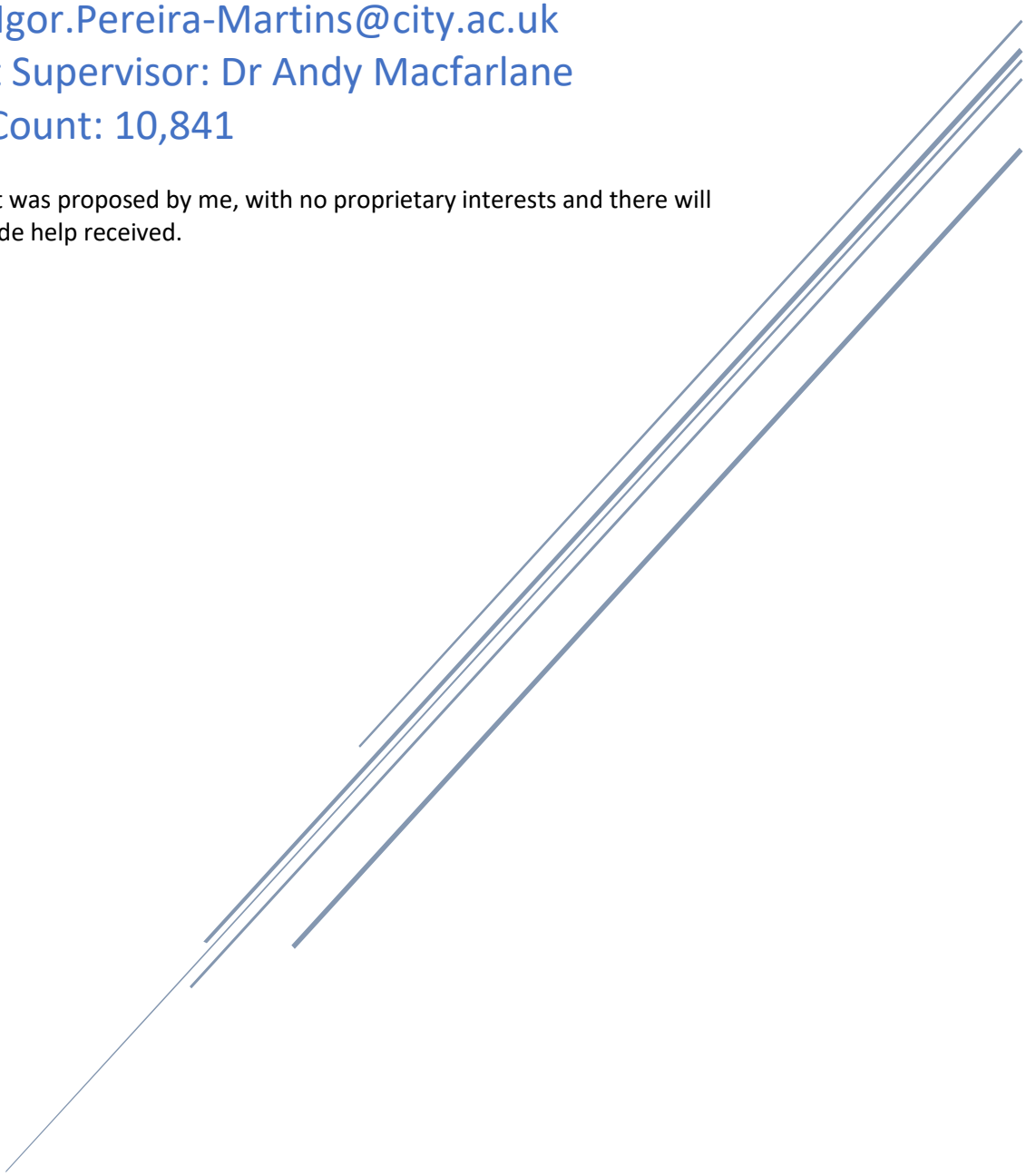
# TEACHING AN ARTIFICIAL INTELLIGENCE TO PLAY A COMPLEX TURN-BASED STRATEGY GAME – PROJECT REPORT

By: Igor Pereira Martins
Email: Igor.Pereira-Martins@city.ac.uk
Project Supervisor: Dr Andy Macfarlane
Word Count: 10,841

This project was proposed by me, with no proprietary interests and there will be no outside help received.



CITY, UNIVERSITY OF LONDON
BSC (HONS) COMPUTER SCIENCE

# Table of Contents

Abstract ..... 4
Chapter 1 – Introduction ..... 5
1.1 – Problem to be Solved ..... 5
1.2 – Project Objectives ..... 5
1.2.1 – Main Objective ..... 5
1.2.2 – Sub-Objective 1 ..... 5
1.2.3 – Sub-Objective 2 ..... 6
1.2.4 – Sub-Objective 3 ..... 6
1.2.5 – Optional Objective ..... 6
1.3 – Beneficiaries ..... 6
1.4 – Work Performed ..... 6
1.5 – Assumptions and Limitations ..... 7
Chapter 2 – Output Summary ..... 8
Chapter 3 – Literature Review ..... 9
3.1 – Computer Vision Techniques ..... 9
3.2 – Machine Learning ..... 9
3.3 – Similar Projects ..... 10
3.4 – Past Projects ..... 10
Chapter 4 – Methods ..... 11
4.1 – Lifecycle ..... 11
4.2 – Tools ..... 12
4.2.1 – Python ..... 12
4.2.1.1 – Pytesseract ..... 12
4.2.1.2 – Pyautogui ..... 12
4.2.1.3 – Difflib's SequenceMatcher ..... 12
4.2.1.4 – Pandas ..... 12
4.2.1.5 – Pillow ..... 12
4.2.1.6 – Keyboard ..... 12
4.2.2 – Data Frames ..... 12
4.2.2.1 – Bulbapedia's databases ..... 12
4.2.2.2 – Kaggle databases ..... 12
4.2.3 – Requirement collection ..... 13
4.3 – Version 1: Recognition and Database Manipulation ..... 13
4.3.1 – Pokémon Recognition ..... 13
4.3.2 – Database Manipulation ..... 13

4.3.3 – Testing and Evaluation ..... 13
4.4 – Version 2: Basic Pokémon and Move Selection ..... 14
4.4.1 – Pokémon Selection ..... 14
4.4.2 – Move Selection ..... 15
4.4.3 – Testing and Evaluation ..... 15
4.5 – Version 3: Improved Recognition ..... 15
4.5.1 – Turn Recognition ..... 15
4.5.2 – Opponent Recognition ..... 15
4.5.3 – Testing and Evaluation ..... 15
Version 4: Improved Pokémon and Move Selection ..... 15
4.5.1 – Improved Value Calculation ..... 15
4.5.2 – Improved Pokémon Selection ..... 16
4.5.4 – Final Bug Fixes ..... 16
4.5.3 – Testing and Evaluation ..... 16
Chapter 5 – Results ..... 17
5.1 – Needed Information ..... 17
5.2 – Datasets ..... 18
5.3 – Version 1 ..... 19
5.3.1 – Pokémon Recognition Prototype ..... 19
5.2.2 – Pokémon Recognition ..... 20
5.2.3 – Database Manipulation ..... 21
5.3.3 – Testing Outcomes ..... 21
5.3 – Version 2 ..... 23
5.3.1 – Pokémon Selection ..... 23
5.3.2 – Move Selection ..... 24
5.3.3 – Type Chart Maker ..... 24
5.3.4 – Testing Outcomes ..... 24
5.4 – Version 3 ..... 25
5.4.1 – Turn Recognition ..... 25
5.4.2 – Improved Opponent Recognition ..... 26
5.4.3 – Testing Outcomes ..... 26
5.5 – Version 4 ..... 28
5.5.1 – Improved Value Calculation ..... 28
5.5.2 – Improved Pokémon and Move Selection ..... 28
5.5.3 – Improved Turn Recognition ..... 29
5.5.4 – Testing Outcomes ..... 29

Chapter 6 – Reflection ..... 30
6.1 – Objectives ..... 30
6.1.1 - Main Objective ..... 30
6.1.2 - Sub Objectives ..... 30
6.1.2.1 – Sub Objective 1 ..... 30
6.1.2.1 – Sub Objective 2 ..... 30
6.1.2.1 – Sub Objective 3 ..... 31
6.1.3 - Optional Objective ..... 31
6.2 – What was learnt ..... 31
6.3 – Future work ..... 31
References ..... 32
Appendices ..... 33
Appendix A: Project Definition Document ..... 33
Research Ethics Review Form: ..... 37
Appendix B: Reuse Summary ..... 41
Appendix C: Requirements ..... 42
Appendix D: Design Documents ..... 44
Decision Tree for Simple Move Selection ..... 44
Decision Tree for Complex Move Selection ..... 45
Appendix E: Test Plans and Results ..... 46
Version 1 Test Template ..... 46
Version 1 Results – Test 1 ..... 46
Version 1 Results – Test 2 ..... 47
Version 1 Results – Test 3 ..... 48
Version 2 - Test Plan ..... 49
Version 2 - Results ..... 49
Version 4 – Test Template ..... 52
Appendix F: Source Code ..... 53
Code ..... 53
Appendix G: Data sets ..... 65
Appendix H: Executables ..... 65
Appendix I: Websites ..... 65

# Abstract

The objective of this project is to create an artificial intelligence that uses a mix of computer vision techniques and statistical analysis to play a complex strategy game at a decent level of proficiency.

The artificial intelligence's proficiency will be measured by how well it performs at the game, this includes winning games, or defeating as many opponents as possible before being defeated, it will also be measured on how well it performs in a Turing test.

This project will serve as a benchmark for how Artificial intelligence techniques have improved over the years, and it will attempt to test the limit of the capability of an artificial intelligence to play games.

The main outcomes of this project were: An AI capable of playing a full game of Pokémon to a level of skill where it bested players with several years of experience and still presents an opportunity to improve even further at the game, a dataset holding information on hundreds of different Pokémon and their different possible move sets in a competitive scene, which could help newer players to learn the game as well as more experienced players as an aid during battle, and a data set storing the effectiveness of every type on every Pokémon that currently exists.

# Chapter 1 – Introduction

## 1.1 – Problem to be Solved

Since the dawn of artificial intelligence, Computer Scientists have been pitting them against various games, “One of the earliest examples of AI playing games comes from 1951-1952, with a computerised game of Nim.” (ThinkAutomation, 2021).

Although the games started out being quite simple, AI rapidly progressed to harder and harder games, Chess in 1997 with IBM’s Deep Blue achieving victory against the then world champion, Garry Kasparov, the game of Go in 2016 with Google’s AlphaGo, and perhaps more pertinent examples to this project, Marl/O playing a game of Mario in 2015.

With all these examples in mind, it begs the question: with recent advancements in Machine Learning and Data Science, how complex can the game we teach an AI to play be? To answer this question, I have chosen to teach an AI to play a game that falls under the same genre of things that have been done before, but with much more complex features.

For this project I will be teaching an AI to participate in a Pokémon Battle, hopefully achieving a competitive standard of skill, but at minimum having it be able achieve victory against an opponent with the skill of at least the highest level of NPC opponent.

To truly push the limits of my AI, I will be training it to play on Pokémon Showdown, “a web-based Pokémon battle simulator. It is free open-source software and written in JavaScript and Node.js.” (Bulbapedia, n.d.), Specifically Randomised Battles, where your team and moves are randomly chosen for you, but follow several patterns based on move sets that perform best in competitive environments.

There are some existing examples of AI being trained to perform in Pokémon battles, and example of such is Future Sight AI that includes a google extension for battle predictions, the website can be found here: https://www.Pokémonbattlepredictor.com/home

## 1.2 – Project Objectives

The AI had four main versions, each aiming to achieve an objective or improve upon an already complete objective.

### 1.2.1 – Main Objective

The main objective of this project was to build an AI that can make advantageous decisions during a Pokémon battle, To the level where, it can pass a Turing Test

Testing this objective involved showing footage of the AI and a human player facing each other and having several participants try and conclude which one was the AI and for what reason they had made this choice, as well as having participants play against the AI and provide a critical assessment to the actions the AI chose to perform.

### 1.2.2 – Sub-Objective 1

The first sub objective was training the AI to recognise all existing Pokémon with as close to perfect accuracy as possible, either by training a classifier to recognise the sprites on screen or finding a way to read the text off the screen.

Testing this objective involved having the AI check various teams and seeing how often it managed to extract which Pokémon on were on the team.

### 1.2.3 – Sub-Objective 2

The next sub objective was to create a computer vision component that could look at the Pokémon Showdown UI and recognise all the key aspects needed for the AI to make its decisions, this included: the Pokémon the AI had on the field, the Pokémon on the AI's team, the Pokémon the opponent has on the field, and the moves the AI has available. The AI would then have to extract the correct information from what it sees to come to decision for its next move.

This objective was tested by making the AI display the outputs of its data collection and manually comparing it to the information on the screen.

### 1.2.4 – Sub-Objective 3

The final sub objective was training the AI to decide what to do next based on the information it has obtained, teaching it which decision would be the best choice in various situations.

This objective was tested by having a participant play against the AI and have them assess the choices the AI made and use this feedback to tweak how the AI makes decisions.

### 1.2.5 – Optional Objective

An Optional objective for this project was making the AI fully automated, giving it the ability to challenge opponents on its own and keep playing even after a game ends.

This objective would have been tested by allowing the AI to run until it encountered and error and seeing how far it could get with no human intervention.

## 1.3 – Beneficiaries

The first beneficiary of my project will be those who want to research the use of computer vision and Machine learning to create artificial intelligence to play or participate in more complex games, as they can use my project as a steppingstone or an informative piece that could help them get started with a project of their own or provide inspiration.

The second beneficiary of my project will be Pokémon players, as, if the AI becomes good enough at the game, it could provide significant challenge to experienced players, allowing them to play the game on their own whilst maintaining the difficulty the game used to provide, as well as testing their teams without needing another player. On top of this the datasets created could provide a guide for both new and experienced players, keeping useful information all in one place as well as providing information that would be difficult to find, such as an opponent's possible moves.

The final Beneficiary of my project will be me, not only will this project be graded, but it will also provide me the opportunity to learn and improve various skills in an area I'm interested in working in in the future, such as improving my skills at making artificial intelligence or working with computer vision.

## 1.4 – Work Performed

All the objectives mentioned above have been completed to an acceptable level. A group of datasets have been found, updated, and created to allow the AI to identify all information currently available to it on the GUI as well as search for any relevant information needed to make its decisions. On top of this a system has been created to output all the AI's calculations and decisions, allowing it to be used for the secondary purpose of serving as an assistant to a player rather than playing the game itself.

## 1.5 – Assumptions and Limitations

An assumption was made that the creation of an AI that was constantly tweaking the value of every decision until it learned exactly what path would lead to victory could be made, however, in the given time scope and based on the limits of my knowledge and experience, this was simply not feasible.

The source code for the platform the AI played on would be needed to properly build such a model, and, whilst it is readily available to the public, it is 95% in TypeScript, a language which I have no knowledge in.

This meant I had to limit the project to using computer vision to gather all the necessary information and use static formulas and decision trees for calculating the value of each decision.

