as a profundity guide or item portioned picture.

- Imitation Learning - It is frequently progressively natural to just exhibit the conduct we need a specialist to perform, instead of endeavoring to have it learn through experimentation techniques. In a future discharge, the Unity Machine Learning-Agents toolbox will give the capacity to record all state/activity/reward data for use in administered learning situations, for example, impersonation learning. By using imitation learning, a player can give shows of how an agent ought to act in a situation, and afterward use those exhibitions to prepare an operator in either an independent style, or as an initial phase in a support learning process.

2.4.5 An Evolving Platform

As referenced above, we are eager to be discharging this open beta form of Unity Machine Learning Agents Toolkit today, which can be downloaded from our GitHub page. This discharge is just the start, and we intend to emphasize rapidly and give extra highlights to both those of you who are keen on Unity as a stage for Machine Learning research, and those of you who are centered around the capability of Machine Learning in game turn of events. While this beta discharge is progressively centered around the previous gathering, we will be progressively offering help for the last use-case. As referenced above, we are particularly keen on finding out about use-cases and highlights you might want to see remembered for future arrivals of Unity Machine Learning-Agents Toolkit, and we will be inviting Pull Requests made to the GitHub Repository.

Curriculum Learning – Our Python API presently incorporates a normalized method of using Curriculum Learning during the preparation procedure. For those new, Curriculum learning is a method of preparing an AI model where

progressively troublesome parts of an issue are step by step presented so that the model is in every case ideally tested. Here is a connect to the first paper which presents the perfect officially. All the more for the most part, this thought has been around any longer, for it is the means by which we people commonly learn. On the off chance that you envision any youth grade school training, there is a requesting of classes and subjects. Number juggling is instructed before variable based math, for instance. In like manner, polynomial math is educated before analytics. The abilities and information learned in the prior subjects give a platform to later exercises. A similar standard can be applied to AI, where preparing on simpler undertakings can give a framework to harder errands later on.

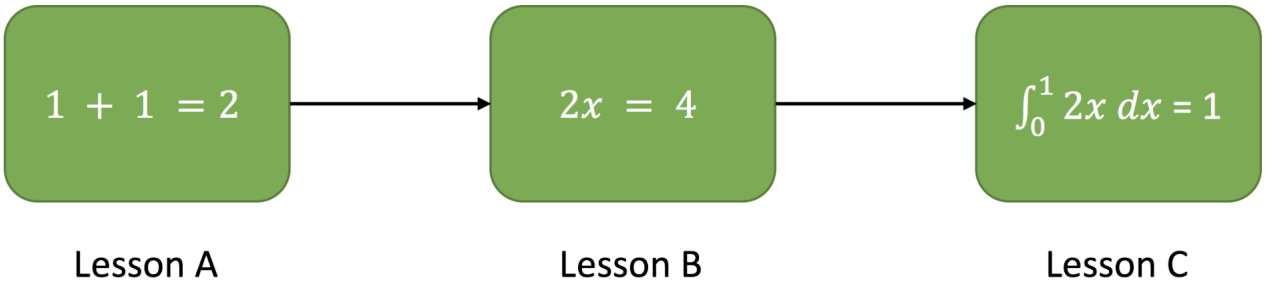

Figure 2.5: Example representing a curriculum for mathematics subject. Lessons start with easier tasks and go more difficult, with each building on the last.

At the point when we consider how Reinforcement Learning really functions, the essential learning signal is a scalar prize got every so often all through preparing. In progressively perplexing or troublesome errands, this prize can regularly be meager, and once in a while accomplished. For instance, envision an undertaking where a specialist needs to drive a square into the right spot to scale a divider and show up at an objective. The beginning stage when preparing an operator to achieve this errand will be an irregular strategy. That beginning strategy will probably include the specialist going here and there aimlessly, and will probably never, or once in a while scale the divider appropriately to get the prize. In the event that we start with a less complex undertaking, for example, advancing toward an unhampered objective, at that point the specialist can without much of a stretch figure out how to achieve the assignment. From that point, we can gradually exacerbate the assignment by expanding the size of the divider, until the operator can finish the at first close outlandish undertaking of

scaling the divider. We are including simply such a domain with Unity Machine Learning-Agents Toolkit v0.2, called Wall Area.

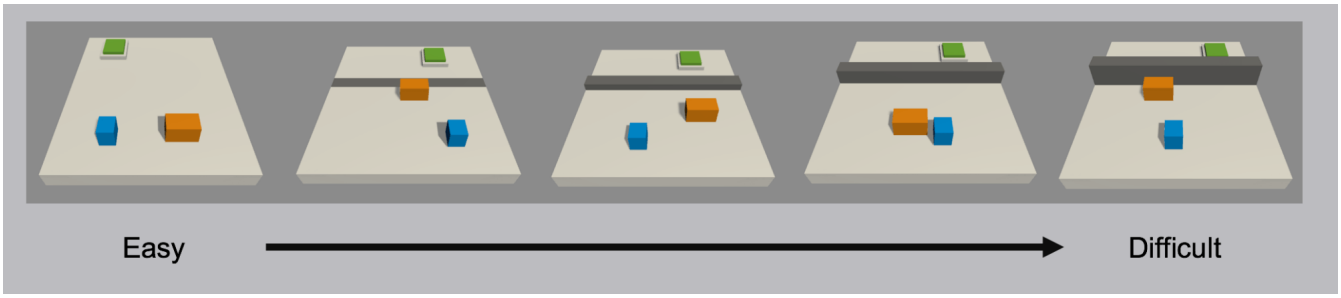

Figure 2.6: Demonstration of a curriculum training scenario in which a progressively taller wall obstructs the path to the goal.

To see this in real life, watch the two expectations to absorb information beneath. Every presentations the prize after some time for a mind prepared utilizing PPO with a similar arrangement of preparing hyperparameters and information from 32 synchronous operators. The thing that matters is that the cerebrum in orange was prepared utilizing the full-stature divider form of the undertaking, and the blue line compares to a mind prepared utilizing an educational program variant of the errand. As should be obvious, without utilizing educational plan learning the operator has a great deal of challenges, and after 3 million stages has still not fathomed the assignment. We feel that by utilizing very much made educational plans, operators prepared utilizing support learning will have the option to achieve assignments in any case substantially more troublesome, with significantly less time. So how can it work? So as to characterize an educational plan, the initial step is to choose which parameters of the earth will shift. On account of the Wall Area condition, what fluctuates is the stature of the divider. We can characterize this as a reset parameter in the Academy object of our scene, and by doing so it becomes customizable by means of the Python API. As opposed to altering it by hand, we at that point make a basic JSON record which depicts the structure of the educational program. Inside it we can set at what focuses in the preparation procedure our divider tallness will change, either dependent on the level of preparing steps which have occurred, or what the normal prize the specialist has gotten in the ongoing past is. Once these are set up, we essentially dispatch ppo.py utilizing the – educational plan document banner to highlight the JSON record, and PPO we will prepare utilizing Curriculum Learning. Obviously we would then be able to monitor the present exercise and progress by means of

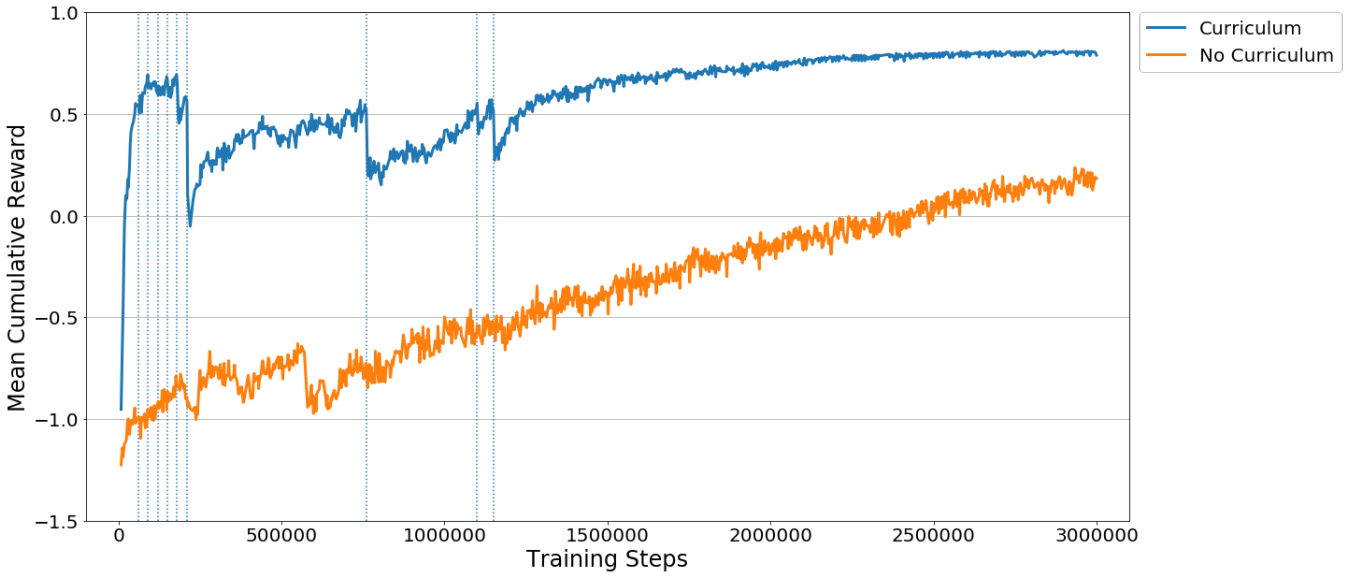

Figure 2.7: Two preparing bends on the Wall Area task. Blue line compares to mind prepared utilizing educational program learning. Orange line compares to mind prepared without educational plan learning. Spotted vertical blue lines compare to an exercise change in the educational plan for the educational program instructional course.

TensorBoard.

```
{
    "measure" : "reward",
    "thresholds" : [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5],
    "min_lesson_length" : 2,
    "signal_smoothing" : true,
    "parameters" :
    {
        "min_wall_height" : [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5],
        "max_wall_height" : [1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0]
    }
}
```

Figure 2.8: An example of a file with JSON format that defines the Curriculum.