# Chapter 2 – Output Summary

| OUTPUT | POKÉMON BATTLE AI |
|---|---|
| **DESCRIPTION** | The fully functional artificial intelligence optimised for playing randomised generation 8 battles on the Pokémon Showdown UI.<br>The file is in the folder Pokémon Battle AI and consists of a total of 608 lines of code including comments, all of which is original code. |
| **TYPE** | Python File |
| **RECIPIENTS AND THEIR BENEFITS** | The Author: this is a graded project that will contribute to the author's University Degree<br>Other researchers: May serve as a starting point or research material for others trying to create similar projects<br>Pokémon Showdown Users: Can be played against at any time if the player has two devices, allowing them to practice without having to look for an opponent and would guarantee no issues on the Opponents end that might end the game. |
| **USE** | Allows the deployment of the AI on any computer with a standard screen size |
| **LOCATION** | The Additional Submissions area on Moodle |

| OUTPUT | MOVE PROBABILITY CALCULATION DATA SET |
|---|---|
| **DESCRIPTION** | A dataset containing the probability of an ever-growing amount of move sets that can be had by the Pokémon that Pokémon Showdown assigns to randomised generation 8 teams. |
| **TYPE** | CSV file |
| **RECIPIENTS AND THEIR BENEFITS** | The Author: Used for training and testing the artificial intelligence during the creation of the project<br>Other researchers: Can be used to jump start an artificial intelligence of a similar type<br>Pokémon Showdown Users: with the help of a small portion of the code this dataset could be used by players to learn the game or simply provide insight that could help whilst playing the game |
| **USE** | Required by the Artificial intelligence for calculations |
| **LOCATION** | The Additional Submissions area on Moodle |

| OUTPUT | FULL TYPE EFFECTIVENESS DATA SET |
|---|---|
| **DESCRIPTION** | A dataset containing the effectiveness of each Pokémon type on every Pokémon that currently exists in standard games. The Dataset consists of 943 row and 21 columns for a total of 19,803 elements. |
| **TYPE** | CSV file |
| **RECIPIENTS AND THEIR BENEFITS** | The Author: Used for training the and testing the artificial intelligence during the creation of the project<br>Other researchers: To my knowledge no similar dataset exists, most are outdated or do not store the right information, this could prove useful to other researchers making similar projects as they won't have to make the dataset from scratch |
| **USE** | Required by the Artificial intelligence for calculations |
| **LOCATION** | The Additional Submissions area on Moodle |

# Chapter 3 – Literature Review

This project requires a deep understanding of applying computer vision to a game environment, as well as the creation of classification and decision models to process the acquired data. To familiarise myself with these techniques in the context I watched several tutorials and development logs surrounding the topic areas which helped me understand the basis upon which I could build the rest of my project in terms of how computer vision techniques interact with a GUI and how similar projects have attempted to complete a similar objective, reading research papers on creating models provided the information needed to create the models for the AI to classify certain information and make decisions based on this information.

## 3.1 – Computer Vision Techniques

"Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs" (IBM, n.d.)

Computer Vision, much like most fields in AI, has recently made great strides, allowing computers to recognise and differentiate more and more complex things, usually this is done using a deep learning model and a convolutional neural network, both things that I have prior experience with and will discuss in greater detail later in this document.

Computer Vision Algorithms need a large amount of data to run, and in order to gather the data I need as well as find some hints and tutorials for computer vision techniques that may help my project I consulted a tutorial blog: Building a Pokedex in Python (found here: https://pyimagesearch.com/2014/03/10/building-pokedex-python-getting-started-step-1-6/). This project details the creation of an algorithm that can recognize Pokémon using various techniques in the same programming language and using the same libraries that I will be using. This will be very useful to the first portion of my project

Code Bullet's video on creating an AI to play the clicker game 'Clicker Heroes' (found here: https://www.youtube.com/watch?v=ldaescGA1dY) gave me great insight into applying Computer vision techniques to a game environment, one example is checking certain spots for a pixel of a certain colour to keep track of when there has been an update in the UI, usually meaning an action is available. The insights I got from this video will help me during the second part of my project.

One of the modules taken in university has helped me with this aspect of my project, namely IN3060 Computer Vision Course, that has given me knowledge on the capabilities of Computer Vision, creating Computer Vision algorithms, and a preliminary insight into the python libraries 'Scikit-image' and 'OpenCV' required to make these algorithms

## 3.2 – Machine Learning

'Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data' (Wikipedia, 2022)

To teach the AI to make decisions I'm going to use various machine learning models, firstly, I will be using a classification model to differentiate Pokémon, recognising which Pokémon are both on the field and in the team. I will be basing this algorithm on the one found in the previously mentioned tutorial blog on Pyimagesearch, with certain differences as that algorithm is trained to recognize a different set of Pokémon.

Secondly, I will be using a Deep Reinforcement Learning algorithm to teach the AI to make decisions, to prepare myself to make this model, I found an article on training an AI to play snake using a DRL algorithm made from python libraries (found here: https://towardsdatascience.com/how-to-teach-an-ai-to-play-games-deep-reinforcement-learning-28f9b920440a). I'll be using the fact that the programming language is the same to base the code for my model on the one in this article, however, I will have to make key changes as the game the model plays is not at all similar.

Two of the modules taken in university has helped me with this aspect of my project, namely IN3062 introduction to artificial intelligence and IN3063 programming and mathematics for artificial intelligence. These have given me an introduction to the basics of AI as well as several opportunities to create varied neural networks, some of which identified images, which will be needed for the first part of my project.

## 3.3 – Similar Projects

To build a greater understanding of the project that I am creating I searched for similar projects, both to see if I could learn anything useful as well as see if there were any features that I could use to fast track certain parts of the project. Regarding this I found a project named 'Future Sight AI' (found here: https://www.youtube.com/watch?v=rhvj7CmTRkg) which does a very similar thing to my planned project, wherein it uses AI to perform Pokémon Battles, however, our projects differ in the fact that Future Sight uses the code of Pokémon Showdown in order to make Predictions, where my project will be using Computer Vision to acquire the information it needs, this means my project could easily be adapted to working with the actual Pokémon games, where the code would not be publicly available.

Future Sight also used thousands upon thousands of recorded Pokémon Showdown Battles to train the model, whereas I plan to train my model with a reinforcement learning technique, having it play the game with a prebuilt set of recommended decisions and testing whether these decisions lead to victory, altering which decision it is likely to make in a situation.

During my research I did find that having the AI perform in ranked battles and using its Trainer Score as a medium of testing how good the AI is as a battler was a very good method, and should my AI reach a standard where it can compete to such a level, I would use it to test my AI.

## 3.4 – Past Projects

Several of my past projects have been helpful in providing the knowledge and skills needed for this project.

Having previously taken the IN3062 - Introduction to Artificial Intelligence and IN3063 - Programming and Mathematics for Artificial Intelligence topics during this year my skills with both creating Artificial Intelligence and manipulating datasets are still fresh as both courses had the creation of Classifiers from data in a dataset as their final project, providing me with the skills needed to do the same in this project.

on top of this, having also done the IN3060 - Computer Vision course, which involved creating a model for Facial Emotion Recognition, providing insight into computer vision techniques and how to manipulate images into formats that are usable for processing.

# Chapter 4 – Methods

## 4.1 – Lifecycle

The model I chose for the development of this project was the DevOps deployment methodology, as it allowed me to make the project in parts and test them before going back and adding more to the project, this provided more versatility than a sequential model like waterfall as the project needs earlier parts to be functioning properly before later parts can be made, a sequential model would require everything to be finished before the testing phase which would lead to serious problems should earlier parts of the project not work which would require an overhaul of the entire project.

I chose to break up the model into several version, with each version adding a key feature, and future versions refining the existing features until the project reached completion.

The first version served the purpose of trawling the screen for information, taking screenshots of select areas and then converting the text in the images into strings that could be processed by the AI and stored in a data set, meant to emulate the eyes of a human player. This version was meant to exist as a proof of concept and was done first and independently as, if it did not function properly, the rest of the code also would not work.

The second version of the code used a very simple formula and information collected by the eyes created in the first version to assign points to the available Pokémon to choose the best option for the current situation. This version cannot play a game to completion as it is not viable past one turn, but it served the purpose of proving that the AI could at least function and now only needed to be optimised for continuous play.

The third version of the code improved the eyes of the AI, making it capable of finding the information it needed regardless of where it appeared on the screen, this was a necessary step as certain information appeared in random places which prevented the AI from playing a game to completion.

The fourth version of the code improved the formulas used for decision making, incorporating everything the AI learned from the UI and its various test games to make the best-informed decision, including considering unconventional data, such as moves that with unique effects, as well as using the data set the AI had been compiling during its infancy

The fifth version of the code would have allowed the AI to challenge opponents on its own continuously until a user told it to stop, this was not done however, both due to the complexity of implementation and time constraints.

## 4.2 – Tools

### 4.2.1 – Python

The programming language I chose for this project was Python, due to it being my best language in terms of skill and being the most popular language for both creating artificial intelligence as well as for computer vision techniques. To complete the project, I used various nonstandard python libraries that are described below:

#### 4.2.1.1 – Pytesseract

For converting the taken screenshots into texts, originally processing the image was going to be done by NumPy, where it would use pixel data to recognise images and then find any needed data in a database. This became unviable when I realised that the screenshots weren't of a high enough quality to process, so I decided to use Pytesseract as a lot of information could be gained by simply reading the text on screen and would run much quicker.

#### 4.2.1.2 – Pyautogui

For moving the mouse to pre calculated positions to reveal information as well as take screenshots of certain areas containing text that needs to be converted.
Keyboard, to allow me to control the AI manually as well as collect information for set up, such as the location the mouse needs to move to as well as the area a screenshot needs to be taken from.

#### 4.2.1.3 – Difflib's SequenceMatcher

For matching the text converted from images to existing information in the database to make sure it is correct and usable for processing.

#### 4.2.1.4 – Pandas

For navigating the various data frames to gather the information needed to fill in the variables for the selection formulas, as well as updating the AI's stored information for use in future decisions.
NumPy to perform the calculations with the variables available.

#### 4.2.1.5 – Pillow

For displaying images during testing.

#### 4.2.1.6 – Keyboard

For allowing manual control of the AI.

### 4.2.2 – Data Frames

Due to deciding to use data frames for my project I turned to various online sources as well as making my own to provide the require information for calculations to my Artificial Intelligence.

#### 4.2.2.1 – Bulbapedia's databases

Bulbapedia is a public encyclopaedia for Pokémon, it contains up to date information on all aspects of the game, including all moves and their effects, as well as having dedicated pages outlining any outliers among the moves and Pokémon and grouping them all together, making it easier to discover any problems that may arise in my project.

#### 4.2.2.2 – Kaggle databases

Kaggle is a subsidiary of Google and provides various datasets for data science and machine learning, I used it to gather whatever dataset I couldn't find on Bulbapedia.

### 4.2.3 – Requirement collection

There were several pieces of software and techniques that were used to document the requirements as well as create the diagrams needed to plan and design this project.

#### 4.2.3.1 – ClickCharts by NCH Software

I used this free alternative to Visual Paradigm to make the Entity relationship diagrams from my datasets as well as the decision trees to plan out my AI's decision making and information gathering.

#### 4.2.3.2 – Templates

I documented my requirements using the Volare template to keep them organised and provide a flow to my programming.

## 4.3 – Version 1: Recognition and Database Manipulation

The first version of the program served to check the six Pokémon available to the AI and lift useful information from both the Pokémon Showdown UI and the various spreadsheets, as well as record information for future use.

### 4.3.1 – Pokémon Recognition

Initially I was going to make a program to recognise the Pokémon sprites on screen, the original plan used NumPy to edit the pixel data of images to clean them up and compare the positioning of black pixels, problem where not all outlines were black, problem where the screenshots didn't have enough quality to process. The program successfully cleaned up the sprites I had available but couldn't recognise the sprites on the screen.

Instead, I made it recognise the text on screen, using Google's Pytesseract to convert screenshots of aspects of the GUI, using Pyautogui's screenshot function to collect the images as well as using the moveTo function to move the mouse to the right positions that would reveal the needed information, before converting the screenshots into strings that the code then cleans up so that it may be properly processed. To this end I used difflib's SequenceMatcher function to match any poorly converted text with the closest piece of information it could be.

### 4.3.2 – Database Manipulation

Once the AI was capable of recognising Pokémon it needed to be able to find any information surrounding the recognised Pokémon, to do this several data sets were created to keep track of this information as well as to store new information the AI collects for future use, such as the move set of each Pokémon the AI comes across.

To this end four datasets were made: 'All_Moves', that stored the information on all the moves in the game, Move Probability, to store all the move sets the AI has seen, 'Pokémon Type Chart', to store the effectiveness of each type on other types, this data set was used in conjunction with a piece of code, 'TypeChart maker.py', to help populate other data sets, and finally 'Pokémon', that stored the information on all the Pokémon in the game.

To access and manipulate these data sets I used the Pandas Library, allowing the AI to open the datasets and locate only the information that was useful to it, as well as write to a dataset.

### 4.3.3 – Testing and Evaluation

This feature was tested by manually comparing the strings that were processed by the program to the screenshots it took and compared how well the information was converted to text as well as checking whether the data written to the datasets was correct.

## 4.4 – Version 2: Basic Pokémon and Move Selection

The Second Version of the program focused on providing basic functionality, allowing the artificial intelligence to take part in a battle but not to a very high standard of performance, it entailed a very simple algorithm for deciding which Pokémon to send out into battle, and an equally simple decision tree to decide which move to use next.

### 4.4.1 – Pokémon Selection

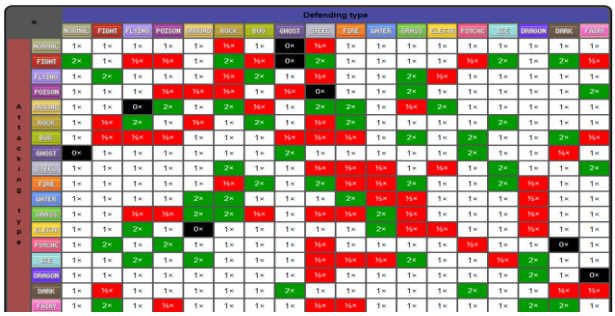
Initially I planned to use a neural network to constantly tweak the values of the variable as the AI plays more games and learns various quirks that could not be programmed into the initial formula.

I later found that a much easier process that would fit better into the time constraints I had to make this project would be to create various spreadsheets that store multiple useful pieces of information that the AI searches for and plugs into various formulas to decide the best course of action.

The formula below is what was used for this version, it allowed the AI to make very simple decisions that allowed me to test it whilst tweaking a better formula for later versions:

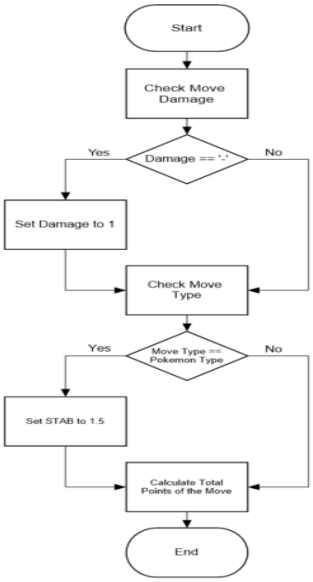
$$\sum(PM \times eff \times STAB)$$

The formula represents the combination of: A moves base damage (PM) which is stored in a dataset that holds information on all the moves in the game, the effectiveness of that move against the opponent (eff) based on type match ups that increase or decrease the damage of a move shown in the chart:




Finally, whether STAB is applied to the move, where STAB is a game mechanic that increases a moves damage by fifty percent if the Pokémon is the same type as the move it is using.

It assigned each Pokémon a certain number of points at the beginning of each turn based on the various variables and then sends out the Pokémon with the highest total move damage.




Initially I planned to use a separate neural network to have the AI decide which move to use that will maximise the damage against the opponent. Possibly provide the AI with widely used skill chains or have it calculated the amount of damage dealt to the same Pokémon based on the number of moves it needed to beat it.

Instead, using the same spreadsheets I used for the previous part of the project I created a simple decision tree to assign points to a move based on the opponent.

This decision tree only considered moves that caused damage whilst ignoring all the other types of moves, this was due to the formula being used to calculate the value of a move being very simple and mostly a proof of concept to use during the early phases of the project.

### 4.4.2 – Move Selection

Making the AI select a move was simply done by storing which move had the most value and then moving the mouse using pyautogui to select where the move would be, since moves are usually in a fixed position, in case of exceptions the AI would click where attacks may appear.

### 4.4.3 – Testing and Evaluation

To test this version, I made the program print the calculated values and ensure that it chose the correct Pokémon and move to select.

## 4.5 – Version 3: Improved Recognition

By this version the project had achieved all three sub objectives to a limited extent, and so this version focused on improving how the AI functioned, however, before the decision making could be improved the recognition needed to be worked on.

### 4.5.1 – Turn Recognition

The next step to make the AI play a full game was making it recognise what turn it was, this was done by using pyautogui to take a screen shot of a certain area of the UI that always contains what turn it is and converting the image into text, if the turn was no longer the same the AI would start collecting information again to continue playing the game.

### 4.5.2 – Opponent Recognition

Due to a problem that arose with identifying the opponent after the first turn, I had to apply a new algorithm for searching for information, to this end I used another Pyautogui feature that allows me to search for whether an image exists on the screen and returns the position of said image.

I chose to search the screen for the Pokémon's type (shown below) once the mouse was in position, as the UI always represents this in the same way, and it is always the same distance away from the information the AI needed.


### 4.5.3 – Testing and Evaluation

To test this version, I made sure that the program could find the information it needed regardless of where on the screen it appeared as well as playing several games to test how good the AI was at recognising that the turn had changed.

## Version 4: Improved Pokémon and Move Selection

### 4.5.1 – Improved Value Calculation

The focus for this version of the project was optimising the AI's ability to make decisions, this involved using much more complex decision tree and formula. As well as optimising the AI's ability to find the necessary UI components and keeping track of what turn it was.

In this version the AI began considering moves that dealt more than just damage, this included setting up stage hazards, using healing moves when health is low and using moves that apply status effects at opportune times.

This required a much more complex decision tree (Appendix D), and an ability to check the health of the Pokémon currently on the field, this was done by checking the colour of the health bar, as, once it stops being green, the Pokémon has health lower than 50 percent which is the most a healing move can heal.

### 4.5.2 – Improved Pokémon Selection

For this step the AI finally had to take into consideration the possible moves of its opponent's Pokémon using the data set that it had been creating since the first version of the project.

This was done by using Pandas to calculate the probability of any move the opponent could have and using a similar algorithm to what the AI itself uses to calculate how much damage these moves could cause; this would then be taken away from the potential damage the AI's Pokémon could deal.

This was done with a new formula:

$$\frac{((PM \times eff \times STAB) - OpponentDamage) \times UsedStat \times Accuracy \times Speed)}{InBattle}$$

Where: PM, eff, STAB, and Speed have been previously discussed, OpponentDamage is the potential damage of all the opponents' possible moves, UsedStat is the stat that governs how much damage the move deals, and Accuracy is the chance of the move hitting, and finally InBattle is either 1 or 2 depending on whether the current calculations are being done for the Pokémon that is currently on the field.

### 4.5.4 – Final Bug Fixes

Finally, in this version, any remaining known bugs were fixed and the AI's ability to recognize the turn it was on was improved.

Using pyautogui's LocateOnScreen function to check whether a new turn has started based on the option to attack being present again instead of reading text from screenshots, as this had proven unreliable and was only used when entirely necessary.

### 4.5.3 – Testing and Evaluation

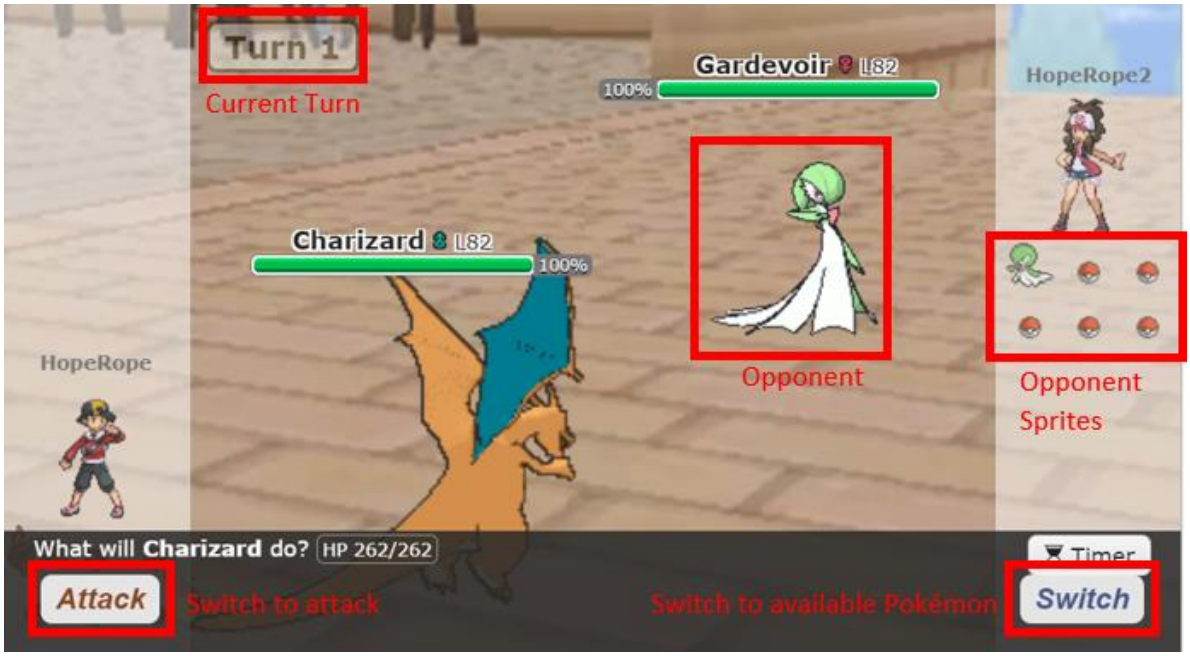
As this was the final version, the AI was tested as a whole, first, the participants played against the AI, helping to point out potential tweaks to the calculations and testing for any bugs that still existed in the code.

Once the AI worked to a good enough standard, three videos were taken of the AI playing against a human player, and then shown to each participant. The participants would then try to guess which one was the AI and why they thought this was so.

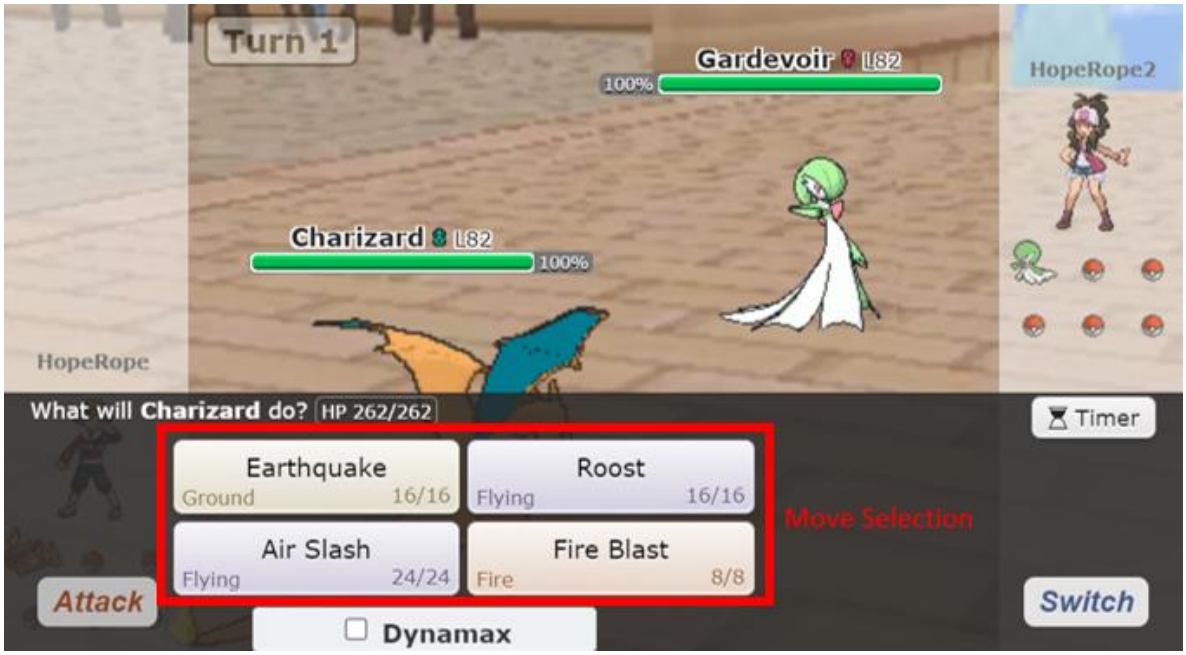
# Chapter 5 – Results

## 5.1 – Needed Information

This section will describe the results of the evaluation of the techniques described in the methods section. To understand these results and why I tested them in the way I did one needs at least a surface understanding of the Pokémon Showdown UI which will be outlined and explained below:




This image shows the general UI, this is what a player is greeted by at the start of every turn, several important aspects of the game are accessed from here, and the AI needed to either interact with these or be capable of finding them on the screen.

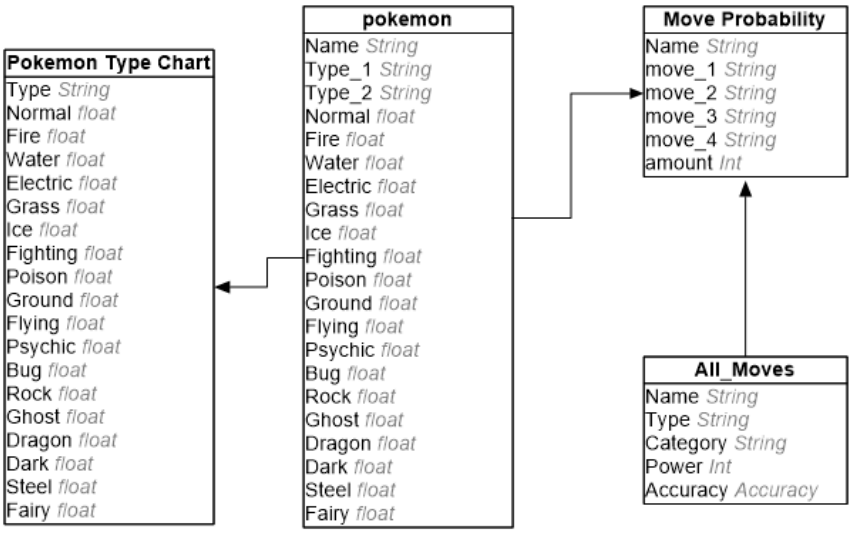



This image shows the move selection menu, where a plyer picks the move the current Pokémon will use for that turn, the AI needed to locate the right move based on either how much damage the move would do or if it is the right turn, which special effect to apply.


This figure shows the Pokémon Selection menu, where a Player chooses which Pokémon to put on the field at the cost of a turn. The AI needed the capability of locating the correct Pokémon based on several factors it would need to learn by interacting with the UI.

## 5.2 – Datasets

The Datasets were created with a mix of Bulbapedia's provided data sets, Kaggle, and filling them in by hand based on research or code created specifically for filling in data based on certain data (Appendix G), there were four data sets that came from this that related to one another as shown in the Entity Relationship Diagram:




'Pokémon Type Chart' consisted of each Pokémon type and the damage modifier applied to the damage it deals against every other type.

Pokémon contains the name of every Pokémon, their types and the effectiveness of every type against them.

Move Probability consisted of the name of every Pokémon the AI has come across, the four moves they had, and how many times that Pokémon has had those four moves.

Finally, All_Moves contain the name of every move in the game, the type of that move, what category it belongs to which denotes the stat that affects it's damage, the damage the move deals, and how often the moves lands a hit.

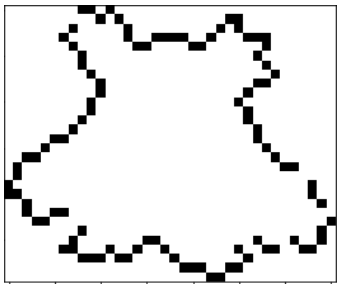
## 5.3 – Version 1

This version's principal objective was to address the first sub requirement: having the AI recognise Pokémon to a high degree of accuracy, therefore the output for this version was several arrays containing information on the Pokémon in the AI's team as well as editing the database containing the move set of the Pokémon the AI had come across.

### 5.3.1 – Pokémon Recognition Prototype

At first the objective was to create a neural network to recognise the sprites shown to the player by the UI, the way a human player would recognise the Pokémon on their team. To do this a folder containing every Pokémon Sprite currently useable in combat was created to train the network.

Then, to reduce how long the model would take to process all the images, as, from prior experience, processing time is a very common issue with the hardware I am working with, a prototype was created. It used several techniques were used to manipulate the pixel data of the sprites into a black and white outline of the sprite cropped down to its smallest size:



There was an issue when it came to creating a classifier, however, as the sprites on screen would always be the same, and it would mean the classifier would need to be able to recognise almost 900 different classes.

Instead of this, the use Pyautogui's screenshotting features and NumPy was considered to identify the Pokémon, checking the pixels of the screenshots, and comparing them to the saved sprite images, this would have taken less time to create, but would have run much slower than the classifier.

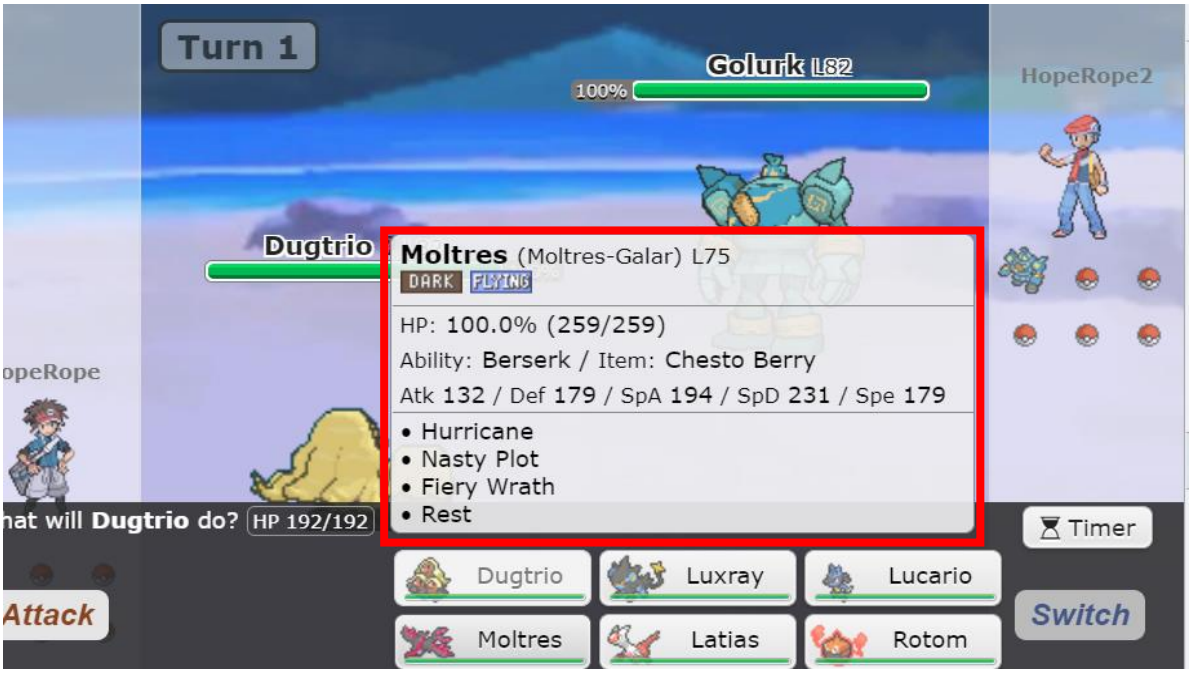
It was later found that the images the AI could screenshot from the UI were not of a high enough quality to compare to the images saved in the folder, therefore an Pytesseract was used to convert the text from screenshots of the UI whilst it displayed information on the AI's Pokémon into strings was created instead.

## 5.2.2 – Pokémon Recognition

As this was simply an attempt to successfully collect any form of information, this version of the AI was very rigid and incapable of playing even one turn of the game. It served only to test the functionality of PyAutogui and Pytesseract.

Due to this, the mouse was moved, and screenshots were taken based on fixed points, and the AI was incapable of reacting to any outliers that might move the information the AI needed to read. It was, however, capable of collecting information, which was already a step up from the failed first prototype.

The image below displays the area the AI would take a screenshot of for the first Pokémon (More examples in Appendix):




Running this through Pytesseract created six lists storing each line of information as a separate element, this was to make the information easier to manipulate, however, these lists were full of noise and needed to be cleaned up with several filter to keep only the information the AI needed.

```
['Charizard 8 Ls2', 'HP: 100.0% (262/262)', 'Ability: Solar Power / Item:
Heavy-Duty Boots', 'Atk 185 / Def 175 / SpA 226 / SpD 187 / Spe 211', '', '«
Earthquake', '* Roost', '', '¢ Air Slash', '', 'e Fire Blast']
```

To convert this information to a useable format, I created and algorithm that handled each element individually, converting it to the required form, as well as removing any unnecessary pieces of information, leaving the AI with usable data:

```
[['Charizard', 'HP: 100.0% (262/262)', 'Ability: Solar Power', 'Item: Heavy-
Duty Boots', 'Atk 185', 'Def 175', 'SpA 226', 'SpD 187', 'Spe 211',
'Earthquake', 'Roost', 'Air Slash', 'Fire Blast']]
```

During this process several exceptions needed to be considered, such as names that were more than one word, or alternate Pokémon forms, after which the information was separated and stored in the Move Probability data set for later use.