Broadcasting – The internal, heuristic, and player brains now all incorporate a "Communicate" highlight, which is dynamic of course. At the point when dynamic, the states, activities, and awards for all operators connected to that cerebrum will at that point be available from the Python API. This is as opposed to v0.1, where just the outside brain could send data to the Python API. This element can be utilized to record, break down, or store data from these cerebrum types on Python. In particular, this component makes impersonation learning conceivable, where information from a player, heuristic, or internal brain can be utilized as the oversight sign to prepare a different system without expecting to characterize a prize capacity, or notwithstanding a prize capacity to enlarge the preparation signal. We figure this can give another road to how game designers

consider getting wise conduct from their frameworks.

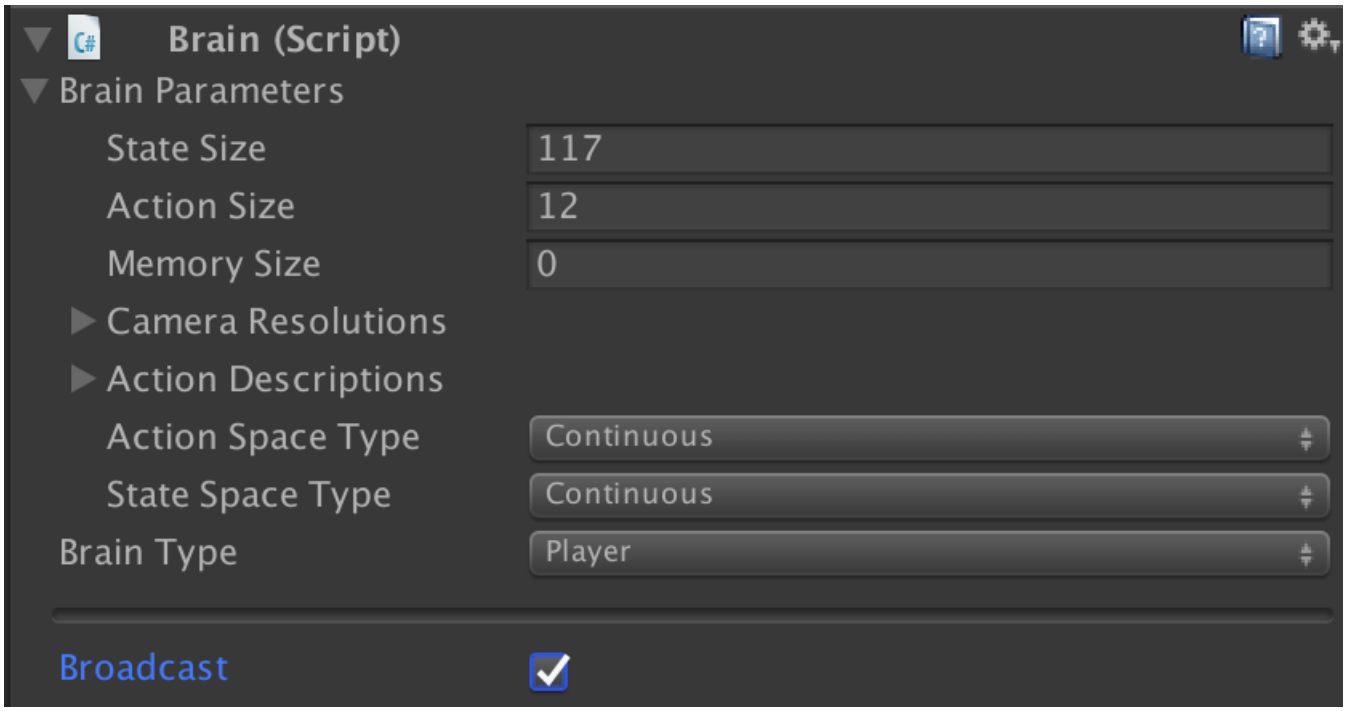

Figure 2.9: Brain Inspector window.

Flexible Monitor – We have revamped the Agent Monitor to give increasingly broad convenience. Though the first Monitor had a fixed arrangement of measurements about a specialist which could be shown, the new Monitor currently takes into account showing any ideal data identified with operators. You should simply call Monitor.Log() to show data either on the screen or over a specialist inside the scene.

2.4.6 Python Package

The gave Python bundle contains a class considered Unity Environment that can be utilized to dispatch and interface with Unity executables (just as the Editor) which contain the required parts depicted previously. Correspondence among Python and Unity takes place by means of a gRPC correspondence convention, and uses protobuf messages. We likewise give a lot of covering APIs, which can be utilized to speak with and control Unity learning conditions through the standard exercise center interface utilized by numerous analysts and calculation designers[6]. These exercise center coverings empower analysts to all the more effectively trade in Unity situations to a current support learning framework previously structured around the exercise center interface.

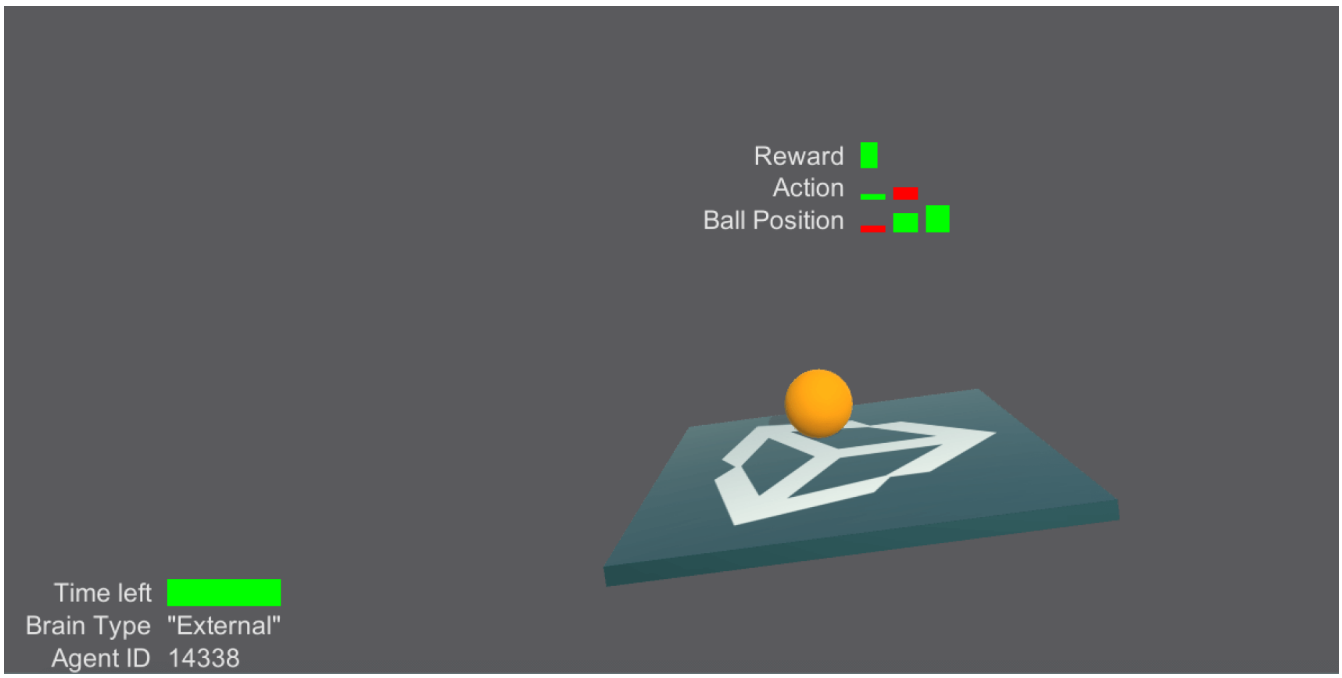

Figure 2.10: Using the Monitor.

2.4.7 Performance Metrics

It is fundamental that a situation have the option to give more prominent than continuous reenactment speed. It is conceivable to build Unity Machine Learning-Agents reproductions up to one hundred times constant. The conceivable speed increment practically speaking, be that as it may, will differ dependent on the computational assets accessible, just as the unpredictability of the earth. In the Unity Engine, game rationale, including material science, can be run freely from the rendering of edges. All things considered, situations which don't depend on visual perceptions, for example, those that utilization beam throws for model, can profit by reenactment at speeds more noteworthy than those that do.

2.4.8 Human-in-the-loop Training

Utilizing human contribution to manage the learning procedure is alluring as misusing a human's area ability accelerates learning and encourages the specialist figure out how to carry on in a way lined up with human desires. Various preparing structures have been concentrated in the writing ,for example, figuring out how to mimic master directions , people giving evaluative criticism to the specialist ,or people controlling the operator's watched states and activities. The accomplishment of the last two groups of calculations is in huge part reliant on

how the human interfaces with the operator during realizing which is exceptionally troublesome or incomprehensible with the current set of stages. Then again, impersonation learning is an altogether more standard field of research which we conjecture is mostly in light of the fact that recording master shows requires next to no additional usefulness from the stages themselves. A substitute line of work researches how to structure specialists that don't figure out how to abstain from being hindered by people given that it might keep them from accepting potential compensation [7].

Preparing inside a visual domain proofreader, for example, the Unity Editor, takes into consideration an intelligent and communitarian learning process between the human and operator. The supervisor offers constant access to the preparation scene so a human can communicate with the specialist during preparing just through mouse clicks. Potential intercessions incorporate however are not constrained to delaying the scene, hauling GameObjects inside the scene, adding or evacuating GameObjects to the scene, and in any event, expecting control of the specialist through console orders. This usefulness will make the genuine demonstration of managing criticism and changing the earth during preparing direct lifting a significant weight in this field of research.

2.4.9 Training Agents Alongside Humans

Creating games with the help of agents has a long history in the space of game structure [8]. Of specific incentive to the game advancement network is the capacity to prepare adaptable practices for non-playable characters (NPC) as either companion or then again adversary to the player. Contained inside this preparation dynamic is the under-investigated look into issue of preparing specialists to be trying to people yet not all that prevailing that the human doesn't take part in future challenge. This may not line up with a Reinforcement Learning specialist's objective of learning an ideal procedure. Preparing specialists to perform at a specific player quality has been accomplished by means of social cloning and molding the arrangement on a gauge of the expertise of the player that created the exhibition [9]. Accordingly, when a specific quality is wanted, the system can be adapted. Be that as it may, we accept there to be novel Reinforcement Learning plans which look to enhance the standard anticipated return inside a scene however additionally should advance the quantity of anticipated future scenes.

A detailing of this sort could lead to another group of Reinforcement Learning calculations and have suggestions for existential worries for AI for example, the worth arrangement issue[10].