## 5.2.3 – Database Manipulation

One of the primary outputs of this project was a data set containing every Pokémon the AI has come across, their move sets, and how often this Pokémon has appeared with that move set.

This output would be used in later parts of the project to allow the AI to make informed decisions based on the opponents Pokémon as well as its own, this would allow the AI to grow dynamically and make better decisions the more it plays.

An extract from the dataset is shown below:

| | Name | move_1 | move_2 | move_3 | move_4 | amount |
|---|---|---|---|---|---|---|
| 2 | Abomasnow | Aurora Veil | Blizzard | Earthquake | Wood Hamme | 1 |
| 3 | Aerodactyl | Aqua Tail | Earthquake | Retaliate | Stone Edge | 1 |
| 4 | Aromatisse | Calm Mind | Moonblast | Protect | Wish | 1 |
| 5 | Audino | Heal Bell | Knock Off | Protect | Wish | 1 |

Creating this dataset involved condensing the information collected by the AI further into the Pokémon's name and its four moves, sorting the moves into alphabetical order so that the same move set would simply increase the amount it had been seen.

## 5.3.3 – Testing Outcomes

This version was to be tested by manually comparing the information gathered by the AI to the actual information given by the UI.

To do this the AI was run on several games, of which three separate games that contained all the different data that needed to be considered were documented fully for a total of 18 Pokémon, of these 18, the AI was capable of recognising 12 Pokémon, it was here that Pytesseract was found unreliable, where certain times it would have no problems converting the screenshots, other times it would fail to obtain text from the exact same screenshot, this problem was addressed by reducing the size of the screenshot taken and increasing the amount of screenshots to contain only certain information as outlined below:


The test was then run again on these three games, increasing the accuracy of the AI to 16 out of 18.

As the AI was being tested it was revealed that several Pokémon were missing from the dataset and researching for these missing Pokémon was required. This led to the discovery of the fact that alternate forms and names with spaces did not function well with how the Project cleaned up the noise of the screenshots. Therefore, the function that cleaned up the acquired information needed to be updated to consider these issues.

This was remedied by searching for keywords that denoted alternative forms and creating a list with all Pokémon with spaces in their name and cleaning up the noise in a different fashion:

| **Before** | **After** |
|---|---|
| arr[0] = arr[0].split(' ', 1)[0] | if ' '.join(arr[0].split(" ", 2)[:2]) in SpaceNames:<br>arr[0] = ' '.join(arr[0].split(" ", 2)[:2])<br><br>elif 'Alola' in arr[0] or 'Galar' in arr[0]:<br>arr[0] = arr[0].split(')', 1)[0] + ")"<br><br>else:<br>arr[0] = arr[0].split(' ', 1)[0] |

Finally, there would be occasions the AI collected only snippets of the information required, so method of converting these snippets into usable information was devised using difflib's SequenceMatcher to compare the information that was gathered to information the AI had stored in the other datasets, namely the 'All_Moves' and 'Pokémon' datasets, finding the closest thing to the gathered information, and using that.

Implementing this increased the AI's accuracy to the necessary 18 out of 18.

## 5.3 – Version 2

This version of the project aimed to fulfil the third sub requirement by adding on to the capabilities of version 1, in this version a basic algorithm is used to select the Pokémon with the potential to deal the most damage against the current opponent and should said Pokémon already be out on the field, use the move with the highest potential of damage against the current opponent.

### 5.3.1 – Pokémon Selection

The next step in making the AI capable of playing the game was training it to recognise which of it's six Pokémon was the best choice against the opponent's current Pokémon, the first step to achieving this was giving it the ability to check what Pokémon was currently on the opposing side.

This was done in a similar fashion to how the AI checked its own Pokémon: by taking a screenshot of a predetermined area on the screen and converting the image into a string that could be processed by the program.

An example of this bounding box is shown below:


Once this was achieved and the Name was run through the same noise remover used on its own Pokémon, the AI would cross reference the Opponent's name with the 'Pokémon' dataset to check if it had collected the correct data, if this was not the case it would use the same method as in the previous version, difflib's SequenceMatcher, to check what the closest piece of real information was to what it had gathered and use that for its calculations instead.

Each of the Pokémon's moves were then assigned a value based on the formula 'PM x Effectiveness x STAB' (described in 4.4.1). Where 'Effectiveness' was taken from the output of a separate program, 'TypeChart Maker.py', (described in 5.3.3)

These calculations were then outputted to the console to keep track of the AI's decisions and allow this version to be tested, the format of this output is shown below:

Finally, once each possible moves' value was calculated the AI would send out the Pokémon with the highest total, if that Pokémon happened to already be on the field, it would instead open the attack menu and select a move.

## 5.3.2 – Move Selection

As the AI already had the value of each move, selecting the moves themselves was done simply by locating that move on the screen, done in this version by a list of fixed positions, and using PyAutogui to move the mouse and clicking on the chosen move.

## 5.3.3 – Type Chart Maker

In this version the effectiveness of each type was used to calculate the total value of every move, to upon searching for a complete dataset with every Pokémon and these respective values on both Bulbapedia and Kaggle, it was found that there were no premade datasets that matched the needs of this project.

Therefore, the only way to gather this information would have been to go Pokémon by Pokémon and check the effectiveness or find a way to automatically create my own.

The latter option provided a much more efficient solution, and, using two of the data sets available to me: 'Pokémon' and 'Pokémon Type Chart' I created a program that would calculate and fill in the type effectiveness for every Pokémon in the dataset.

This led to two outputs of this Project: First a future proofed program that can calculate the effectiveness of every type against any Pokémon if its types are on hand, and a dataset that stores all these values in the same place and that can be updated as more Pokémon are created for each generation of the game.

The type effectiveness multipliers are stored in the last eighteen columns of the 'Pokémon' dataset, a snippet of which is shown below containing only the first 3 rows and 10 columns (the whole dataset can be found at ):

| Name | Type_1 | Type_2 | Normal | Fire | Water | Electric | Grass | Ice | Fighting |
|---|---|---|---|---|---|---|---|---|---|
| Bulbasaur | Grass | Poison | 1 | 2 | 0.5 | 0.5 | 0.25 | 2 | 0.5 |
| Ivysaur | Grass | Poison | 1 | 2 | 0.5 | 0.5 | 0.25 | 2 | 0.5 |
| Venusaur | Grass | Poison | 1 | 2 | 0.5 | 0.5 | 0.25 | 2 | 0.5 |

## 5.3.4 – Testing Outcomes

This version was to be tested by manually first by checking that the AI was calculating the values of each move correctly, and then checking to see if the AI was correctly choosing the move it had calculated.

To test the calculations the AI was run on three separate games for a total of 72 moves, of these 72, the AI correctly calculated all 72 moves, so there were no issues with the AI's ability to find and plug in the correct values into the formula.

An issue did arise, however, when it came to the AI choosing which Pokémon to switch into. When it was tested on these same games the AI would choose the Pokémon correctly, however if it needed to attack it would often choose the wrong move, this was discovered to be because the list of moves was sorted into alphabetical order so that when they were stored in the Move Probability move set it would be possible to find repeat moves.

This issue was fixed by creating two separate list: one that would be sorted before being exported to the dataset, and one that would be kept unsorted so that it reflected the order the moves were stored in the UI.

## 5.4 – Version 3

As the three sub objectives had already been started on or even completed, this version aimed to improve the AI's ability to recognise information and make it capable of autonomously play a whole game. This was done by giving the AI the ability to recognise what turn the battle was currently on as well as reworking the AI's ability to recognise the Opponent so the method would function on turns beyond the first.

### 5.4.1 – Turn Recognition

For the AI to play a full game a while loop needed to be implemented that would keep the program running until the end of the game, this created the problem of the AI going through the motions without stopping or considering the human element involved in playing the game.

If the AI had finished calculating and making its decision before the opponent, it would simply begin the next turn despite the UI not being prepared to accept the required inputs, this created  need for some form of lock that would keep the AI waiting for the next turn to begin.

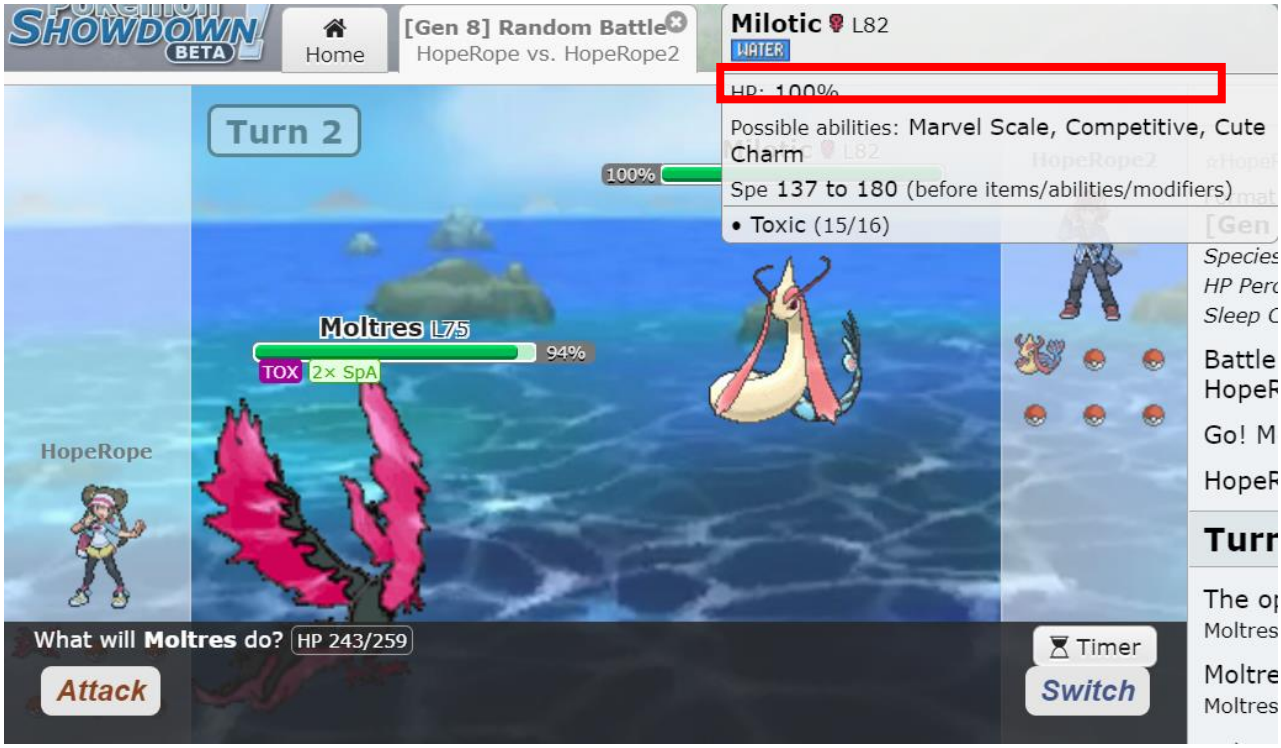
This was done much like how most of the AI's information gathering had been done so far, making the AI take a screenshot of where the UI displayed which turn it was and convert the information held there into text so that it could compare it to the information it already held.

The position of the Screenshot is showed below:


Once this was done the AI kept a record of what turn it was on and once it noticed that the current turn was no longer the same as the one it thought it was still on it would begin the process of playing a turn again.

## 5.4.2 – Improved Opponent Recognition

Once the AI was given the capacity of playing multiple turns problems began to crop up when it came to recognising the opponent past the first turn, this problem was caused by the UI being inconsistent with where the information would show up past after the first turn, an example of this is shown below, with the bounding box that the AI used to take screenshots clearly no longer taking screenshots of the correct area:




To fix this issue, the AI had to be able to actively search for the area it would need a screenshot on every time it needed information on the Opponent, this was done by giving the AI an image of every type label that was extracted directly from the UI and making the AI look for these type labels.

The images were taken from the UI as screenshots, and PyAutoGui's locateOnScreen function was used to find it on the screen, this was approach was researched to prevent the same issues as what had happened with my prototype, ensuring that the image found on screen and the on the AI was looking for would be the same.

The outcome of this new method was the AI being capable of always finding its opponents data, allowing to play a game until the end, making it possible to test the AI.

The knowledge this new method provided also allowed me to consider an new approach to Pokémon Recognition, by using this same concept to gather information on the Pokémon in the field faster and more accurately than the current method, but this was considered unfeasible, as I would need to potentially screenshot and manually crop over 800 Pokémon as I had done for the 18 types, and that having the AI check every one of these sprites in the folder would potentially be slower than the current method anyway.

## 5.4.3 – Testing Outcomes

As the AI could finally play a full game, I began testing how well the newly implemented features performed, as well as how well the AI performed as a whole, taking notes of any final tweaks that would need to be performed to achieve the main objective.

Initially the AI's ability to recognise what turn it was on worked properly, however, a problem arose when the turn entered the double digits, this was due to the location of the screenshot no longer encompassing the whole portion of the UI that showed the screen, this meant the AI could no longer keep track of the turn correctly, and on occasion it would start the next turn before it was time causing itself to crash.

To fix this, the AI was made to change the size of the area where it would take a screenshot once the turn hit double digits, this ensured accuracy in the earlier turns, and allowing it to continue functioning late in the game.

The improved Opponent recognition was tested on five games, for a total of 30 Pokémon, the new system proved to be perfectly capable of taking the screenshots from the correct locations as all 30 of the Opponent Pokémon that needed to be recognised had clean screenshots of their names taken, and any issues that arose came from Pytesseract's unreliability.

This gave me an insight into the reliability of the LocateOnScreen function, and it was decided that in future versions any features that could be replaced or optimised using this function would be, as the AI had constant issues with failing to collect information causing it to crash.

This was a particularly big issue, as the AI could only start a game from the first turn due to how turn recognition was set up and every time it failed to properly convert images to text a whole new battle would need to be started.

Finally, the overall performance of the AI was tested over several games by a participant, and suggestions were made about what the AI should have or could have done each turn.

During this testing several improvements were proposed for the AI:

The ability to consider moves other than attacking moves was a prominent comment, as often a Pokémon had a set of moves that were used to take advantage of several status moves, and the AI only considered raw damage.

A tweak to the AI's calculations that considered whether a Pokémon was already on the field or not, this was due to the AI swapping out too often and wasting turns.

The ability to take the opponent's possible moves into account as it kept disregarding the damage the Opponent could do to its own Pokémon and simply sending out the Pokémon who would deal the most damage.

Finally, the ability to take a Pokémon's statistics into account, this would not only improve the calculation of a moves value but would also stop the AI from using a Pokémon that had its statistics heavily modified, making it worse than it seemed initially.

## 5.5 – Version 4

As this was the final version of the project the aim was to fully address the main objective, to do this the AI needed to consider all aspects of the game and be able to use all the moves available to it to decide the best course of action as well as allow it to mimic a human player as much as possible.

### 5.5.1 – Improved Value Calculation

The first thing that needed to be done was improving the AI's ability to calculate the value of each move a Pokémon could take.

This required several changes, the first of which was giving the AI the ability to consider moves that didn't directly deal damage, this was done by having the AI search the data sets to check which one of the three types the moves could be: Physical, Status, and Special, if the move was a Status move then the damage of this move would be changed based on how long the current Pokémon has been on the field for.

This meant that the AI would use status moves when it's Pokémon still have high health and can risk wasting a turn of damage and that the AI would switch to dealing as much damage as it could whenever its health got low.

Secondly, the AI needed the ability to consider healing moves separately to other status moves, this is because if a healing move was used to early it would waste a turn and make it obvious that the AI was not human. This was done by checking the colour of the Pokémon's health bar, as anything other than a green bar represented Health that was lower than 50% and that was the best time to use a healing move. Initially this was done by trying to read the health information off the UI, however, checking colour was much more reliable.