2.5 Anatomy of Environments and Simulators

In this segment, we detail a portion of the attributes of situations and test systems we accept are expected to propel the condition of the field in AI look into. We utilize the term condition to allude to the space in which an artificial specialist acts and test system to allude to the stage which figures the earth

2.5.1 Environment Properties

As calculations can unravel progressively difficult undertakings, the intricacy of the conditions themselves must increment so as to keep on giving significant difficulties. The specific tomahawks of natural multifaceted nature we accept are basic are tangible, physical, task rationale, and social. In this subsection, we layout the job every one of these play in the cutting edge in AI.

- Sensory Complexity - The ongoing advances in profound learning have to a great extent been driven by the capacity of neural systems to process a lot of visual, sound-related, and message based information. ImageNet, a huge database of characteristic pictures with related names, was basic in empowering models, for example, ResNet, and Inception to be prepared to approach human-level item acknowledgment execution. While ImageNet was essentially utilized for static picture acknowledgment errands, its key segment of visual multifaceted nature is vital for some genuine world decisionmaking issues, for example, self-driving vehicles, family unit robots, and unmanned independent vehicles . Also, propels in PC vision calculations, specifically around convolutional neural systems, were the inspiration for the pixel-to-control approach in the end found in the Deep-Q organize.

- Physical Complexity - Huge numbers of the applied errands analysts are keen on understanding with AI include rich tactile data, however a rich control conspire in which specialists can communicate with their dynamic

surroundings in complex manners . The requirement for complex communication regularly accompanies the requirement for conditions which reproduce the physical properties of the objective area, commonly this present reality. This authenticity is fundamental to issues where the objective is to move a strategy learned inside a test system to this present reality, as would be the situation for most mechanical autonomy applications .

- Task Logic Complexity - A third pivot is the multifaceted nature of the assignments defined inside the earth. The round of Go, for instance, which has since quite a while ago filled in as a proving ground for AI investigate, contains neither complex visuals nor complex physical collaborations. Or maybe, the intricacy originates from the huge inquiry space of potential outcomes open to the specialist at some random time, and the difficulty in assessing the estimation of a given load up configuration. Important reproduction stages should empower architects to normally make such issues for the learning operators inside them. These intricate undertakings may show various leveled structure, a sign of human knowledge, or fluctuate from occasion to case, in this manner requiring meta-learning or speculation to illuminate. The undertakings may likewise be introduced in a successive way, where free examining from a fixed appropriation is beyond the realm of imagination. This is frequently the situation for human errand securing in reality, and the capacity to learn new assignments after some time is viewed as a key-part of persistent getting the hang of (Ring, 1994), and eventually frameworks equipped for artificial general insight.

- Social Complexity - The securing of complex aptitudes by means of learning in warm blooded animals is accepted to have developed connected at the hip with their capacity to hold connections inside their social gatherings. In any event one in number case of this exists inside the human species, with language principally being the advancement of an apparatus for correspondence in a social setting. In that capacity, the improvement of social conduct among gatherings of specialists is exceptionally compelling to numerous analysts in the field of AI. There are likewise classes of complex conduct which must be completed at the populace level, for example, the coordination expected to assemble present day urban areas. Also, the ca-

pacity for numerous species to interface with each other is a sign of the improvement of biological systems on the planet, and would be attractive to recreate too. A reenactment stage intended to permit the investigation of correspondence and social conduct should then give a powerful multi-specialist structure which empowers cooperation between operators of both a similar populace just as communication between gatherings of operators drawn from isolated disseminations.

2.5.2 Simulation Properties

Notwithstanding the properties above, there are down to earth requirements forced by the test system itself which must be thought about when structuring conditions for experimentation. Specifically, reenacted situations must be flexibly constrained by the analyst and must disagreement a quick and appropriated way so as to give the emphasis time required to test investigate.

- Fast Distributed Simulation - Contingent upon the example efficiency of the technique utilized, current AI calculations frequently require up to billions of tests so as to merge to an ideal arrangement[11]. Thus, the capacity to gather that information as fast as conceivable is foremost. One of the most engaging properties of a reproduction is the capacity for it to be run at a speed frequently significant degrees more noteworthy than that of the physical world. Notwithstanding this speed up, reenactments can regularly be run in equal, taking into consideration significant degrees more noteworthy information assortment than ongoing sequential involvement with the physical world. The quicker such calculations can be prepared, the more noteworthy the speed of emphasis and experimentation that can occur, prompting quicker advancement of novel techniques.

- Flexible Control - A test system should likewise permit the specialist or designer a flexible degree of command over the configuration of the reproduction itself, both during advancement and at runtime. While regarding the recreation as a black-box has been sufficient for specific advances as of late[12], by and large it likewise represses utilization of various propelled AI approaches in which progressively unique input between the preparation procedure and the specialists is basic. Educational program learning,

for instance, involves at first giving a simplified variant of an undertaking to an operator, and gradually expanding the assignment multifaceted nature as the specialist's presentation builds. This strategy was utilized to accomplish close to human-level execution in an ongoing VizDoom rivalry. Such methodologies are predicated on the presumption that the client has the ability to change the reproduction to make such educational programs in the first place. Also, space randomization includes bringing enough inconstancy into the reenactment so the models learned inside the recreation can sum up to this present reality. This frequently works by guaranteeing that the information conveyance of this present reality is secured inside the entirety of the varieties introduced inside the reproduction. This variety is particularly significant if the specialist relies upon visual properties of the earth to play out its assignment. It is frequently the situation that without space randomization, models prepared in recreation suffer from a "reality hole" and perform ineffectively. Solidly, performing area randomization frequently includes progressively controlling surfaces, lighting, material science, and item situation inside a scene.[13]

2.6 The Unity Platform

Unity is a continuous 3D improvement stage that comprises of a rendering and material science motor just as a graphical UI called the Unity Editor. Solidarity has gotten across the board selection in the gaming, AEC (Architecture, Engineering, Construction), auto, and film enterprises and is utilized by an enormous network of game designers to make an assortment of intelligent recreations, extending from little versatile and program based games to high-spending console games and AR/VR encounters.

Unity's recorded spotlight on building up a broadly useful motor to help an assortment of stages, designer experience levels, and game sorts makes the Unity motor a perfect applicant reproduction stage for AI examine. The flexibility of the hidden motor empowers the making of assignments running from basic 2D gridworld issues to complex 3D technique games, material science based riddles, or multi-specialist serious games conceivable. Not at all like a significant number of the exploration stages talked about over, the fundamental motor isn't confined

to any specific classification of interactivity or recreation, making Unity a general stage. Moreover, the Unity Editor empowers quick prototyping and improvement of games and reproduced conditions.

A Unity Project comprises of an assortment of Assets. These commonly relate to files inside the Project. Scenes are an exceptional kind of Asset which define the earth or level of a Project. Scenes contain a definition of a various leveled piece of GameObjects, which compare to the genuine items (either physical or simply sensible) inside the earth. The conduct and capacity of each GameObject is controlled by the parts joined to it. There are an assortment of inherent parts furnished with the Unity Editor, including Cameras, Meshes, Renderers, Rigid-Bodies, and numerous others. It is additionally conceivable to define custom parts utilizing C contents or outside modules.

2.6.1 Engine Properties

Environment Properties

- Sensory Complexity - The Unity engine empowers high-fidelity graphical rendering. It underpins pre-prepared just as continuous lighting and the capacity to define custom shaders, either automatically or by means of a visual scripting language. All things considered, it is conceivable to rapidly render close photorealistic symbolism to be utilized as preparing information for an AI model. It is likewise conceivable to render profundity data, object covers, infrared, or pictures with clamor infused into it using custom shaders. Besides, the motor gives a methods for defining sound signs which can fill in as potential extra observational data to learning specialists, just as beam cast based location frameworks which can reproduce Lidar.

- Physical Complexity - Physical marvels in Unity conditions can be reproduced with either the Nvidia PhysX or Havok Physics motors. This empowers inquire about in situations with mimicked unbending body, delicate body, molecule, and fluid elements just as ragdoll material science. Moreover, the extensible idea of the stage empowers the utilization of extra outsider material science motors whenever wanted. For instance, there are modules accessible for Unity which give both the Bullet and MuJoCo material science motors as options to PhysX.

- Task Logic Complexity - The Unity Engine gives a rich and flexible scripting framework by means of C. This framework empowers any type of interactivity or recreation to be defined and powerfully controlled. Notwithstanding the scripting language, the GameObject and segment framework empowers dealing with various occurrences of specialists, arrangements, and situations, making it conceivable to define complex progressive assignments, or errands which would require meta-figuring out how to understand.

- Social Complexity - The idea of the Unity scripting language and segment framework makes the presenting of multi-operator situations basic and clear. In fact, on the grounds that the stage was intended to help the improvement of multi-player computer games, various valuable deliberations are as of now gave out of the crate, for example, the Multiplayer Networking framework.

Simulation Properties

- Fast Distributed Simulation - The material science and edge rendering of the Unity motor occur nonconcurrently. In that capacity, it is conceivable to enormously speed up the basic reproduction without the need to expand the casing pace of the rendering procedure. It is additionally conceivable to run Unity recreations without rendering in the event that it isn't basic to the reenactment. In situations where rendering is attractive, for example, gaining from pixels, it is conceivable to control the edge rate and speed of game rationale. Broad control of the rendering quality likewise makes it conceivable to enormously build the edge rate when wanted. The additional capacities of the Unity motor do add extra overhead when endeavoring to mimic in a huge scope disseminated design. The memory impression of a Unity reproduction is likewise bigger than that of conditions from different stages, for example, an Atari game in the ALE.

- Flexible Control - It is conceivable to control most parts of the reenactment automatically, empowering analysts to define educational programs, ill-disposed situations, or other complex techniques for changing the learning condition during the preparation procedure. For instance, GameObjects can be restrictively made and pulverized continuously.

2.6.2 Unity Editor and Services

The Unity Editor is a graphical UI used to make the substance for 2D, 3D and AR/VR encounters. It is accessible on Windows, Mac and Linux. The Unity

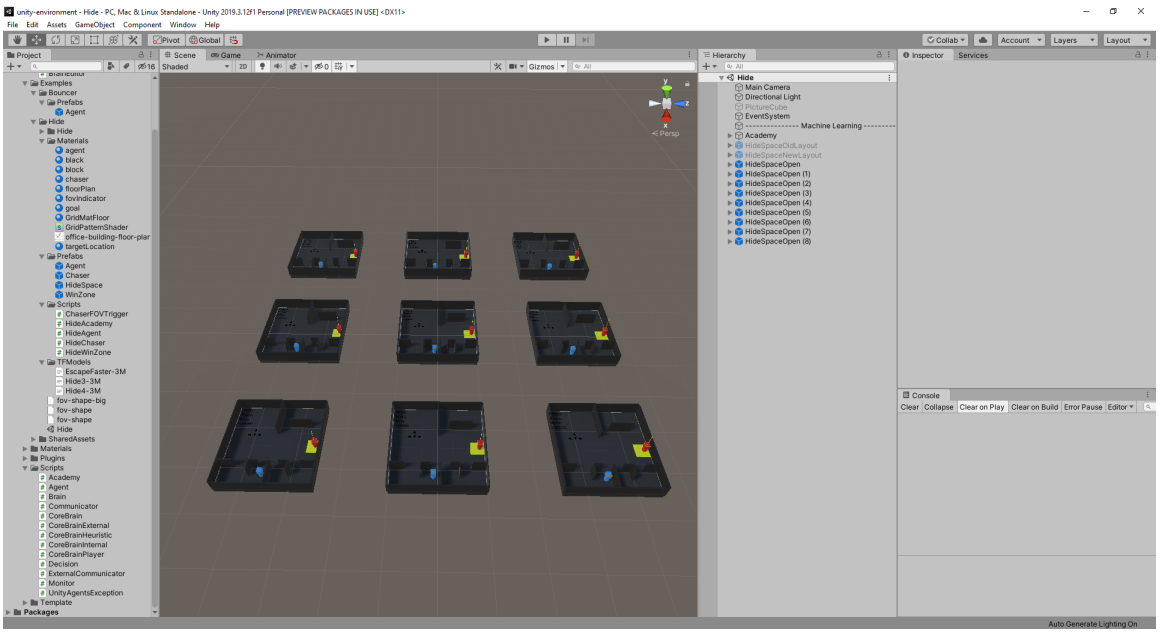

Figure 2.11: The Unity Editor window on Windows 10.

Editor and its administrations give extra benefits to AI examine:

- 1. Create custom Scenes - Unity gives countless aides and instructional exercises on the most proficient method to make Scenes inside the Editor. This empowers engineers to rapidly try different things with new situations of fluctuating complexities, or novel undertakings. Moreover, an online resource store which contains countless free and paid resources gives clients access to a gigantic decent variety of pre-manufactured substances for their scene.

- 2. Record local, expert demonstrations - The Unity Editor incorporates a Play mode which empowers a designer to start a reenactment and control at least one of the operators in the Scene by means of a console or game controller. This can be utilized for creating master exhibitions to prepare and assess Imitation learning calculations.

- 3. Record large-scale demonstrations - One the most remarkable highlights of the Unity Editor is the capacity to assemble a Scene to run on in excess

of 20 stages going from wearables to versatile and supports. This empowers engineers to appropriate their Scenes to countless gadgets (either secretly or freely through stores, for example, the Apple App Store or Google Play). This can encourage recording master showings from an enormous number of specialists or estimating human-level execution from a client (or player) populace.

3. Training agent using Reinforcement Learning

The original objective was to develop a machine learning agent that would be able to hide from a scripted Artificial Intelligence for an unlimited time and would be able to escape chasing traditional Artificial Intelligence in case being spotted.

3.1 Learning Environment

As a practical example, an area with one game object under the agent's control and belonging to the blue team and a game object under the simple, deterministic algorithm control and belonging to the red team is used. After completing training of the neural network, the blue team's game object must be able to hide from the traditional Artificial Intelligence of the red team - this will be the criterion of a successfully trained agent. As an area, a flat rectangular area that simulates the surface of a room, is used. In total, 9 areas were established, located one close to the other at a distance of 35 meters. This approach allows to significantly accelerate the training of agents due to their number and more intensive accumulation of information.

Each round is as follows: there are all the necessary game objects on the area and they are placed in random order. Then the chasing begins and continues until the traditional Artificial Intelligence catches the agent. After the chasing ends, blue agent and red Artificial Intelligence are placed randomly but not close to each other, then the round begins again. The game uses the mechanics of simple movement, with constant speed, without any accelerations or deceleration. The red Artificial Intelligence starts patrolling the area choosing randomly a point to patrol. It has a cone area that represents the red one's sight. The blue agent

always knows where is the red one is placed by sending indefinitely raycasts towards it. When the blue agent is in the sight area the red one will start chasing it until the red one will lose it out of sight. In that case the red one will continue to patrol the area.

3.2 Neural network architecture

For proper operation of the neural network, an important factor is the correct choice of its architecture and parameters. In this case, based on the existing information on the experience of using different types of neural networks for different tasks, several different configurations were selected and tested. The network parameters that showed the best results are as follows:

- Architecture: feedforward neural network
- Batch size: 512
- Batches per epoch: 8
- Number of layers: 2
- Hidden units: 128
- Max steps: 3.0e5
- Gamma: 0.99
- Lambda: 0.95
- Learning rate: 3.0e-4
- Time horizon: 2048
- Normalize: False

The feedforward neural network architecture and the tanh activation function are the default parameters that are used by the ml-agents' environment for agents training. All other parameters are configured in the file trainerconfig.yaml from the on-line ml-agents and have been selected based on experience.

3.3 Input Signals

The signal is transmitted inside the agent using the AddVec-torObs function built into ML-Agents. To achieve the best results during training, the input signals must be normalized to the range of [-1, 1] or [0, 1]. Due to the value normalization, the neural network finds a solution faster - this is a good practice in the design of neural networks. But in our case it is not necessary. In the final implementation, the blue agent has several input signals:

- Physical condition
- Current speed
- Enemy's speed
- Enemy's position
- Win zone point position (later in the Escape section)

3.4 Output Signals

The output signals of the neural network allow to control the blue agent. At the output, the neural network of the practical example has 4 signals:

- Up motion signal
- Down motion signal
- Right motion signal
- Left motion signal

Each of the signals is transferred to a secondary low-level module responsible for the control of the agent.

3.5 Rewards

For the correct work of Reinforcement Learning algorithm, it is necessary to use the reward/punishment system at training. This is a very important point; it

influences the result behavior of the neural network. Rewards are issued using the AddReward (value) method. The ML-Agents documentation states that rewards should be issued only according to the final result, not the actions, but in this practical example, the issuance of small rewards for certain actions positively affected the results. The final reward system is as follows:

- Time the agent could remain uncaught in seconds (in the Hide scenario) as a reward
- Time the agent needed to reach the win zone in seconds as a punishment (in the Escape scenario)
- 10 points as a reward if the agent managed to reach the win zone uncaught (in the Escape scenario)
- 0 points if the agent was caught (in the Escape scenario)

In the Escape scenario the reward was calculated by the next logic: 10 minus time consumed to get to the win zone. If it took more than 10 seconds then reward would be 0 as in cases the agent would be caught in the process.

3.6 Hide

Inspired by the multiple games of the stealth genre like Tom Clancy's Splinter Cell and Metal Gear Solid. The project expected the scripted Artificial Intelligence to keep watch over the environment by regularly walking or traveling around or through it and require the machine learning agent to be in a field of view and afterward an observable pathway inside that field of view. To achieve this, there was built a 3D model in blender using a modified cylinder and turned the model into a trigger area. The agent sends raycasts to the scripted Artificial Intelligence each amount of short time to look for where the scripted Artificial Intelligence's sight is directed. At the point when the agent both inside observable pathway and the field of view, the Artificial Intelligence refreshes its objective position to the position of the agent. In case the agent is out of sight direction, the last known position of the agent becomes the destination, and after reaching the destination, the scripted Artificial Intelligence will return to keep watch over the environment.

```
20      private void OnTriggerEnter(Collider other)
21      {
22          if (other.gameObject.tag == "Player")
23          {
24              if(showDebug)
25                  Debug.Log("Agent in FOV");
26              //hideChaser.RaycastToAgent();
27              hideChaser.agentInFOV = true;
28          }
29      }
30
31      private void OnTriggerStay(Collider other)
32      {
33          if (other.gameObject.tag == "Player")
34          {
35              //hideChaser.RaycastToAgent();
36              hideChaser.agentInFOV = true;
37          }
38      }
39
40      private void OnTriggerExit(Collider other)
41      {
42          if (other.gameObject.tag == "Player")
43          {
44              //hideChaser.RaycastToAgent();
45              //hideChaser.agentInLOS = false;
46              hideChaser.agentInFOV = false;
47          }
48      }
49  }
```

Figure 3.1: The red scripted AI's sight. Detection of the agent.

After testing different configurations of hyperparameters, the most successful configuration was found at a time from a values set that happened to be resulted training the agent.

General parameters:

- Max Steps : 30e5
- Run Path : "ppo"
- Load Model : False
- Train Model : True
- Summary Frequency : 10000
- Save Frequency : 50000
- Environment Name : "Hide3"

```
85      void Evasion()
86      {
87          if (evadedTimerActive == true)
88          {
89              if (evadedTimer > 0)
90              {
91                  evadedTimer -= Time.fixedDeltaTime;
92                  rewardBuffer += reward;
93              }
94              else
95              {
96                  evadedTimerActive = false;
97                  Debug.Log("Evaded in " + HideSpace.name);
98                  reward += rewardBuffer * 4.0f;
99              }
100         }
101     }
```

Figure 3.2: The evasion method is called every Fixed amount of time.

- Curriculum File : None

Algorithm-specific parameters for tuning:

- Gamma : 0.99
- Lambda : 0.95
- Time Horizon : 2048
- Beta : 1e-3
- Number of Epoch : 8
- Number of Layers : 2
- Epsilon : 0.2
- Buffer Size : 20480
- Learning Rate : 3e-4
- Hidden Units : 128
- Batch Size : 512

```
62    void MoveChaser()
63    {
64        // If Agent Has Not Been Seen
65        // Pick next room upon reaching destination
66        if (Mathf.Abs((currentTarget.transform.position - this.transform.position).magnitude) < positionDoneThreshold
67            && (chasingAgent == false))
68        {
69            //Debug.Log("Chaser Arrived At Location: " + currentTarget);
70            PickNextRoom();
71        }
72
73        // If Agent Has Been Seen
74        // Pick next room upon reaching last known location
75        if ((Mathf.Abs((lastKnownAgentLocation - this.transform.position).magnitude) < positionDoneThreshold)
76            && (chasingAgent == true) && (agentInLOS == false))
77        {
78            //Debug.Log("Chaser Arrived At LastKnownAgentLocation");
79            chasingAgent = false;
80            PickNextRoom();
81        }
82
83        // If Agent is currently seen, chase
84        if (chasingAgent == true)
85        {
86            //Debug.Log("Chase Agent");
87            ChaseAgent();
88        }
89    }
90
```

Figure 3.3: The red scripted AI patrolling and detection logic.

```
101    private void OnCollisionEnter(Collision collision)
102    {
103        if (collision.gameObject == myAgent.transform.gameObject)
104        {
105            HideAgent agent = myAgent.GetComponent<HideAgent>();
106            agent.done = true;
107            agent.reward -= 10f;
108            agent.losses += 1;
109        }
110    }
```

Figure 3.4: The punishment for the agent in case it is caught by scripted red AI.