Thirdly the extra pieces of information needed to plug into the new formula (shown in 4.5.2) needed to be gathered, the correct stat that affected the moves damage as well as the Pokémon's Accuracy and Speed were collected by converting a screenshot of the Pokémon's information from the UI, and the Opponent's damage was done by taking the information gathered from the Move Probability data set that the AI had been creating through out it's existence, finally in battle simply checked if it was calculating moves for the Pokémon currently out, if not then value was halved to take into account the turn that would be wasted switching the current Pokémon out.

Finally, the AI needed to take into consideration Pokémon that had lost all their HP and could not be used, this was to stop it from confusing itself into believing it was using a Pokémon that could no longer be sent out, if any Pokémon was no longer useable, all its points were set to 0.

### 5.5.2 – Improved Pokémon and Move Selection

To make the AI more flexible, this version overhauled the way the AI found both where its Pokémon and its Moves were located.

On top of this the AI was taught to recognise changes in the UI for when if it was still attacking or if a Pokémon had been sent out successfully, allowing it to react to effects that hadn't previously been considered, such as a Pokémon being forced to switch, or a move being locked by an ability, status, or item.

This used pyAutoGui's LocateOnScreen function to calculate these positions and check for these situations each turn, rather than using fixed positions as it had previously, allowing the AI to react to uncommon changes in the UI caused by certain effects that would previously cause it to crash.

### 5.5.3 – Improved Turn Recognition

Finally, the AI's ability to recognise what turn it was on was changed to also incorporate PyAutoGui's LocateOnScreen function, this was because the current method still proved unreliable and made testing the AI when there were errors difficult.

Instead of keeping track of the turn based on what the UI told it, the AI started checking for signs that a new turn has begun with the re-emergence of key UI elements that were only present at the during the decision-making portion of each turn.

This stopped the AI from having false starts entirely and came with the extra benefit of being able to restart the AI mid battle.

### 5.5.4 – Testing Outcomes

In this version there were two stages of testing: The first was having the more experienced Participants play against the AI, to see if the AI could beat them in a match and so that they could offer any final tweaks to the calculations based on how it played against them, and performing a Turing Test with Participants of various levels of experience, having them watch the AI play the game against a human player and see if they could recognise who was the human and who was the AI (full results of this test in Appendix E).

For the First test the AI played against a participant that with several years of experience in 3 matches, unfortunately the outcome of all three of these matches ended with the AI's defeat, however this was not attributed to the AI's lack of skill but rather that the teams randomly given to either player were simply unfavourable to the AI on all three occasions, in two of these games the AI came close to winning, however the Opponent's last few Pokémon had an overwhelming advantage against the AI's remaining Pokémon, and in the third match the Opponent was given a set up that was only beatable by very specific circumstances that the AI did not have access to.

In these fights it was suggested that the AI be taught how to recognise sets of moves that could be used in certain combinations that would allow for a more drawn-out victory rather than prioritising quick efficient victories, however due to such the number of possible combinations this was not considered feasible within the given time frame.

Finally, for the last test, five participants of varying experience, ranging from several years in the game to complete beginners, were shown three separate battles of the AI playing against a human player, and they were requested to guess which one of the two was the AI and give a reason for their guess, if the reason was not considered something concrete, and their guess was simply a lucky, it would still be considered as them being unable to discover which player was the AI.

In this test: the players with more experience were capable of discovering the AI with a good reason around two thirds of the time, being able to recognise that certain moves weren't optimal when it came to considering strategies that spanned several turns; players with slight knowledge of the game noticed the AI about a third of the time, as it would sometimes make mistakes such as swapping out a Pokémon on low health when leaving them in to take the hit and sacrificing them for a free swap in would have been a better choice; whilst completely inexperienced players could not give good reason for which one they thought was the AI, likely due the being unable to spot things that they would only learn by playing the game longer.

Overall, the AI was only discovered with good reason five out of fifteen times, 4 out of 6 times for expert players, which can be considered a successful Turing test.

# Chapter 6 – Reflection

## 6.1 – Objectives

The objectives of this project were:

- To build an AI that can make advantageous decisions during a Pokémon battle, To the level where, it can pass a Turing Test
- Training the AI to recognise all existing Pokémon with as close to perfect accuracy as
- To create a computer vision component that could look at the Pokémon Showdown UI and recognise all the key aspects needed for the AI to make its decisions
- Training the AI to decide what to do next based on the information it has obtained
- An Optional objective of making the AI fully automated, giving it the ability to challenge opponents on its own and keep playing even after a game ends.

### 6.1.1 - Main Objective

The main objective was completed successfully. A Turing test was performed on the AI with several participants and based on the results the AI was only recognised a third of the time, much higher than the required 70% of the time to pass a Turing test. This is directly shown in the final testing outputs (shown in appendix E).

There were of course issues with the final version of the project, the AI was not perfect, and struggled to beat skilled players, as well as still failing to imitate a human player at higher levels of skill.

This could be remedied by further refinement of algorithm the AI uses for calculating move value.

### 6.1.2 - Sub Objectives

Of the main three sub-objectives, this project managed to complete two fully and one partially.

The three sub objectives were tested manually and were integral to the Artificial Intelligence's ability to participate in a battle and have a Turing test applied to it.

#### 6.1.2.1 – Sub Objective 1

The Artificial Intelligence could identify all and any Pokémon in its team and on the field with the only issue coming from Pytesseract occasionally being unreliable with its conversion of text to images, completing the first sub objective. This objective was fully completed in version 1 of the project.

This was not a perfect solution, as the AI would still occasionally fail to read the text correctly, and finding the information was dependent on the size of the screen, which makes the AI unusable on certain computers.

#### 6.1.2.1 – Sub Objective 2

The AI could find and interact with every UI element on the screen by using PyAutoGui's LocateOnScreen, making the AI capable of finding these elements regardless of where they appear on the screen. This objective was fully completed in version 4 of the project.

The solution sometimes relied on constantly clicking in certain areas of the screen to prepare for unexpected changes to the UI, which could sometimes cause the UI to freeze interrupting the AI.

#### 6.1.2.1 – Sub Objective 3

Finally, the AI calculated the value of every move, having been altered several times based on participant feedback. However, not all the feed back could be implemented, therefore this objective could not be considered as fully complete.

### 6.1.3 - Optional Objective

The optional objective was not completed to any capacity, this was because it was considered unfeasible in the given time, as the AI was not Optimised for performing more than one battle and navigating the Pokemon Showdown challenging AI was much more difficult than the battle UI.

## 6.2 – What was learnt

The most important thing achieved through out this project was creating an Artificial Intelligence that was good enough to beat someone with my level of experience in a Pokemon Battle, showing that the area of game automation has indeed been growing since the very first game playing AI.

Throughout development, I learnt various techniques to resolve problems surrounding Artificial Intelligence in games, such as giving an AI the ability to look at a UI and pick out the important aspects of a game without having to resort to complex methods such as training a neural network, or how to create a simple evolving algorithm that allows and AI to become better at a game the more it plays.

Despite this solution being much more efficient and elegant than initially planned, it is still not perfect.

One of the literatures discussed early on in this paper, Future Sight AI, performs the same task as my own AI at a much higher level, using much more complex methods, however, Future Sight was built using the source code of Pokemon Showdown, being trained on thousands of games, whilst my own approach created an AI the could play the game with no training at all and still perform at a decently high level, and I feel this suited the time constraints I had much better than using an AI the required a fully trained deep learning neural network.

## 6.3 – Future work

This project has been a very fruitful learning experience, it has significantly improved my skills in the areas of Computer Vision and has inspired me to build Artificial Intelligence Models for other, potentially even more complex games, and seeing just how far I can push these models when under no time constraints.

It has also taught me that I need to improve in several aspects when it comes to large scale projects, such as sticking to schedules and keeping track of data, all skills that will help me greatly in any future work that I undertake.

# References


Bulbapedia. (n.d.). *Pokémon Showdown*. Retrieved from Bulbapedia, the community-driven Pokémon encyclopedia: https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_Showdown#:~:text=Features.%20Pok%C3%A9mon%20Showdown%20was%20the%20first%20simulator%20to,also%20has%20support%20for%20single%20battles%20in%20all

Comi, M. (2021, July 10). *How to teach an AI to play games: Deep Reinforcement Learning*. Retrieved from Towards Data Science: https://towardsdatascience.com/how-to-teach-an-ai-to-play-games-deep-reinforcement-learning-28f9b920440a

IBM. (n.d.). *What is Computer Vision? | IBM*. Retrieved from Ibm.com: https://www.ibm.com/topics/computer-vision

Rosebrock, A. (n.d.). *Building a Pokedex in Python: Getting Started (Step 1 of 6) - PyImageSearch*. Retrieved from PyImageSearch: https://pyimagesearch.com/2014/03/10/building-pokedex-python-getting-started-step-1-6/

The Third Build. (n.d.). *Pokemon Battle Predictor*. Retrieved from Pokemonbattlepredictor.com: https://www.pokemonbattlepredictor.com/home

ThinkAutomation. (2021, February 11). *Game AI: why do we teach AI to play games?* Retrieved from ThinkAutomation: https://www.thinkautomation.com/bots-and-ai/game-ai-why-do-we-teach-ai-to-play-games/

Wikipedia. (2022, February 15). *Machine learning*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Machine_learning

The Third Build. (2021). How an A.I. is Becoming the World's Best Pokémon Player. [Online Video]. 20 July 2021. Available from: https://www.youtube.com/watch?v=rhvj7CmTRkg. [Accessed: 26 January 2022].

Code Bullet. (2022). Creating an A.I. to DESTROY Clicker Heroes. [Online Video]. 18 January 2022. Available from: https://www.youtube.com/watch?v=ldaescGA1dY. [Accessed: 22 December 2021].

# Appendices

## Appendix A: Project Definition Document

# Problem Definition Document

BSc (Hons) Computer Science

Teaching an artificial intelligence to play a complex turn-based strategy game

By: Igor Pereira Martins

Email: Igor.Pereira-Martins@city.ac.uk

Consultant: Dr Andy Macfarlane

This project was proposed by me, with no proprietary interests and there will be no outside help received.

Word count: 1,472

# Problem to be Solved

Since the dawn of artificial intelligence, Computer Scientists have been pitting them against various games, “One of the earliest examples of AI playing games comes from 1951-1952, with a computerised game of Nim.” (ThinkAutomation, 2021).

Although the games started out being quite simple, AI rapidly progressed to harder and harder games, Chess in 1997 with IBM’s Deep Blue achieving victory against the then world champion, Garry Kasparov, the game of Go in 2016 with Google’s AlphaGo, and perhaps more pertinent examples to this project, Marl/O playing a game of Mario in 2015.

With all these examples in mind, it begs the question: with recent advancements in Machine Learning and Data Science, how complex can the game we teach an AI to play be? To answer this question, I have chosen to teach an AI to play a game that falls under the same genre of things that have been done before, but with much more complex features.

For this project I will be teaching an AI to participate in a Pokémon Battle, hopefully achieving a competitive standard of skill, but at minimum having it be able achieve victory against an opponent with the skill of at least the highest level of NPC opponent.

To truly push the limits of my AI, I will be training it to play on Pokémon Showdown, “a web-based Pokémon battle simulator. It is free open-source software and written in JavaScript and Node.js.” (Bulbapedia, n.d.), Specifically Randomised Battles, where your team and moves are randomly chosen for you, but follow several patterns based on move sets that perform best in competitive environments.

There are some existing examples of AI being trained to perform in Pokémon battles, and example of such is Future Sight AI that includes a google extension for battle predictions, the website can be found here: https://www.pokemonbattlepredictor.com/home

# Project Objectives and research questions

| Objective | Testing |
|---|---|
| The main objective of this project will be to build an AI that can make advantageous decisions during a Pokémon battle, To the level where if a Turing Test is applied, observers will not be able to tell the difference between the AI and a human player. | Show various Participants recording of the AI vs a human player and see if they can guess which one is which. |
| The first sub objective will be training the AI to recognise all existing Pokémon with a high degree of accuracy, this will be the basis for a lot of the AI’s decision making, this will likely be done with either a neural network that takes an image and evaluates all its pixels, or with the use of computer vision, likely by finding the edges of the Pokémon as they have very distinct silhouettes | I can simply test this objective by making the AI identify Pokémon and seeing how many it identifies correctly. |

| | |
|---|---|
| The next sub objective will be creating a computer vision programme that can look at the Pokémon Showdown UI and recognise all the key aspects needed for the AI to make its decisions, this includes: the Pokémon the AI has on the field, the Pokémon on the AI's team, the Pokémon the opponent has on the field, and the moves the AI has available. The AI will then have to extract the correct information from what it sees to come to decision for its next move. | I will test that this objective is complete my manually creating scenarios for the AI to look at and then checking if the collected information matches the given information. |
| The final sub objective will be training the AI to decide what to do next based on the information it has obtained, teaching it which decision would be the best choice in various situations. | To test this, I will make the AI battle against various human players, including myself and getting their feedback on whether the AI made the correct choice in any given scenario. |

## Beneficiaries

The first beneficiary of my project will likely be those who want to research the use of computer vision and Machine learning to create artificial intelligence to play or participate in more complex games, as they can use my project as a steppingstone or an informative piece that could help them get started with a project of their own or provide inspiration.

The second beneficiary of my project could be Pokémon players, as, if the AI becomes good enough at the game, it could provide significant challenge to experienced players, allowing them to play the game on their own whilst maintaining the difficulty the game used to provide, as well as testing their teams without needing another player.

The final Beneficiary of my project will be me, not only will this project be graded, but it will also provide me the opportunity to learn and improve various skills in an area I'm interested in working in in the future.

## Work Plan

Week 1 – 2: Complete Proposal and design elements of the documents in preparation for the implementation of the code, performing the research needed, as well as obtaining any necessary data needed for training the AI.

Week 3 - 4: Create the algorithm that recognizes Pokémon, fine tuning it to work specifically with the Pokémon showdown UI, as well as providing the AI with a database it can then use to find useful information, such as the types of the Pokémon, keeping track of all the testing results for the final report, as well as all steps taken to create the algorithm.

Week 5 – 6: Create the Computer vision code necessary for inspecting the elements of the Pokémon Showdown UI and converting the obtained information into data needed to process a decision, such as the type of the Pokémon available on the AI's team and the Pokémon the opponent currently has on the field and deciding whether the matchup is favourable.

Week 7 – 8: Using the algorithms made in the previous weeks, create a behavioural model that teaches the AI to make basic decisions such as using moves that'll maximise damage or switching out

Pokémon based on type effectiveness as well as giving it the ability to interact with the Pokémon showdown UI.

Week 9 – 10: Improve the behavioural model to allow the AI to perform at a much higher level, teaching it to use competitive move sets in the correct way, as well as attempting to predict the next move its opponent will make and try to counter it if necessary.

Week 11 – Due Date: putting the final touches on the project by extensive testing, and putting any finishing touches on the final report.

| Objective | Start Date | End Date | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 | Week 11 | Easter |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Proposal | Week 1 | Week 2 | | | | | | | | | | | | |
| Research | Week 2 | Week 2 | | | | | | | | | | | | |
| Acquire Data | Week 2 | Week 2 | | | | | | | | | | | | |
| Pokémon Classification Algorithm | Week 3 | Week 4 | | | | | | | | | | | | |
| Pokémon Data Search | Week 4 | Week 4 | | | | | | | | | | | | |
| Testing | Week 3 | Week 10 | | | | | | | | | | | | |
| Computer vision code for data collection and analysis | Week 5 | Week 6 | | | | | | | | | | | | |
| Simple behavioural model | Week 7 | Week 8 | | | | | | | | | | | | |
| Complex behavioural model | Week 9 | Week 10 | | | | | | | | | | | | |
| Write Report | Week 3 | Week 11 | | | | | | | | | | | | |
| Consultant meetings | Week 2 | Week 11 | | | | | | | | | | | | |

## Risks

| Objective | Risks | likelihood | Severity | Score | Actions |
|---|---|---|---|---|---|
| Version 1.1 – Use computer vision to recognise a Pokémon | Unfamiliarity with creating computer vision code | 1 | 5 | 5 | Attend my Computer Vision lectures as well as watch online tutorials to fast track any skills I need to finish the code in time. |
| Version 1.2 – Finding a database for the AI to search through | I might not find a database with all the information I need, which would mean I would not be able to train my AI to recognize the right information | 1 | 5 | 5 | Use online resources to make my own database, a lot of the necessary data is well documented and public. |
| Version 2.1 – using the Pokémon Showdown UI to help train the AI | The UI may make things complicated for computer vision, e.g., transparent elements or changing pixel colours | 2 | 4 | 8 | Study the UI to figure out any problems ahead of time, and figure out exactly how I will use computer vision to recognize certain elements |
| Version 3.1 – Teach the AI to make decisions | The AI might reach the right goals in the wrong way, for example, the goal might be picking the best move and the AI may always pick an attacking move to finish the current | 3 | 3 | 9 | Carefully observe the AI in action as well as enlist several participants to test the AI and catch any problems I may miss. |

| | | | | | |
|---|---|---|---|---|---|
| | fight earlier, but this might cost it the whole battle | | | | |
| Version 3.2 – Improve AI decision making | The Scale of the project at this point may be too big for me to finish in time. | 1 | 5 | 5 | Consider reducing the decisions the AI can make, instead making it only provide suggestions for your next move. |
| Version 3.3 – Test AI against human players | I must make sure that all tests are ethical involving willing participants | 1 | 5 | 5 | Enlist participants early on and make them fill in all the necessary forms. |
| Version 4 – Improve the behavioural model to a competitive level | The hardware I have may not be able to handle the increase load of the new neural network | 3 | 3 | 9 | Allocate extra time to make sure there is enough for training, in the worst-case decrease complexity |

## Research Ethics Review Form:

| | | |
|---|---|---|
| **A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/** | | *Delete as appropriate* |
| 1.1 | Does your research require approval from the National Research Ethics Service (NRES)?<br>*e.g. because you are recruiting current NHS patients or staff?*<br>*If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/* | **NO** |
| 1.2 | Will you recruit participants who fall under the auspices of the Mental Capacity Act?<br>*Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/* | **NO** |
| 1.3 | Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation?<br>*Such research needs to be authorised by the ethics approval system of the National Offender Management Service.* | **NO** |
| **A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/** | | *Delete as appropriate* |
| 2.1 | Does your research involve participants who are unable to give informed consent?<br>*For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.* | **NO** |

| | | |
|---|---|---|
| 2.2 | Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities? | **NO** |
| 2.3 | Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)? | **NO** |
| 2.4 | Does your project involve participants disclosing information about special category or sensitive subjects?<br>*For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings* | **NO** |
| 2.5 | Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study?<br>*Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/* | **NO** |
| 2.6 | Does your research involve invasive or intrusive procedures?<br>*These may include, but are not limited to, electrical stimulation, heat, cold or bruising.* | **NO** |
| 2.7 | Does your research involve animals? | **NO** |
| 2.8 | Does your research involve the administration of drugs, placebos or other substances to study participants? | **NO** |
| **A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/**<br>**Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.** | | *Delete as appropriate* |
| 3.1 | Does your research involve participants who are under the age of 18? | **NO** |
| 3.2 | Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)?<br>*This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.* | **NO** |
| 3.3 | Are participants recruited because they are staff or students of City, University of London?<br>*For example, students studying on a particular course or module.*<br>*If yes, then approval is also required from the Head of Department or Programme Director.* | **NO** |
| 3.4 | Does your research involve intentional deception of participants? | **NO** |
| 3.5 | Does your research involve participants taking part without their informed consent? | **NO** |

| | | |
|---|---|---|
| 3.5 | Is the risk posed to participants greater than that in normal working life? | **NO** |
| 3.7 | Is the risk posed to you, the researcher(s), greater than that in normal working life? | **NO** |
| **A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK.**<br>**If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.**<br>**If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.** | | *Delete as appropriate* |
| 4 | Does your project involve human participants or their identifiable personal data?<br>*For example, as interviewees, respondents to a survey or participants in testing.* | **YES** |

# PART B: Ethics Proportionate Review Form

| **B.1 The following questions must be answered fully.**<br>**All grey instructions must be removed.** | | *Delete as appropriate* |
|---|---|---|
| 1.1. | Will you ensure that participants taking part in your project are fully informed about the purpose of the research? | **YES** |
| 1.2 | Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept? | **YES** |
| 1.3 | When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty? | **YES** |
| 1.4 | Will consent be obtained from the participants in your project?<br>Consent from participants will be necessary if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc.<br>*If YES, you must attach drafts of the participant information sheet(s) and consent form(s) that you will use in section B.3 or, in the case of an existing dataset, provide details of how consent has been obtained.*<br>*You must also retain the completed forms for subsequent inspection. Failure to provide the completed consent request forms will result in withdrawal of any earlier ethical approval of your project.* | **YES** |
| 1.5 | Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential? | **YES** |

| **B.2 If the answer to the following question (B2) is YES, you must provide details** | | *Delete as appropriate* |
|---|---|---|
| 2 | Will the research be conducted in the participant's home or other non-University location?<br>*If **YES**, you must provide details of how your safety will be ensured.* | **NO** |

| **B.3 Attachments**<br>**ALL of the following documents MUST be provided to supervisors if applicable.**<br>**All must be considered prior to final approval by supervisors.**<br>**A written record of final approval must be provided and retained.** | ***YES*** | ***NO*** | ***Not Applicable*** |
|---|---|---|---|
| Details on how safety will be assured in any non-University location, including risk assessment if required (see B2) | | | |
| Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5)<br>*Any personal data must be acquired, stored and made accessible in ways that are GDPR compliant.* | | | |
| Full protocol for any workshops or interviews** | | | |
| Participant information sheet(s)** | | | |
| Consent form(s)** | | | |
| Questionnaire(s)**<br>*sharing a Qualtrics survey with your supervisor is recommended.* | | | |
| Topic guide(s) for interviews and focus groups** | | | |
| Permission from external organisations or Head of Department**<br>*e.g. for recruitment of participants* | | | |

# References

Bulbapedia. (n.d.). Pokémon Showdown. Retrieved from Bulbapedia, the community-driven Pokémon encyclopedia: https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_Showdown#:~:text=Features.%20Pok%C3%A9mon%20Showdown%20was%20the%20first%20simulator%20to,also%20has%20support%20for%20single%20battles%20in%20all

ThinkAutomation. (2021, February 11). Game AI: why do we teach AI to play games? Retrieved from ThinkAutomation: https://www.thinkautomation.com/bots-and-ai/game-ai-why-do-we-teach-ai-to-play-games/

# Appendix B: Reuse Summary

Python pillow library – found here: https://python-pillow.org/

Python pytesseract library – download found here: https://pypi.org/project/pytesseract/#files
pyTesseract executable found in Moodle extra submissions

Python Pyautogui library – download found here:
https://pyautogui.readthedocs.io/en/latest/install.html

Python Keyboard library – download found here: https://pypi.org/project/keyboard/

Python pandas library – download found here: https://pandas.pydata.org/

Python difflib library – download found here: https://docs.python.org/3/library/difflib.html

Python numpy library – download found here: https://numpy.org/install/

# Appendix C: Requirements

| | |
|---|---|
| **Requirement #** | 001 |
| **Description** | Find information on screen |
| **Rationale** | For the AI to operate at all it needs to be able to find where the various UI elements are on the screen and move the mouse to these locations. |
| **Fit Criterion** | This requirement will be successful if the AI can move the cursor to the correct UI element when needed. |
| **Priority** | Very high |

| | |
|---|---|
| **Requirement #** | 002 |
| **Description** | Identify the information on screen |
| **Rationale** | For the AI to calculate its next move it needed to be able to convert any information on screen into a useable format |
| **Fit Criterion** | This requirement will be successful if the AI can convert any information it sees on screen into a useable format to use in its calculations |
| **Priority** | Very high |

| | |
|---|---|
| **Requirement #** | 003 |
| **Description** | Calculate the value of possible decisions |
| **Rationale** | For the AI to play the game it needs to be able use available information to calculate its next move. |
| **Fit Criterion** | This requirement will be successful when the AI can perform simple calculations based on provided information. |
| **Priority** | Very high |

| | |
|---|---|
| **Requirement #** | 004 |
| **Description** | Perform the decided move |
| **Rationale** | For the AI to play the game it needs to be able to perform the decisions it makes based on its calculations |
| **Fit Criterion** | This requirement will be successful when the AI can perform the correct move based on its calculations |
| **Priority** | Very High |

| | |
|---|---|
| **Requirement #** | 005 |
| **Description** | Recognise when a turn is over |
| **Rationale** | For the AI to play the game it needs to be able to understand when a new turn begins so it can start calculating it's moves again |
| **Fit Criterion** | This requirement will be successful when the AI can recognise the beginning of a new turn |
| **Priority** | Very High |

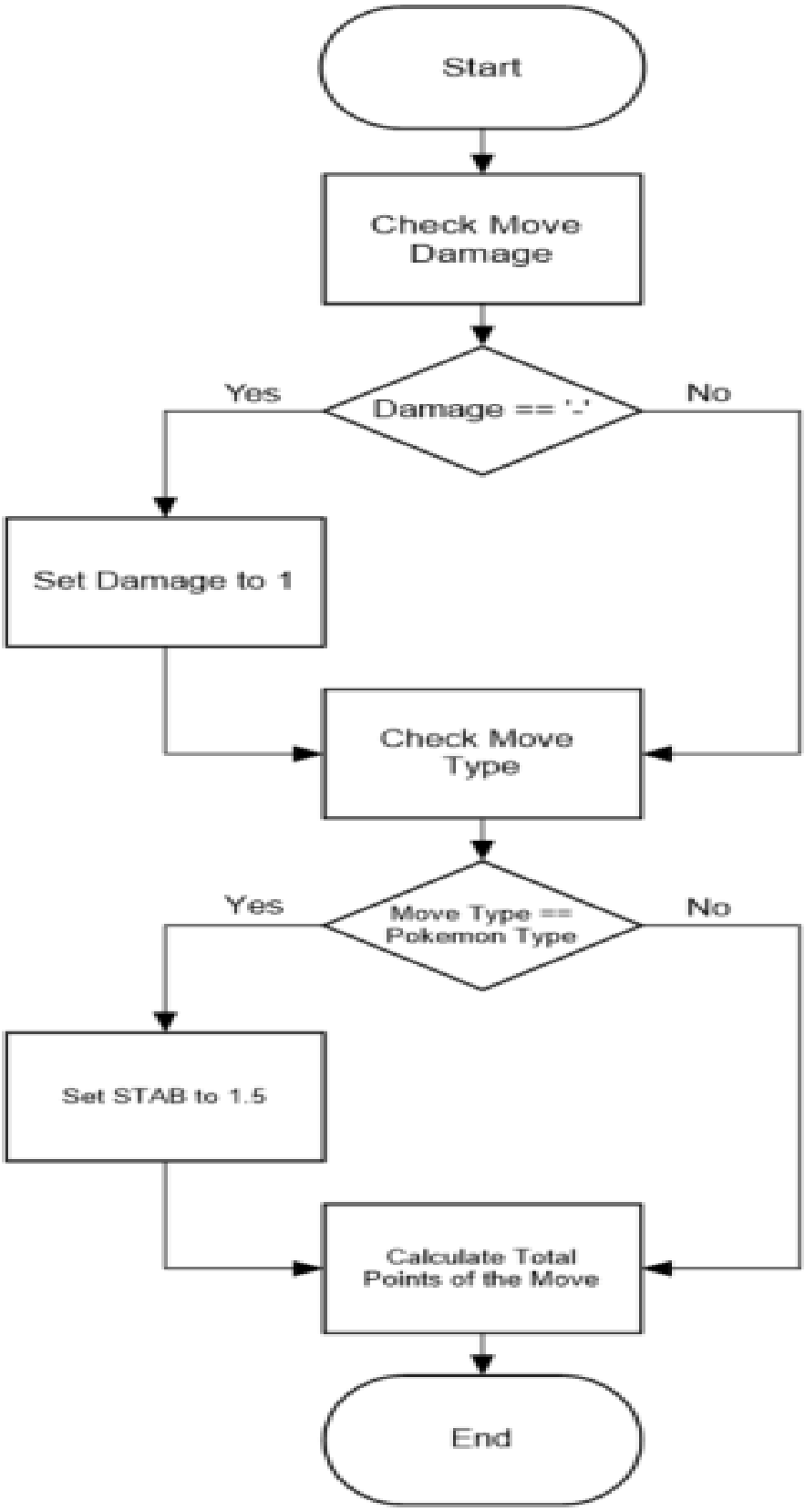
| | |
|---|---|
| **Requirement #** | 006 |
| **Description** | Perform Complex Calculations to decide moves |
| **Rationale** | For the AI to reach a high skill level it needs to consider various variables that would make calculations more complex |
| **Fit Criterion** | This requirement will be successful when the AI can perform calculations that take into consideration complex variables that are not required for basic functionality |
| **Priority** | Moderate |

| | |
|---|---|
| **Requirement #** | 007 |
| **Description** | Take into consideration the Opponent's possible moves |
| **Rationale** | For the AI to reach the highest skill level it needs to consider the possible moves its opponent could take on that turn |
| **Fit Criterion** | This requirement will be successful when the AI can calculate the probability of what moves the Opponent may have without having seen those moves |
| **Priority** | Low |

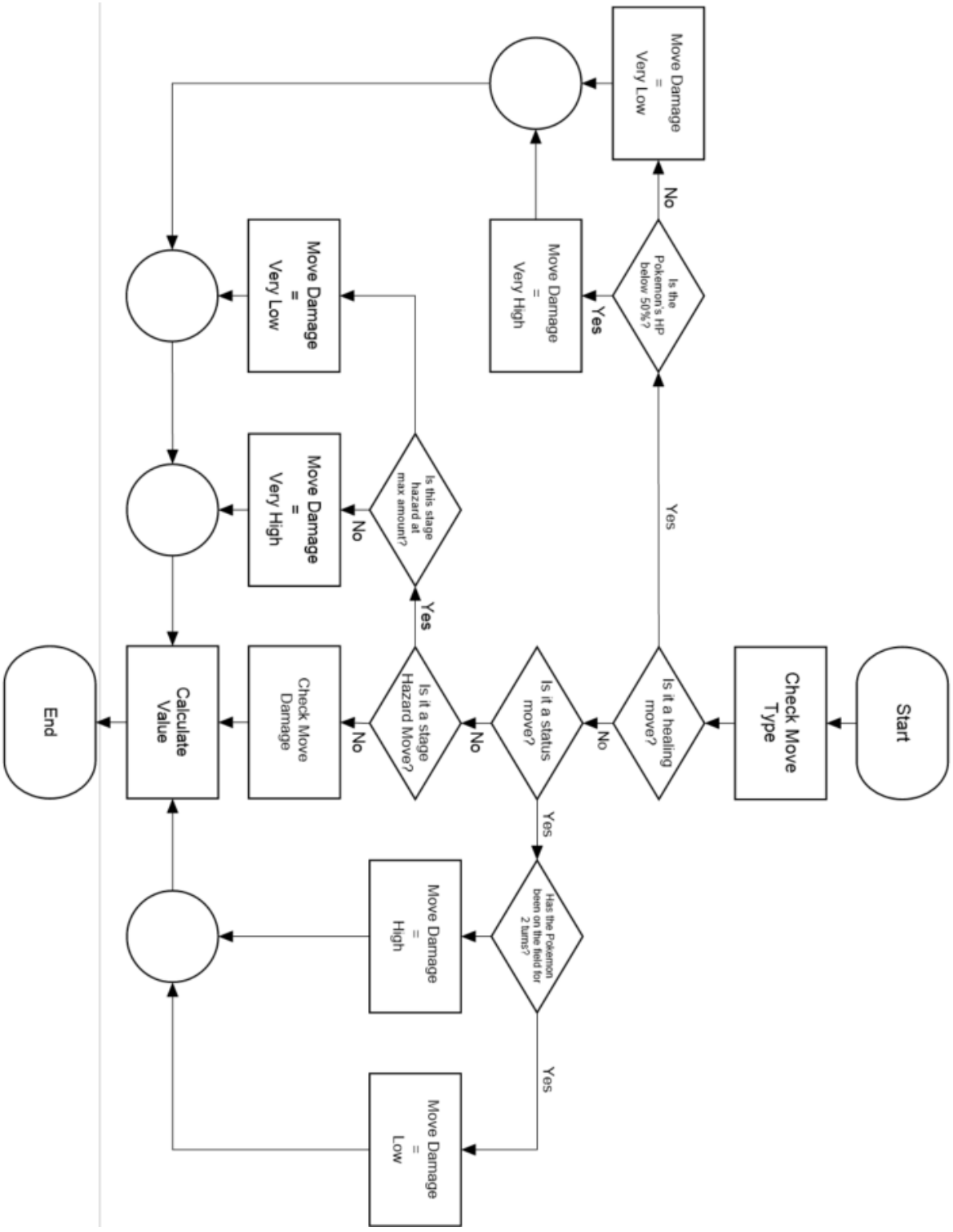
| | |
|---|---|
| **Requirement #** | 008 |
| **Description** | Perform the decisions whilst taking future turns into account |
| **Rationale** | For the AI to reach the highest skill level it needs to consider how it's decisions may impact future turns |
| **Fit Criterion** | This requirement will be successful when the AI can make decisions that consider long term effects such as damage over time or status increases instead of just trying to maximise damage for that turn |
| **Priority** | Low |