- normalize : False

The agent's behaviour and the widely differing results of the training was unsatisfactory, so the work needed to be checked up with documentation of best practices on the machine learning agents git repository. And it was decided to increase the buffer size. The attempt resulted in a more stable, but still not very magnificent smooth increment in cumulative reward. In the work it was decided to continue to increase the buffer_size hyperparameter and train the agent until stability or behaviour enhancement occurs.

General parameters:

- Max Steps : 30e5
- Run Path : "ppo"

```
### General parameters
max_steps = 30e5 # Set maximum number of steps to run environment.
run_path = "ppo" # The sub-directory name for model and summary statistics
load_model = False # Whether to load a saved model.
train_model = True # Whether to train the model.
summary_freq = 10000 # Frequency at which to save training statistics.
save_freq = 50000 # Frequency at which to save model.
env_name = "Hide3" # Name of the training environment file.
#curriculum_file = "curricula/hide.json"
#curriculum_file = "curricula/escape.json"
curriculum_file = None

### Algorithm-specific parameters for tuning
gamma = 0.99 # Reward discount rate.
lambd = 0.95 # Lambda parameter for GAE.
time_horizon = 2048 # How many steps to collect per agent before adding to buffer.
beta = 1e-3 # Strength of entropy regularization
num_epoch = 8 # Number of gradient descent steps per batch of experiences.
num_layers = 2 # Number of hidden layers between state/observation encoding and value/policy layers.
epsilon = 0.2 # Acceptable threshold around ratio of old and new policy probabilities.
buffer_size = 20480 # How large the experience buffer should be before gradient descent.
learning_rate = 3e-4 # Model learning rate.
hidden_units = 128 # Number of units in hidden layer.
batch_size = 512 # How many experiences per gradient descent update step.
normalize = False

### Logging dictionary for hyperparameters
hyperparameter_dict = {'max_steps':max_steps, 'run_path':run_path, 'env_name':env_name,
    'curriculum_file':curriculum_file, 'gamma':gamma, 'lambd':lambd, 'time_horizon':time_horizon,
    'beta':beta, 'num_epoch':num_epoch, 'epsilon':epsilon, 'buffe_size':buffer_size,
    'leaning_rate':learning_rate, 'hidden_units':hidden_units, 'batch_size':batch_size}
```

Figure 3.5: Hyperparameters for training the Hide Scenario (Hide3).

- Load Model : False
- Train Model : True
- Summary Frequency : 10000
- Save Frequency : 50000
- Environment Name : "Hide4"
- Curriculum File : None

Algorithm-specific parameters for tuning:

- Gamma : 0.99
- Lambda : 0.95
- Time Horizon : 2048
- Beta : 1e-3
- Number of Epoch : 8
- Number of Layers : 2

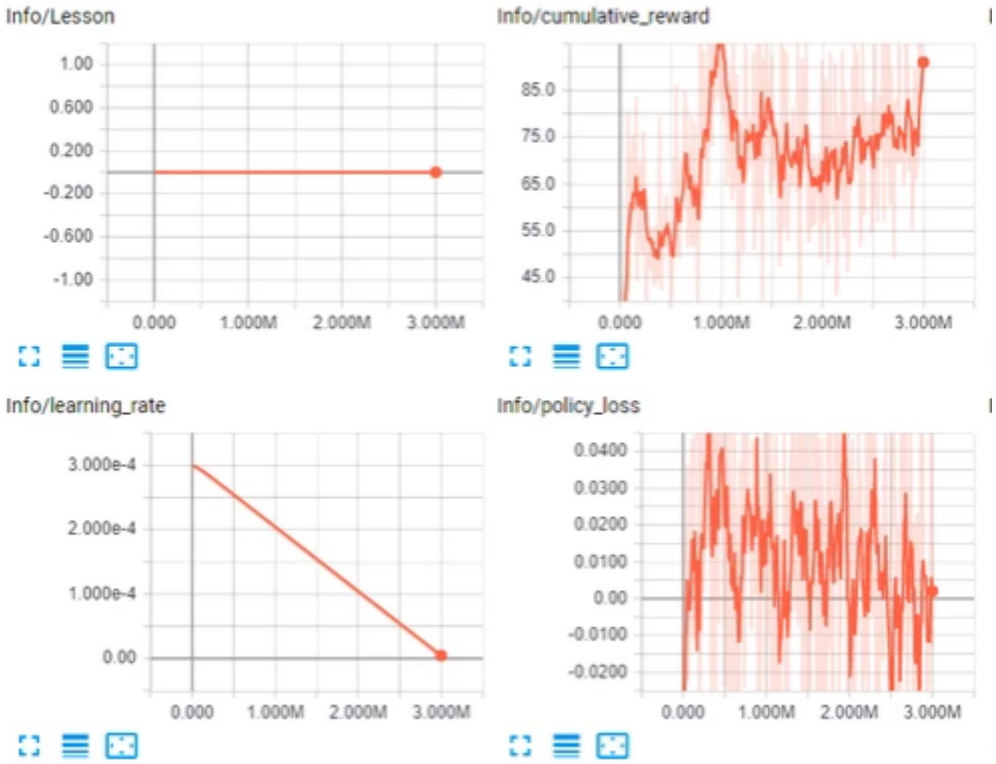

Figure 3.6: TensorBoard graphs for training the Hide scenario (Hide3) 1

- Epsilon : 0.2
- Buffer Size : 40960
- Learning Rate : 3e-4
- Hidden Units : 128
- Batch Size : 512
- normalize : False

At a time, the work could show enhanced training parameters, and the behaviour of the agent tends to show less confusion and stuck incidents.

Seeing the unstable behaviour of the training, this project was decided not to try curriculum training for the 'Hide' scenario.

3.7 The Limits of Simple Avoidance

After enough amount of time, the agent was always caught by the scripted chasing Artificial Intelligence. The agent couldn't find a set of approaches in which it was

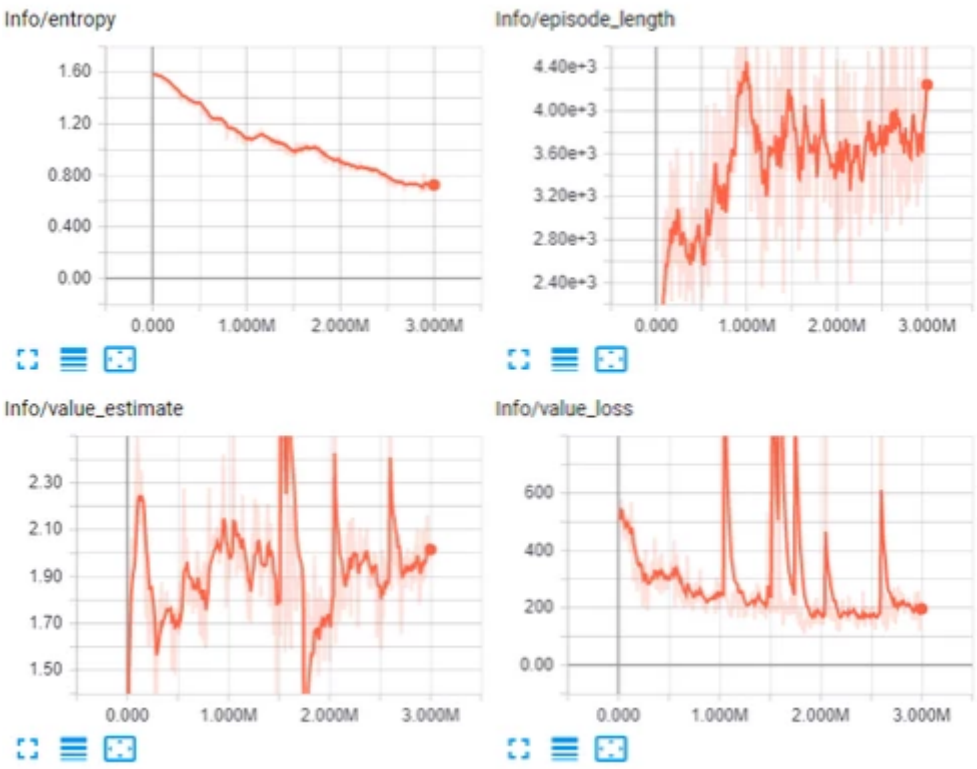

Figure 3.7: TensorBoard graphs for training the Hide scenario (Hide3) 2

able to escape from any point from the slightly slower moving scripted Artificial Intelligence indefinitely. This caused a thinking of the approach of the scenario. This scenario was definitely not practical.

If two similarly competent individuals were placed into a comparative shut environment, and one could only run from the other. After some time, the one pursuing would probably always end up the winner. The situation of indefinitely escaping from being caught in a small area does not occur out of a game of tag between the kids, and all things considered they regularly use a playground or an entire park. On the off chance that this were state a situation of an intruder, there would be some kind of fight used between the two and one would turn out the winner. Or then again the individual hiding would have the option to discover a place to hide where the one looking for the other individual would not have the option to find them at all.