# Appendix D: Design Documents

## Decision Tree for Simple Move Selection



## Decision Tree for Complex Move Selection



# Appendix E: Test Plans and Results

## Version 1 Test Template

| Test No. | Correct Value | Given Value | Correct? |
|---|---|---|---|
| 1 | | | |

### Version 1 Results – Test 1

| No. | Correct Value | Given Value | Correct? |
|---|---|---|---|
| 1 | ['Mr. Mime', ' Dazzling Gleam ', 'Focus Blast', 'Nasty Plot', 'Psychic'] | ['Mr.', ' Dazzling Gleam ', 'Focus Blast', 'Nasty Plot', 'Psychic'] | |
| 2 | ['Electivire', 'Earthquake', 'Flamethrower', 'Wild Charge', 'Ice Punch'] | ['Electivire', 'Earthquake', 'Flamethrower', 'Wild Charge', '  '] | |
| 3 | ['Perrserker', 'Crunch', 'Iron Head', 'U-turn', 'Close Combat'] | ['Perrserker', '¢ Crunch', '¢ Iron Head', 'e U-turn', 'e Close Combat'] | |
| 4 | ['Keldeo', 'Calm Mind', 'Substitute', 'Scald', 'Secret Sword'] | ['Keldeo', 'Calm Mind', 'Substitute', 'Scald', 'Secret Sword'] | |
| 5 | ['Blissey', 'Seismic Toss', 'Soft-Boiled', 'Teleport', 'Toxic'] | ['Blissey', 'Seismic Toss', 'Soft-Boiled', 'Teleport', 'Toxic'] | |
| 6 | ['Vanilluxe', 'Aurora Veil', 'Flash Cannon', 'Freeze-Dry', 'Blizzard'] | ['Vanilluxe', 'Aurora Veil', 'Flash Cannon', 'Freeze-Dry', 'Blizzard'] | |
| 7 | ['Heatran', 'Protect', 'Flash Cannon', 'Toxic', 'Lava Plume'] | ['Heatran', ' ', ' ', ' ', ' '] | |
| 8 | ['Manectric', 'Thunderbolt', 'Overheat', 'Volt Switch', 'Switcheroo'] | ['Manectric', 'Thunderbolt', 'Overheat', 'Volt Switch', 'Switcheroo'] | |
| 9 | ['Tangrowth', 'Earthquake', 'Leech Seed', 'Sludge Bomb', 'Giga Drain'] | ['Tangrowth', 'Earthquake', 'Leech Seed', 'Sludge Bomb', 'Giga Drain'] | |
| 10 | ['Kyurem', 'Draco Meteor', 'Substitute', 'Roost', 'Freeze-Dry'] | ['Kyurem', 'Substitute', 'Roost', 'Freeze-Dry', ' '] | |
| 11 | ['Druddigon', 'Earthquake', 'Outrage', 'Sucker Punch', 'Stealth Rock'] | ['Druddigon', 'Earthquake', 'Outrage', 'Sucker Punch', 'Stealth Rock'] | |
| 12 | ['Wishiwashi', 'Scald', 'Ice Beam', 'Earthquake', 'U-turn'] | ['Wishiwashi', 'Scald', 'Ice Beam', 'Earthquake', 'U-turn'] | |
| 13 | ['Linoone', 'Belly Drum', 'Extreme Speed', 'Stomping Tantrum', 'Throat Chop'] | ['Linoone', 'Belly Drum', 'Extreme Speed', 'Stomping Tantrum', 'Throat Chop'] | |
| 14 | ['Hydreigon', 'Roost', 'Draco Meteor', 'Flash Cannon', 'Dark Pulse'] | ['Hydreigon', 'Roost', 'Draco Meteor', 'Flash Cannon', 'Dark Pulse'] | |
| 15 | ['Marowak (Marowak-Alola)', 'Flame Charge', 'Flare Blitz', 'Poltergeist', 'Stone Edge'] | ['Marowak', 'Flame Charge', 'Flare Blitz', 'Poltergeist', 'Stone Edge'] | |
| 16 | ['Gardevoir', 'Substitute', 'Psyshock', 'Mystical Fire', 'Moonblast'] | ['HP:', 'Ability: Trace / Item: Leftovers', 'Psyshock', 'Mystical Fire', 'Moonblast'] | |
| 17 | ['Rayquaza', 'Dragon Ascent', '* Swords Dance', 'V-create', 'Extreme Speed'] | ['Rayquaza', 'Dragon Ascent', '* Swords Dance', 'V-create', 'Extreme Speed'] | |
| 18 | ['Golisopod', 'Leech Life', 'First Impression', 'Knock Off', 'Liquidation'] | ['Golisopod', 'Leech Life', 'First Impression', 'Knock Off', 'Liquidation'] | |

## Version 1 Results – Test 2

| No. | Correct Value | Given Value | Correct? |
|---|---|---|---|
| **1** | ['Mr. Mime', ' Dazzling Gleam ', 'Focus Blast', 'Nasty Plot', 'Psychic'] | ['Mr.', ' Dazzling Gleam ', 'Focus Blast', 'Nasty Plot', 'Psychic'] | |
| **2** | ['Electivire', 'Earthquake', 'Flamethrower', 'Wild Charge', 'Ice Punch'] | ['Electivire', 'Earthquake', 'Flamethrower', 'Wild Charge', 'Ice Punch'] | |
| **3** | ['Perrserker', 'Crunch', 'Iron Head', 'U-turn', 'Close Combat'] | ['Perrserker', '¢ Crunch', '¢ Iron Head', 'e U-turn', 'e Close Combat'] | |
| **4** | ['Keldeo', 'Calm Mind', 'Substitute', 'Scald', 'Secret Sword'] | ['Keldeo', 'Calm Mind', 'Substitute', 'Scald', 'Secret Sword'] | |
| **5** | ['Blissey', 'Seismic Toss', 'Soft-Boiled', 'Teleport', 'Toxic'] | ['Blissey', 'Seismic Toss', 'Soft-Boiled', 'Teleport', 'Toxic'] | |
| **6** | ['Vanilluxe', 'Aurora Veil', 'Flash Cannon', 'Freeze-Dry', 'Blizzard'] | ['Vanilluxe', 'Aurora Veil', 'Flash Cannon', 'Freeze-Dry', 'Blizzard'] | |
| **7** | ['Heatran', 'Protect', 'Flash Cannon', 'Toxic', 'Lava Plume'] | ['Heatran', 'Protect', 'Flash Cannon', 'Toxic', 'Lava Plume'] | |
| **8** | ['Manectric', 'Thunderbolt', 'Overheat', 'Volt Switch', 'Switcheroo'] | ['Manectric', 'Thunderbolt', 'Overheat', 'Volt Switch', 'Switcheroo'] | |
| **9** | ['Tangrowth', 'Earthquake', 'Leech Seed', 'Sludge Bomb', 'Giga Drain'] | ['Tangrowth', 'Earthquake', 'Leech Seed', 'Sludge Bomb', 'Giga Drain'] | |
| **10** | ['Kyurem', 'Draco Meteor', 'Substitute', 'Roost', 'Freeze-Dry'] | ['Kyurem', 'Draco Meteor', 'Substitute', 'Roost', 'Freeze-Dry'] | |
| **11** | ['Druddigon', 'Earthquake', 'Outrage', 'Sucker Punch', 'Stealth Rock'] | ['Druddigon', 'Earthquake', 'Outrage', 'Sucker Punch', 'Stealth Rock'] | |
| **12** | ['Wishiwashi', 'Scald', 'Ice Beam', 'Earthquake', 'U-turn'] | ['Wishiwashi', 'Scald', 'Ice Beam', 'Earthquake', 'U-turn'] | |
| **13** | ['Linoone', 'Belly Drum', 'Extreme Speed', 'Stomping Tantrum', 'Throat Chop'] | ['Linoone', 'Belly Drum', 'Extreme Speed', 'Stomping Tantrum', 'Throat Chop'] | |
| **14** | ['Hydreigon', 'Roost', 'Draco Meteor', 'Flash Cannon', 'Dark Pulse'] | ['Hydreigon', 'Roost', 'Draco Meteor', 'Flash Cannon', 'Dark Pulse'] | |
| **15** | ['Marowak (Marowak-Alola)', 'Flame Charge', 'Flare Blitz', 'Poltergeist', 'Stone Edge'] | ['Marowak', 'Flame Charge', 'Flare Blitz', 'Poltergeist', 'Stone Edge'] | |
| **16** | ['Gardevoir', 'Substitute', 'Psyshock', 'Mystical Fire', 'Moonblast'] | ['Gardevoir', 'Substitute', 'Psyshock', 'Mystical Fire', 'Moonblast'] | |
| **17** | ['Rayquaza', 'Dragon Ascent', '* Swords Dance', 'V-create', 'Extreme Speed'] | ['Rayquaza', 'Dragon Ascent', '* Swords Dance', 'V-create', 'Extreme Speed'] | |
| **18** | ['Golisopod', 'Leech Life', 'First Impression', 'Knock Off', 'Liquidation'] | ['Golisopod', 'Leech Life', 'First Impression', 'Knock Off', 'Liquidation'] | |

## Version 1 Results – Test 3

| No. | Correct Value | Given Value | Correct? |
|---|---|---|---|
| **1** | ['Mr. Mime', ' Dazzling Gleam ', 'Focus Blast', 'Nasty Plot', 'Psychic'] | ['Mr. Mime', ' Dazzling Gleam ', 'Focus Blast', 'Nasty Plot', 'Psychic'] | |
| **2** | ['Electivire', 'Earthquake', 'Flamethrower', 'Wild Charge', 'Ice Punch'] | ['Electivire', 'Earthquake', 'Flamethrower', 'Wild Charge', 'Ice Punch'] | |
| **3** | ['Perrserker', 'Crunch', 'Iron Head', 'U-turn', 'Close Combat'] | ['Perrserker', '¢ Crunch', '¢ Iron Head', 'e U-turn', 'e Close Combat'] | |
| **4** | ['Keldeo', 'Calm Mind', 'Substitute', 'Scald', 'Secret Sword'] | ['Keldeo', 'Calm Mind', 'Substitute', 'Scald', 'Secret Sword'] | |
| **5** | ['Blissey', 'Seismic Toss', 'Soft-Boiled', 'Teleport', 'Toxic'] | ['Blissey', 'Seismic Toss', 'Soft-Boiled', 'Teleport', 'Toxic'] | |
| **6** | ['Vanilluxe', 'Aurora Veil', 'Flash Cannon', 'Freeze-Dry', 'Blizzard'] | ['Vanilluxe', 'Aurora Veil', 'Flash Cannon', 'Freeze-Dry', 'Blizzard'] | |
| **7** | ['Heatran', 'Protect', 'Flash Cannon', 'Toxic', 'Lava Plume'] | ['Heatran', 'Protect', 'Flash Cannon', 'Toxic', 'Lava Plume'] | |
| **8** | ['Manectric', 'Thunderbolt', 'Overheat', 'Volt Switch', 'Switcheroo'] | ['Manectric', 'Thunderbolt', 'Overheat', 'Volt Switch', 'Switcheroo'] | |
| **9** | ['Tangrowth', 'Earthquake', 'Leech Seed', 'Sludge Bomb', 'Giga Drain'] | ['Tangrowth', 'Earthquake', 'Leech Seed', 'Sludge Bomb', 'Giga Drain'] | |
| **10** | ['Kyurem', 'Draco Meteor', 'Substitute', 'Roost', 'Freeze-Dry'] | ['Kyurem', 'Draco Meteor', 'Substitute', 'Roost', 'Freeze-Dry'] | |
| **11** | ['Druddigon', 'Earthquake', 'Outrage', 'Sucker Punch', 'Stealth Rock'] | ['Druddigon', 'Earthquake', 'Outrage', 'Sucker Punch', 'Stealth Rock'] | |
| **12** | ['Wishiwashi', 'Scald', 'Ice Beam', 'Earthquake', 'U-turn'] | ['Wishiwashi', 'Scald', 'Ice Beam', 'Earthquake', 'U-turn'] | |
| **13** | ['Linoone', 'Belly Drum', 'Extreme Speed', 'Stomping Tantrum', 'Throat Chop'] | ['Linoone', 'Belly Drum', 'Extreme Speed', 'Stomping Tantrum', 'Throat Chop'] | |
| **14** | ['Hydreigon', 'Roost', 'Draco Meteor', 'Flash Cannon', 'Dark Pulse'] | ['Hydreigon', 'Roost', 'Draco Meteor', 'Flash Cannon', 'Dark Pulse'] | |
| **15** | ['Marowak (Marowak-Alola)', 'Flame Charge', 'Flare Blitz', 'Poltergeist', 'Stone Edge'] | ['Marowak (Marowak-Alola)', 'Flame Charge', 'Flare Blitz', 'Poltergeist', 'Stone Edge'] | |
| **16** | ['Gardevoir', 'Substitute', 'Psyshock', 'Mystical Fire', 'Moonblast'] | ['Gardevoir', 'Substitute', 'Psyshock', 'Mystical Fire', 'Moonblast'] | |
| **17** | ['Rayquaza', 'Dragon Ascent', 'Swords Dance', 'V-create', 'Extreme Speed'] | ['Rayquaza', 'Dragon Ascent', 'Swords Dance', 'V-create', 'Extreme Speed'] | |
| **18** | ['Golisopod', 'Leech Life', 'First Impression', 'Knock Off', 'Liquidation'] | ['Golisopod', 'Leech Life', 'First Impression', 'Knock Off', 'Liquidation'] | |

## Version 2 - Test Plan

| **Battle Number:** | | | |
|---|---|---|---|
| **Opponent:** | | | |
| **Pokémon** | **Correct Value** | **Given Value** | **Correct?** |
| | | | |