It was believed that there was an optimal set of decisions in which the agent could avoid the scripted Artificial Intelligence indefinitely in this environment, but it was impossible to retrieve because of the 'Hide' training implementation limitations. The project headed to try another scenario.

```
### General parameters
max_steps = 30e5 # Set maximum number of steps to run environment.
run_path = "ppo" # The sub-directory name for model and summary statistics
load_model = False # Whether to load a saved model.
train_model = True # Whether to train the model.
summary_freq = 10000 # Frequency at which to save training statistics.
save_freq = 50000 # Frequency at which to save model.
env_name = "Hide4" # Name of the training environment file.
#curriculum_file = "curricula/hide.json"
#curriculum_file = "curricula/escape.json"
curriculum_file = None

### Algorithm-specific parameters for tuning
gamma = 0.99 # Reward discount rate.
lambd = 0.95 # Lambda parameter for GAE.
time_horizon = 2048 # How many steps to collect per agent before adding to buffer.
beta = 1e-3 # Strength of entropy regularization
num_epoch = 8 # Number of gradient descent steps per batch of experiences.
num_layers = 2 # Number of hidden layers between state/observation encoding and value/policy layers.
epsilon = 0.2 # Acceptable threshold around ratio of old and new policy probabilities.
buffer_size = 40960 # How large the experience buffer should be before gradient descent.
learning_rate = 3e-4 # Model learning rate.
hidden_units = 128 # Number of units in hidden layer.
batch_size = 512 # How many experiences per gradient descent update step.
normalize = False

### Logging dictionary for hyperparameters
hyperparameter_dict = {'max_steps':max_steps, 'run_path':run_path, 'env_name':env_name,
    'curriculum_file':curriculum_file, 'gamma':gamma, 'lambd':lambd, 'time_horizon':time_horizon,
    'beta':beta, 'num_epoch':num_epoch, 'epsilon':epsilon, 'buffe_size':buffer_size,
    'leaning_rate':learning_rate, 'hidden_units':hidden_units, 'batch_size':batch_size}
```

Figure 3.8: Improved hyperparameters, better tuned for training the Hide Scenario (Hide4)

3.8 Escape

A working scenario seemed to be escaping from the small area out into a bigger one. The scenario was reworked to include some "traps". This also gave a condition for the agent to increase its chances to increase total escape time given that the speed of the pursuer by default is higher than the speed of the agent. After testing some values in the curriculum, it was discovered that if the set of decisions of the agent was calculated with a high average reward and level out, and then sneak up near a very high good outcome frequency before increasing the chasing Artificial Intelligence's speed and learning to run away from a much faster pursuer, it would be the best option. Giving the scripted Artificial Intelligence simply more speed of 1.0 at the start comparing to the previous speed at the start of 0.5 that was used in the previous scenario also was more successful decision.

General parameters:

- Max Steps : 30e5
- Run Path : "ppo"
- Load Model : False

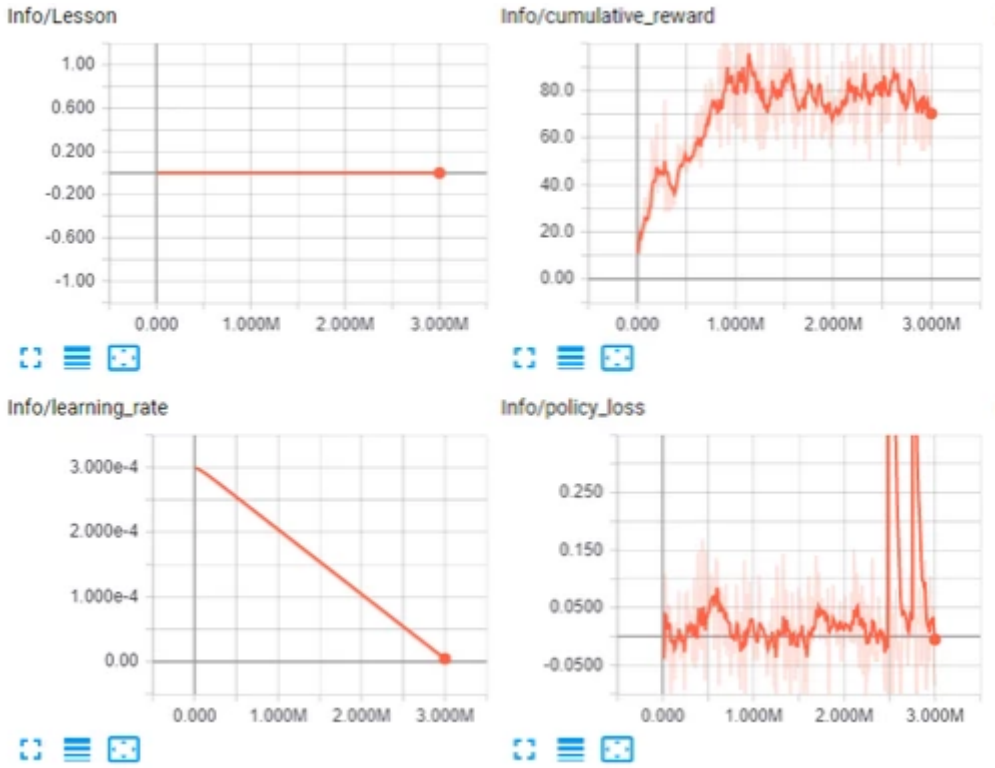

Figure 3.9: TensorBoard graphs showing more stability throughout training the Hide scenario (Hide4) 1

- Train Model : True
- Summary Frequency : 10000
- Save Frequency : 50000
- Environment Name : "Hide4"
- Curriculum File : "curricula/escape.json"

Algorithm-specific parameters for tuning:

- Gamma : 0.99
- Lambda : 0.95
- Time Horizon : 2048
- Beta : 1e-3
- Number of Epoch : 8
- Number of Layers : 2

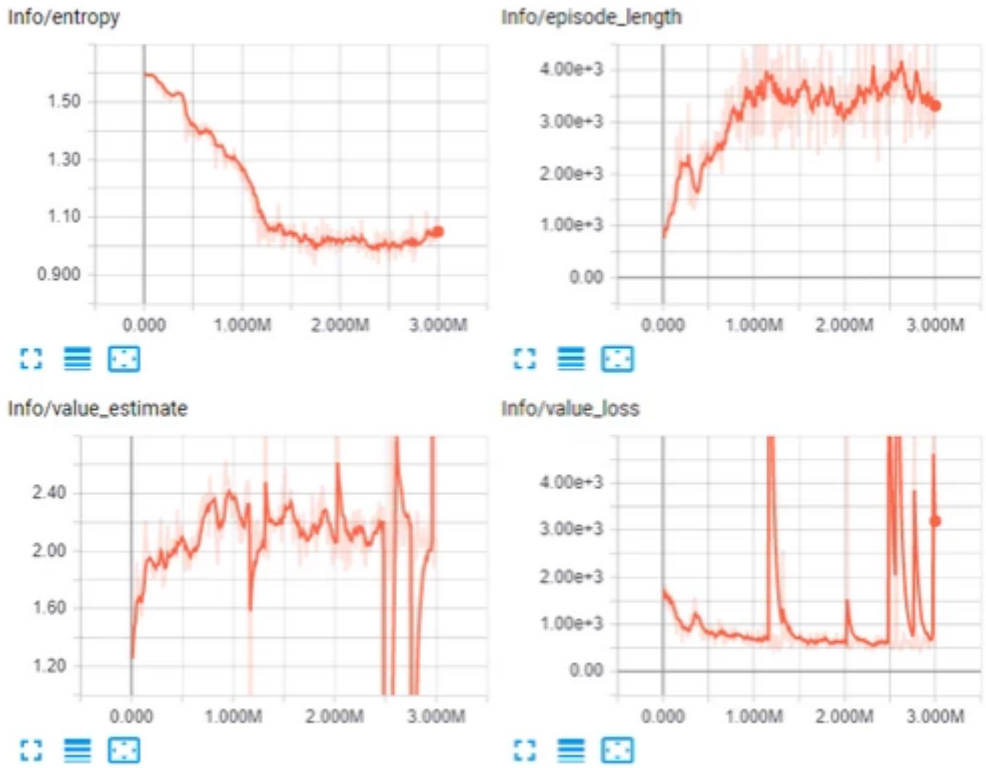

Figure 3.10: TensorBoard graphs showing more stability throughout training the Hide scenario (Hide4) 2

- Epsilon : 0.2
- Buffer Size : 10240
- Learning Rate : 3e-4
- Hidden Units : 128
- Batch Size : 1024
- normalize : False

This method appears to be worked admirably as the cumulative reward for a pursuer in the case of its maximum speed got to the value of 9.48, which roughly speaking is close to the flawless reward value of 10 as it can get to it with the penalty function which occurs to be in the escape mode. [14]

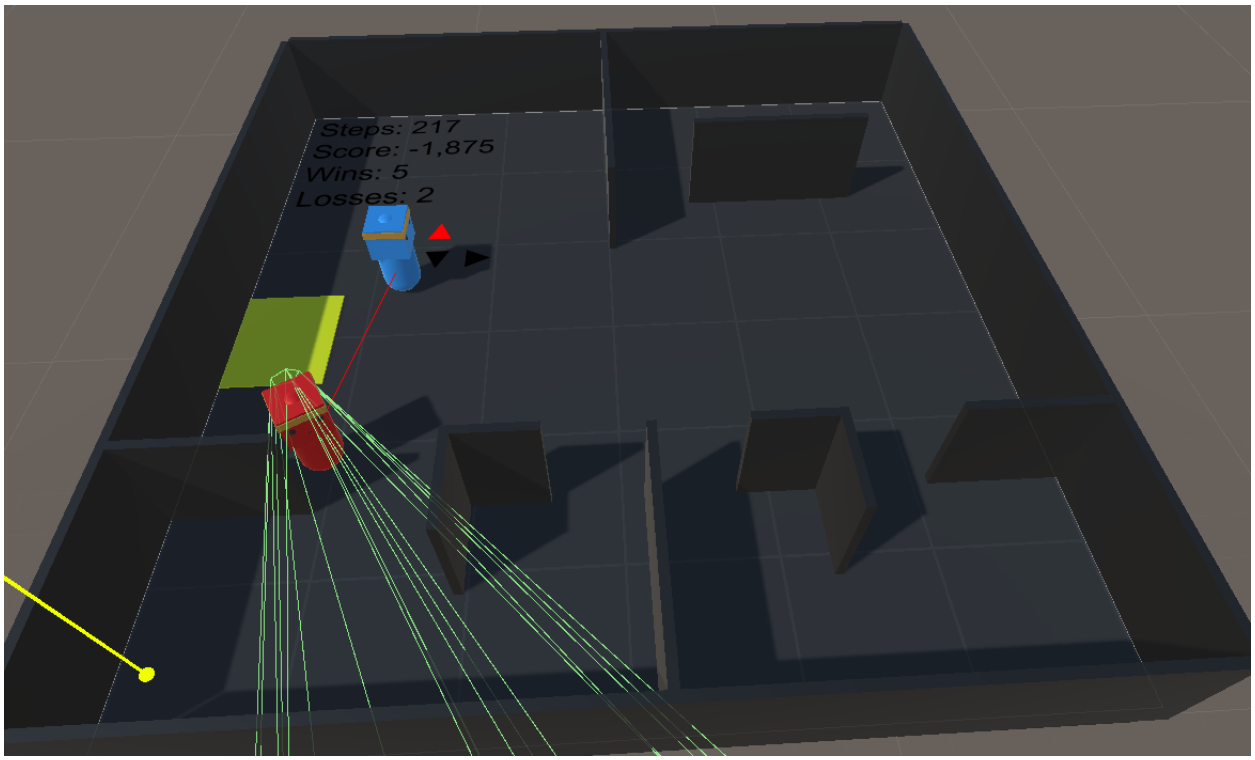

Figure 3.11: The ML Agent about to enter the win zone and score 10 points.

```
85      void Evasion()
86      {
87          if (evadedTimerActive == true)
88          {
89              if (evadedTimer > 0)
90              {
91                  evadedTimer -= Time.fixedDeltaTime;
92                  rewardBuffer += reward;
93              }
94              else
95              {
96                  evadedTimerActive = false;
97                  Debug.Log("Evaded in " + HideSpace.name);
98
99                  // Multiply reward for living if hiding, negate living penalty if escaping
100                 if (hideAcademy.mode == HideAcademy.Mode.Hide)
101                     reward += rewardBuffer * 4.0f;
102                 else if (hideAcademy.mode == HideAcademy.Mode.Escape)
103                     reward -= rewardBuffer;
104             }
105         }
106     }
```

Figure 3.12: As we had two Academy modes evasion reward system is reworked.

```
12      private void OnTriggerEnter(Collider other)
13      {
14          if(other.tag == "Player")
15          {
16              HideAgent hideAgent = other.gameObject.GetComponent<HideAgent>();
17              hideAgent.done = true;
18              hideAgent.reward += 10.0f;
19              hideAgent.wins += 1;
20              hideAgent.AgentReset();
21          }
22      }
23  }
```

Figure 3.13: Reworking how reward will be calculated.

```
1   {
2       "measure" : "reward",
3       "thresholds" : [9.0, 9.1, 9.25],
4       "min_lesson_length" : 2,
5       "signal_smoothing" : true,
6       "parameters" :
7       {
8           "chaser_speed" : [1.0, 2.0, 3.0, 4.0]
9       }
10  }
```

Figure 3.14: The Escape curriculum file

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HideAcademy : Academy
{
    public enum Mode
    {
        Hide, Escape
    }

    public Mode mode;
    public float chaserSpeed;

    public override void AcademyReset()
    {
        chaserSpeed = (float)resetParameters["chaser_speed"];
    }

    public override void AcademyStep()
    {

    }
}
```

Figure 3.15: Resetting the speed parameter from academy curriculum.

```
### General parameters
max_steps = 30e5 # Set maximum number of steps to run environment.
run_path = "ppo" # The sub-directory name for model and summary statistics
load_model = False # Whether to load a saved model.
train_model = True # Whether to train the model.
summary_freq = 10000 # Frequency at which to save training statistics.
save_freq = 50000 # Frequency at which to save model.
env_name = "EscapeFaster" # Name of the training environment file.
curriculum_file = "curricula/escape.json"
#curriculum_file = None

### Algorithm-specific parameters for tuning
gamma = 0.99 # Reward discount rate.
lambd = 0.95 # Lambda parameter for GAE.
time_horizon = 2048 # How many steps to collect per agent before adding to buffer.
beta = 1e-3 # Strength of entropy regularization
num_epoch = 8 # Number of gradient descent steps per batch of experiences.
num_layers = 2 # Number of hidden layers between state/observation encoding and value/policy layers.
epsilon = 0.2 # Acceptable threshold around ratio of old and new policy probabilities.
buffer_size = 10240 # How large the experience buffer should be before gradient descent.
learning_rate = 3e-4 # Model learning rate.
hidden_units = 128 # Number of units in hidden layer.
batch_size = 1024 # How many experiences per gradient descent update step.
normalize = False

### Logging dictionary for hyperparameters
hyperparameter_dict = {'max_steps':max_steps, 'run_path':run_path, 'env_name':env_name,
    'curriculum_file':curriculum_file, 'gamma':gamma, 'lambd':lambd, 'time_horizon':time_horizon,
    'beta':beta, 'num_epoch':num_epoch, 'epsilon':epsilon, 'buffe_size':buffer_size,
    'leaning_rate':learning_rate, 'hidden_units':hidden_units, 'batch_size':batch_size}
```

Figure 3.16: Hyperparameters for training the Escape agent (EscapeFaster)

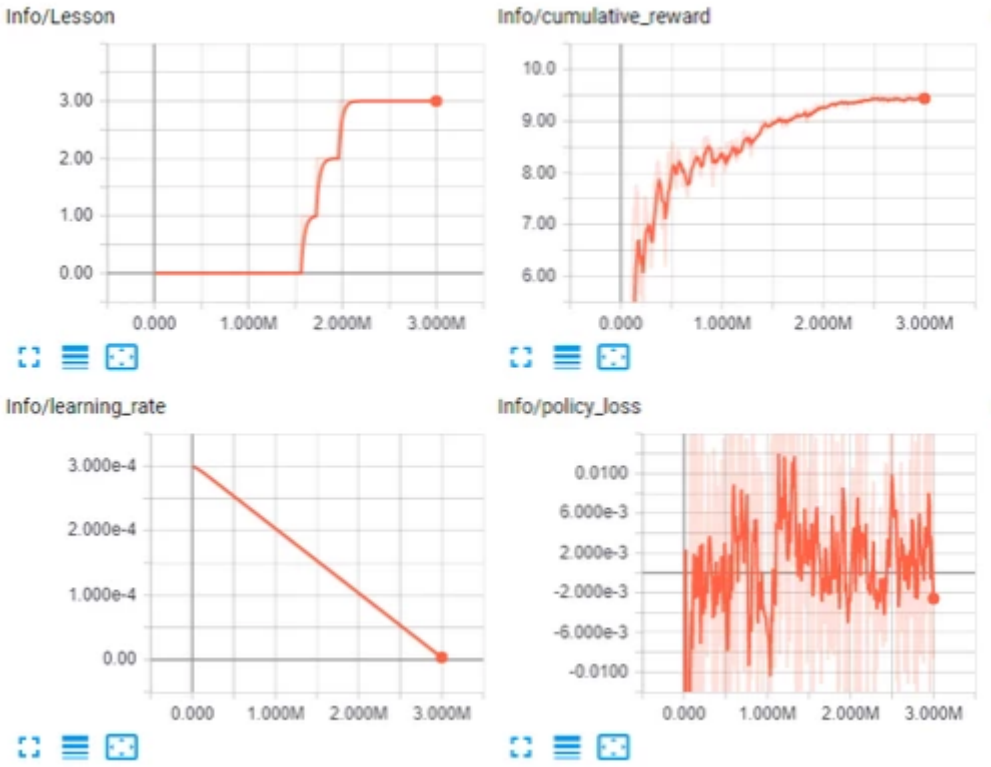

Figure 3.17: TensorBoard graphs for the Escape scenario (training to 3 million steps) 1

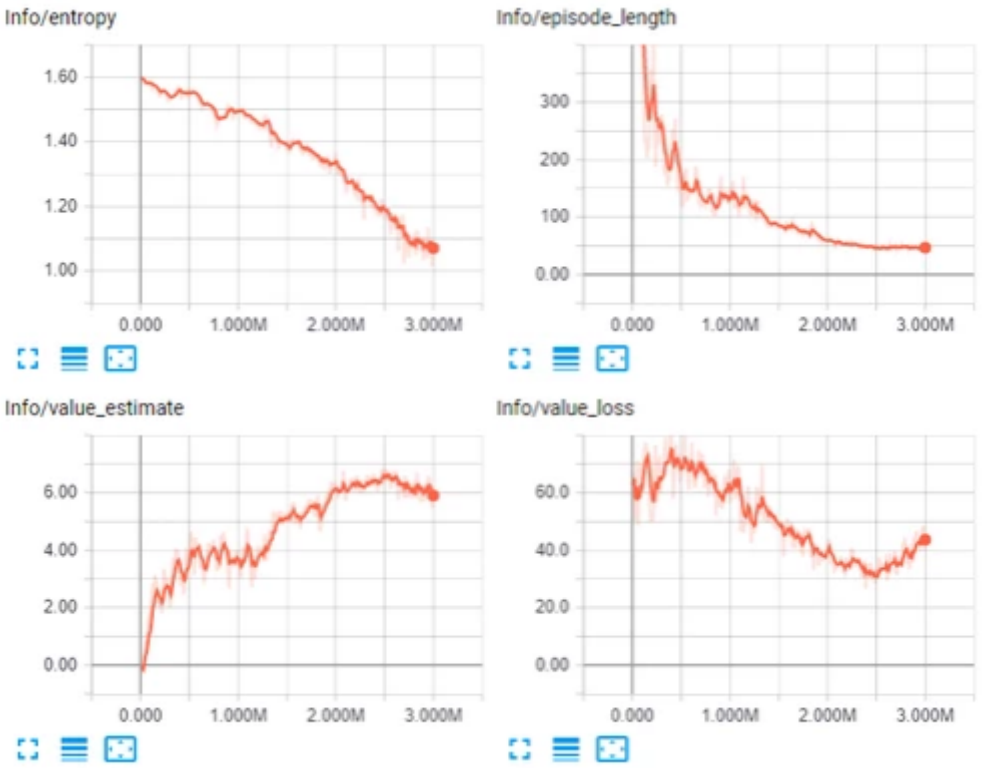

Figure 3.18: TensorBoard graphs for the Escape scenario (training to 3 million steps) 2

4. Training agent using Imitation Learning method

4.1 Hide and Escape Scenarios

In the first two versions of the environment scenario, the goal for the agent was to avoid being captured by the traditional AI. The reward was calculated by how many seconds the agent would manage to avoid the traditional AI. And in order to enhance learning speed, there were nine parallel environments with nine agents that trained simultaneously. After performing 3 million attempts in the first scenario progress of avoiding continuality occurred. But it was inconsistent and by the time there were still cases when the agent would be captured too fast. After testing some values of parameters in the curriculum, increasing buffer size twice, there were improvements in the learning, and results were more effective than in the first scenario, but still, it had flaws and inconsistent reward curve after the same 3 million attempts. Then a new approach of training was decided to perform. There was a problem that it never is possible to avoid the traditional AI indefinitely in a closed area like our environment. And even if the agent could eventually train to avoid indefinitely it meant that there was no win condition for the environment. It was decided that an agent should have an endpoint and the win condition. In this scenario, the reward was calculated by 10 minus how many seconds it took for the agent to reach the aim area. The expected result was achieved after the same 3 million attempts the agent could reach the aim area in less than a second [[14]].

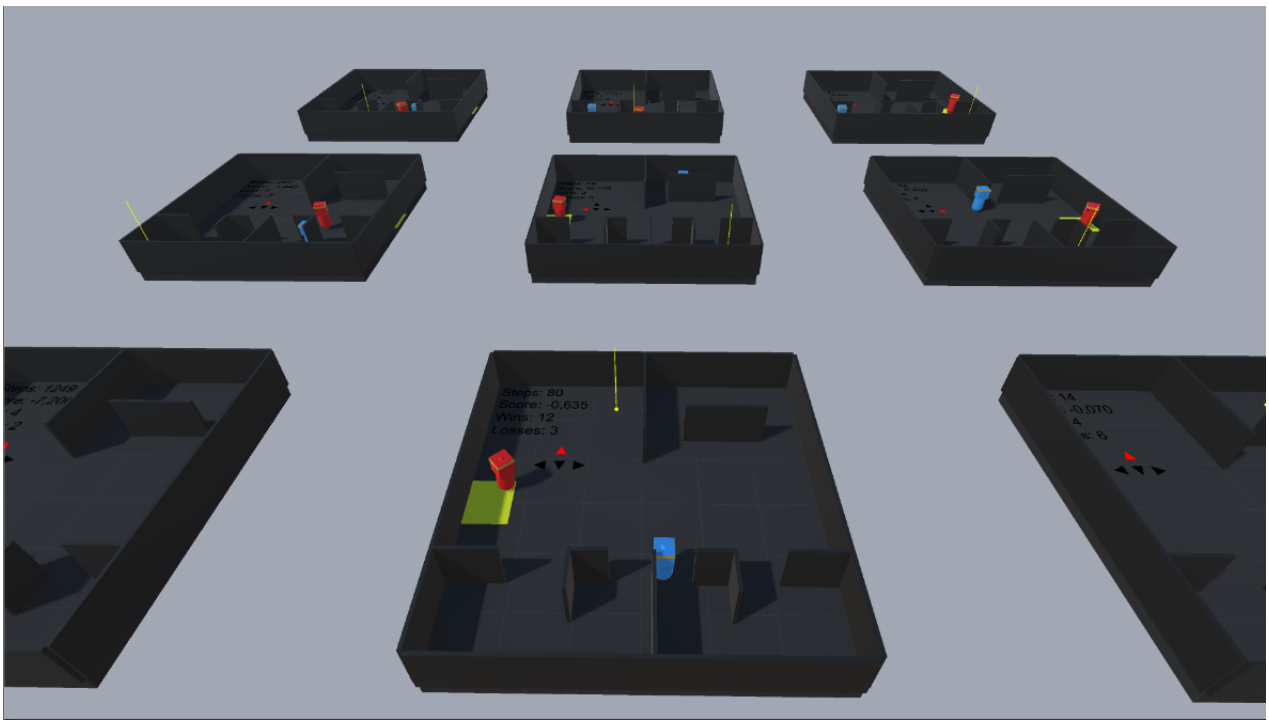

Figure 4.1: The training is enhanced by creating nine parallel environments.

4.2 Human Escape Scenario

After all the settings have been completed, it is possible to build an executable file that will be used by the Tensorflow environment. In order to start the learning process, it is necessary to insert the following command: python python/learn.py build/build.exe –run-id=0133 –train -slow A window with the game will open, and the person using the previously assigned commands will be able to control the agent. It is very important to prepare, you need to be able to play well - the better the Teacher plays, the better the neural network learns. For a game scene, it is desirable to prepare two cameras in advance. One camera (main) will be aimed at the teacher's tank. The second one (auxiliary) - at the tank of one of the trained agents - it may be placed on a screen quarter in one of the corners. This method will allow to monitor the progress of learning.

It's worth displaying all the debug information on the screen, because it's easy to make a mistake in the code and it's much better to learn about it right away and not after a failed learning attempt. It is also very helpful to use the Tensorboard tool, which displays the progress of learning. The most important indication is cumulative reward - it displays the average reward of each agent per round. Over time, the statistics values should rise.

After we reached sufficient results by using Reinforcement Learning it was time for the next step which is applying Imitation Learning to training ML agent in the same environment. Although the controls for the environment were super simple it was very time consuming to record enough data for upcoming calculations. Also, a flaw occurred in the behavior of a human being since the aim area is randomly placed in the environment, including the character and traditional AI, after each attempt. It took some time for a real person to detect its aim area and the position of the character. While the agent would know the environment state in the first frame of the attempt. And because of this, the best result that the agent would achieve after applying Reinforcement Learning was a little less than 2 seconds, which is three times more than the result of Escape scenario. But in this scenario, the agent manages to behave like a real person and if this result is expected by the developers, then one second is not that important comparing to the realism.

4.3 Comparing the methods

Comparing two methods of learning is difficult since they achieve different results and behaviors. And the intersection results may only occur for early training which may be random and not satisfactory. It is obvious that Imitation Learning approach consumed more time to perform due to recording realtime human behavior which also consumes human resources more. If the developers seek the most effective results then Reinforcement Learning method will be more preferable. And if their intentions are to develop an AI that is close to human behavior, then Imitation Learning method is the best choice. However in both approaches there were some cases when even after 3 million steps of learning, the agent was able to reach the point only when the traditional AI is gone in a very long distance. And as the traditional AI patrols the environment randomly, there were cases, when the agent was waiting unnecessarily for a long time. That happened because the agent does not consider barriers in his path like walls and also he does not consider the direction of the traditional AI.

5. Conclusion

This study explores the opportunities and benefits of single use of Reinforcement Leaning and simultaneous use of Reinforcement Learning and Imitation Learning in artificial intelligence development for video games. Tools for creating the Learning Environment and learning AI agents have been considered. Practical recommendations, allowing to optimize the parameters and characteristics of the neural network and to conduct more effective training, were given. In the final result, a video game agent, which controls the character, effectively uses the available game mechanics and whose behavior is similar to a human, was created and trained. And also a video game agent who shows the most effective result was developed and trained.

Starting to work with this tool in 2018, it was still in the development, in beta. The research was proceeded on a beta version. For now the tool is fully released, on 12th of May 2020 it was announced that ML-Agents Unity package is officially released, meaning that all of the planned features are completed that are still not learned in this work. And there are more features coming in the future as Unity team and the community are supporting and improving this amazing toolkit.

References

[1] V. Mnih, A. Puigdomenech Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning.", 2016.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, *The Arcade Learning Environment: An evaluation platform for general agents*, ser. International series of monographs on physics. J. Artif. Intell. Res., 2013.

[3] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange. (2018). Unity: A general platform for intelligent agents., [Online]. Available: `https://github.com/Unity-Technologies/ml-agents`.

[4] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning.", 2017.

[5] (2018). Openai. openai five., [Online]. Available: `https://blog.openai.com/openai-five/`.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym.", 2016.

[7] L. Orseau and S. Armstrong, "Safely interruptible agents.", 2016.

[8] Y. Zhao, I. Borovikov, F. de Mesentier Silva, A. Beirami, J. Rupert, C. Somers, J. Harder, J. Kolen, J. Pinto, R. Pourabolghasem, J. Pestrak, H. Chaput, M. Sardari, L. Lin, S. Narravula, N. Aghdaie, and K. Zaman, "Winning isn't everything:enhancing game development with intelligent agents.", 2019.

[9] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, and M. Jaderberg. (2019). Alphastar: Mastering the real-time strategy game starcraft ii., [Online]. Available: `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`.

[10] N. Bostrom, “Superintelligence: Paths, dangers, strategies.”, 2014.

[11] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, “Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures.”, 2018.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, P. Georg, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning.”, 2015.

[13] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, “Unity: A general platform for intelligent agents.”, 2020.

[14] M. Urmanov, M. Alimanova, and A. Nurkey, “Training unity machine learning agents using reinforcement learning method”, 2019.

A. Training Academy code

A.1 Evasion Algorithm of Hiding agent

```
void Evasion()
{
if (evadedTimerActive == true)
{
if (evadedTimer > 0)
{
evadedTimer -= Time.fixedDeltaTime;
rewardBuffer += reward;
}
else
{
evadedTimerActive = false;
Debug.Log("Evaded in " + HideSpace.name);
// Multiply reward for living if hiding, negate living
penalty if escaping
if (hideAcademy.mode == HideAcademy.Mode.Hide)
reward += rewardBuffer * 4.0f;
else if (hideAcademy.mode == HideAcademy.Mode.Escape)
reward -= rewardBuffer;
}
}
}
```

A.2 Win Zone Handler

```
private void OnTriggerEnter(Collider other)
{
if(other.tag == "Player")
{
HideAgent hideAgent = other.gameObject.GetComponent<HideAgent>();
hideAgent.done = true;
hideAgent.reward += 10.0f;
hideAgent.wins += 1;
hideAgent.AgentReset();
}
}
```

B. Hyperparameters

Hyperparameters for training the Escape agent (EscapeFaster)

- max_steps = 30e5
- run_path = "ppo"
- load_model = False
- train_model = True
- summary_freq = 10000
- save_freq = 50000
- env_name = "Hide4"
- curriculum_file = "curricula/escape.json"
- gamma = 0.99
- lambd = 0.95
- time_horizon = 2048
- beta = 1e-3
- num_epoch = 8
- num_layers = 2
- epsilon = 0.2
- buffer_size = 10240

- learning_rate = 3e-4
- hidden_units = 128
- batch_size = 1024
- normalize = False
- hyperparameter_dict = {'max_steps':max_steps, 'run_path':run_path, 'env_name':env_name, 'curriculum_file':curriculum_file, 'gamma':gamma, 'lambd':lambd, 'time_horizon':time_horizon, 'beta':beta, 'num_epoch':num_ep 'epsilon':epsilon, 'buffe_size':buffer_size, 'leaning_rate':learning_rate, 'hidden_units':hidden_units, 'batch_size':batch_size}