## Version 2 - Results

| **Battle Number: 1** | | | |
|---|---|---|---|
| **Opponent: Magmortar** | | | |
| **Pokémon** | Correct Value | Given Value | Correct? |
| **Umbreon** | Wish: 1.0 | Wish: 1.0 | |
| | Foul Play: 95.0 | Foul Play: 95.0 | |
| | Toxic: 1.0 | Toxic: 1.0 | |
| | Protect: 1.0 | Protect: 1.0 | |
| | Best: 95.0 | Best: 95.0 | |
| **Groudon** | Stone Edge: 100.0 | Stone Edge: 100.0 | |
| | Swords Dance: 0.5 | Swords Dance: 0.5 | |
| | Heavy Slam: 0.25 | Heavy Slam: 0.25 | |
| | Precipice Blades: 120.0 | Precipice Blades: 120.0 | |
| | Best: 120.0 | Best: 120.0 | |
| **Mimikyu** | Shadow Claw: 35.0 | Shadow Claw: 35.0 | |
| | Swords Dance: 0.5 | Swords Dance: 0.5 | |
| | Drain Punch: 37.5 | Drain Punch: 37.5 | |
| | Play Rough: 33.75 | Play Rough: 33.75 | |
| | Best: 37.5 | Best: 37.5 | |
| **Exeggutor (Exeggutor-Alola)** | Trick Room: 0.5 | Trick Room: 0.5 | |
| | Giga Drain: 18.75 | Giga Drain: 18.75 | |
| | Draco Meteor: 97.5 | Draco Meteor: 97.5 | |
| | Flamethrower: 22.5 | Flamethrower: 22.5 | |
| | Best: 97.5 | Best: 97.5 | |
| **Scolipede** | Protect: 0.5 | Protect: 0.5 | |
| | Megahorn: 30.0 | Megahorn: 30.0 | |
| | Toxic Spikes: 90.0 | Toxic Spikes: 90.0 | |
| | Earthquake: 100.0 | Earthquake: 100.0 | |
| | Best: 100.0 | Best: 100.0 | |
| **Rhydon** | Toxic: 0.5 | Toxic: 0.5 | |
| | Stone Edge: 150.0 | Stone Edge: 150.0 | |
| | Megahorn: 30.0 | Megahorn: 30.0 | |
| | Earthquake: 100.0 | Earthquake: 100.0 | |
| | Best: 150.0 | Best: 150.0 | |

| **Battle Number: 2** | | | |
|---|---|---|---|
| **Opponent: Aegislash** | | | |
| **Pokémon** | Correct Value | Given Value | Correct? |
| **Accelgor** | Focus Blast: 0.0 | Focus Blast: 0.0 | |
| | Sludge Bomb: 0.0 | Sludge Bomb: 0.0 | |
| | Toxic: 0.0 | Toxic: 0.0 | |
| | Bug Buzz: 22.5 | Bug Buzz: 22.5 | |
| | Best: 22.5 | Best: 22.5 | |
| **Whimsicott** | Soft-Boiled: 0.0 | Soft-Boiled: 0.0 | |
| | Toxic: 0.0 | Toxic: 0.0 | |
| | Teleport: 0.25 | Teleport: 0.25 | |
| | Seismic Toss: 0.0 | Seismic Toss: 0.0 | |
| | Best: 0.25 | Best: 0.25 | |
| **Blissey** | Shadow Claw: 35.0 | Shadow Claw: 35.0 | |
| | Swords Dance: 0.5 | Swords Dance: 0.5 | |
| | Drain Punch: 37.5 | Drain Punch: 37.5 | |
| | Play Rough: 33.75 | Play Rough: 33.75 | |
| | Best: 37.5 | Best: 37.5 | |
| **Slurpuff** | Belly Drum: 0.0 | Belly Drum: 0.0 | |
| | Facade: 0.0 | Facade: 0.0 | |
| | Drain Punch: 0.0 | Drain Punch: 0.0 | |
| | Play Rough: 22.5 | Play Rough: 22.5 | |
| | Best: 22.5 | Best: 22.5 | |
| **Decidueye** | Roost: 0.25 | Roost: 0.25 | |
| | Leaf Blade: 22.5 | Leaf Blade: 22.5 | |
| | Swords Dance: 0.0 | Swords Dance: 0.0 | |
| | Poltergeist: 165.0 | Poltergeist: 165.0 | |
| | Best: 165.0 | Best: 165.0 | |
| **Salamence** | Dual Wingbeat: 15.0 | Dual Wingbeat: 15.0 | |
| | Dragon Dance: 0.25 | Dragon Dance: 0.25 | |
| | Outrage: 30.0 | Outrage: 30.0 | |
| | Earthquake: 100.0 | Earthquake: 100.0 | |
| | Best: 100.0 | Best: 100.0 | |

| **Battle Number: 1** | | | |
|---|---|---|---|
| **Opponent: Yveltal** | | | |
| **Pokémon** | Correct Value | Given Value | Correct? |
| **Mimikyu** | Shadow Sneak: 20.0 | Shadow Sneak: 20.0 | |
| | Shadow Claw: 35.0 | Shadow Claw: 35.0 | |
| | Swords Dance: 1.0 | Swords Dance: 1.0 | |
| | Play Rough: 270.0 | Play Rough: 270.0 | |
| | Best: 270.0 | Best: 270.0 | |
| **Regigigas** | Toxic: 0.5 | Toxic: 0.5 | |
| | Protect: 0.5 | Protect: 0.5 | |
| | Body Slam: 42.5 | Body Slam: 42.5 | |
| | Substitute: 0.5 | Substitute: 0.5 | |
| | Best: 42.5 | Best: 42.5 | |
| **Dracozolt** | Low Kick: 0.5 | Low Kick: 0.5 | |
| | Earthquake: 0.0 | Earthquake: 0.0 | |
| | Bolt Beak: 85.0 | Bolt Beak: 85.0 | |
| | Outrage: 90.0 | Outrage: 90.0 | |
| | Best: 90.0 | Best: 90.0 | |
| **Sigilyph** | Air Slash: 56.25 | Air Slash: 56.25 | |
| | Defog: 0.75 | Defog: 0.75 | |
| | Heat Wave: 47.5 | Heat Wave: 47.5 | |
| | Energy Ball: 22.5 | Energy Ball: 22.5 | |
| | Best: 56.25 | Best: 56.25 | |
| **Togedemaru** | Wish: 0.5 | Wish: 0.5 | |
| | Zing Zap: 80.0 | Zing Zap: 80.0 | |
| | Spiky Shield: 0.25 | Spiky Shield: 0.25 | |
| | Iron Head: 60.0 | Iron Head: 60.0 | |
| | Best: 80.0 | Best: 80.0 | |
| **Drifblim** | Thunderbolt: 90.0 | Thunderbolt: 90.0 | |
| | Shadow Ball: 20.0 | Shadow Ball: 20.0 | |
| | Calm Mind: 0.0 | Calm Mind: 0.0 | |
| | Strength Sap: 0.25 | Strength Sap: 0.25 | |
| | Best: 90.0 | Best: 90.0 | |

## Version 4 – Test Template

| Participant | Test Number | Correct Guess | Reason | Reason Valid |
|---|---|---|---|---|
| | | | | |

| Participant | Test Number | Correct Guess | Reason | Reason Valid |
|---|---|---|---|---|
| **Beginner Player 1** | 1 | | Due to the AI losing the battle | |
| | 2 | | Could not give a reason | |
| | 3 | | Could not give a reason | |
| **Beginner Player 2** | 1 | | Player's final Pokémon was very strong against the remaining AI Pokémon, believed this was the AI as it had not seen the other Pokémon so would not use it. | |
| | 2 | | Could not give a reason | |
| | 3 | | Believed the AI used an ineffective move at the wrong time, when it was the Player due to human error. | |
| **Intermediate Player 1** | 1 | | Could not give a reason | |
| | 2 | | AI did not consider lowered stats and kept using the same move despite this | |
| | 3 | | Player miscalculated the damage they would take, thought this was the AI. | |
| **Expert Player 1** | 1 | | Player did not swap out a Pokémon that was at a disadvantage where the participant thought that the AI would | |
| | 2 | | AI did not consider lowered stats and kept using the same move despite this | |
| | 3 | | Could not give a reason | |
| **Expert Player 2** | 1 | | AI used a move that was considered useless at the final turn of the battle | |
| | 2 | | AI did not consider lowered stats and kept using the same move despite this | |
| | 3 | | Believed the AI used an ineffective move at the wrong time, when it was the Player due to human error. | |

# Appendix F: Source Code

To create an Executable version, put this code in the same folder as the datasets and the PyTesseract executable provided in the Moodle extra submissions area, load up a match at https://play.pokemonshowdown.com/, and once the switch and attack buttons have appeared press the '#' to start the AI.

## Code

```
# Import the necessary libraries
from PIL import Image
import pytesseract
import pyautogui
import keyboard
import pandas as pd
from difflib import SequenceMatcher
import time
from os import listdir
import numpy as np

# Function to use difflib's SequenceMatcher to compare strings
def similar(a, b):
    return SequenceMatcher(None, a, b).ratio()

# Function to use pytesseract to convert image to text
def imgConvert(image):
    result = pytesseract.image_to_string(image, lang='eng')
    arr = result.split('\n')[0:-1]
    return arr

# Function that reduces and image only to it's darker pixels,
# removing as much bakcground noise as possible to increase the
# reliability of pyTesseract
def reduceImage(image, intensity):
    image = np.array(image)
    height, width, channel = image.shape
    for x in range(width):
        for y in range(height):
            if np.sum(image[y][x]) > intensity:
                image[y][x] = np.array([255, 255, 255])

    image = Image.fromarray(image)
    return image

# Function that takes in text converted from images and removes noise
def arrClean(arr):
    newArr = []
    # removes blank elements and spaces from the input array
    for i in arr:
        if str.isspace(i):
            arr.remove(i)
        if i == '':
            arr.remove(i)
```

```
# necessary to catch any escaped blanks unknown why the previos lines
# do not catch these
for i in arr:
    if i == '':
        arr.remove(i)

# checks for exceptions in the Pokemon's name before removing noise
if ' '.join(arr[0].split(" ", 2)[:2]) in SpaceNames:
    arr[0] = ' '.join(arr[0].split(" ", 2)[:2])
elif 'Alola' in arr[0] or 'Galar' in arr[0]:
    arr[0] = arr[0].split(')', 1)[0] + ")"
else:
    arr[0] = arr[0].split(' ', 1)[0]

print(arr)
# if the Pokemon name is not correct finds the closest name in the
# dataset and uses that instead
correct = ''
similarity = 0
if arr[0] not in Names_list:
    print(arr[0], " has not been found")
    for i in Names_list:
        if similarity < similar(arr[0], i):
            correct = i
            similarity = similar(arr[0], i)
    arr[0] = correct
    print("Closest Name: ", arr[0])

# reverses the array to make it easier to deal with the moves which
# are stored at the end of the array
arr.reverse()

# Checks if all the moves exist if not find and use the move with
# the most similar name
for i in range(4):
    correct = ''
    similarity = 0
    for j in Moves_list:
        if similarity < similar(arr[i], j):
            correct = j
            similarity = similar(arr[i], j)
    arr[i] = correct

# Puts the array back into the original order
arr.reverse()
ability_item = arr[2].split(" / ")
stats = arr[3].split(" / ")
newArr = arr[:2] + ability_item + stats + arr[4:]

# final check for any empty strings
```

```
    while("" in newArr):
        newArr.remove("")

    # returns the noiseless array
    return(newArr)

# Opens the needed data sets and converts to lists for easier use
Moves_df = pd.read_csv('All_Moves.csv', encoding = 'latin1')
Moves_list = Moves_df['Name'].tolist()
Pokemon_df = pd.read_csv('Pokemon.csv', encoding = 'latin1')
Names_list = Pokemon_df['Name'].tolist()

# Includes the tesseract executable in the path
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

# Stores the screen locations of the four moves which are always
# in the same place
movesPosition = [[259,622],[489,622],[259,691],[489,691]]

# Lists of exceptions that need to considered
SpaceNames = ['Mr. Mime', 'Type: Null', 'Tapu Bulu', 'Tapu Koko', 'Tapu Lele', 'Tapu Fini', 'Mr. Rime']
EntryHazardMoves = [['Spikes',3],['Stealth Rock',1],['G-Max Stonesurge',1],['Toxic Spikes',2],['Sticky
Web',1],['G-Max Steelsurge',1]]
HealMoves = ['Heal Order','Milk Drink','Moonlight','Morning
Sun','Purify','Recover','Rest','Roost','Shore Up','Slack Off','Soft-Boiled','Strength Sap','Synthesis']

# Instantiates the array that keeps track of which Pokemon are still useable
Fainted = [False]*6

# The main gameplay loop
run = True
while run:
    # Empty lists used to store the position of various UI elements that
    # are calculated at the start of the battle
    mousePosition = []
    nameScreenshotPosition = []
    movesScreenshotPosition = []
    HealthScreenshotPosition = []
    StatsScreenshotPosition = []
    defaultStatsList = []

    # Binds the AI to a key so that it only starts when the user wants
    # it to, allowing any necessary preparation
    if keyboard.read_key() == "#":
        # Sets initial variables
        Turn = 1
        StageHazards = []

        # Opens the Switch Menu to allow the AI to collect the
        # information it needs
        pyautogui.moveTo(877, 728)
```

```
    pyautogui.click()

    # Locates the timer button on screen as the where the mouse needs
    # to be to find the information on team changes depending on where
    # this button is
    Pos = pyautogui.locateCenterOnScreen("TeamPokemonPoint.png", confidence = 0.9)
    FirstSlot = [Pos[0]-450,Pos[1]+40]
    # Calculates and appends all the needed positions to gather
    # information to various arrays
    for i in range(3):
        mousePosition.append([FirstSlot[0]+(i*150),FirstSlot[1]])
        nameScreenshotPosition.append([FirstSlot[0]-85+(i*160),FirstSlot[1]-269])
        movesScreenshotPosition.append([FirstSlot[0]-68+(i*160),FirstSlot[1]-128])
        HealthScreenshotPosition.append([FirstSlot[0]-85+(i*160),FirstSlot[1]-215])
        StatsScreenshotPosition.append([FirstSlot[0]-85+(i*160),FirstSlot[1]-158])
    for i in range(3):
        mousePosition.append([FirstSlot[0]+(i*150),FirstSlot[1]+50])
        nameScreenshotPosition.append([FirstSlot[0]-85+(i*160),FirstSlot[1]-269])
        movesScreenshotPosition.append([FirstSlot[0]-68+(i*160),FirstSlot[1]-128])
        HealthScreenshotPosition.append([FirstSlot[0]-85+(i*160),FirstSlot[1]-215])
        StatsScreenshotPosition.append([FirstSlot[0]-85+(i*160),FirstSlot[1]-158])

    # Collects Team Data
    TeamList = []
    for i in range(6):
        # Finds and takes a screenshot of the Pokemon's name
        pyautogui.moveTo(mousePosition[i][0], mousePosition[i][1])
        image = pyautogui.screenshot(region =
(nameScreenshotPosition[i][0],nameScreenshotPosition[i][1],300,23))

        # Makes image easier to convert
        image = reduceImage(image, 300)
        # Converts to screenshot to text
        arr = imgConvert(image)

        # Finds the moves each Pokemon has
        image = pyautogui.screenshot(region =
(movesScreenshotPosition[i][0],movesScreenshotPosition[i][1],200,90))
        moveArr = imgConvert(image)
        # if all moves weren't found try again with less background noise
        if len(moveArr) < 4:
            image = reduceImage(image, 300)
            moveArr = imgConvert(image)

        arr += moveArr
        pokemonArr = []
        pokemonArr = arrClean(arr)
        exportList = []
        slotList = []
        sortList = []
```

```
        # Creates two lists, one for the AI to use during decision
        # making, and one to export to the Move Probabilty data set
        slotList = pokemonArr[1:]
        for j in slotList:
            sortList.append(j)
        sortList.sort()
        slotList.insert(0, pokemonArr[0])
        sortList.insert(0, pokemonArr[0])

        exportList.append(sortList)
        TeamList.append(slotList)

        print(exportList)
        # Collects the Pokemon's statistics in case the AI fails to
        # gather them properly at some point and needs something to
        # fall back on
        image = pyautogui.screenshot(region =
(StatsScreenshotPosition[i][0],StatsScreenshotPosition[i][1],440,23))
        stats = imgConvert(image)
        stats = stats[0]
        if "//" in stats:
            stats = stats.replace("//", "/")
        stats = stats.replace(" ", "")
        stats = stats.split("/")
        stat = []
        for i in stats:
            temp = []
            temp.append(i[:3])
            temp.append(i[3:])
            if temp[1][0] == " ":
                temp[1] = temp[1][1:]
            stat.append(temp)
        defaultStatsList.append(stat)

        # exports the sorted list to the data set
        export = pd.DataFrame (exportList, columns = ['Name', 'Move 1', 'Move 2', 'Move 3' ,'Move
4'])
        export.to_csv('Move Probability.csv', index = False, header=None, mode='a')

    # Cleans up the Move Probability data set to make it easy to use
    MProb_df = pd.read_csv('Move Probability.csv')
    Prob_list = []
    for index, rows in MProb_df.iterrows():
        temp_list =[rows.Name, rows.move_1, rows.move_2, rows.move_3, rows.move_4]
        Prob_list.append(temp_list)

    # Removes repeat values and instead updates the times they've appeared
    uniqueList = []
    for i in Prob_list:
        append = True
        for j in uniqueList:
```

```
            if i == j:
                append = False
            if append:
                uniqueList.append(i)

        for i in uniqueList:
            i.append(Prob_list.count(i))

        export = pd.DataFrame (uniqueList, columns =
['Name','move_1','move_2','move_3','move_4','amount'])
        export.to_csv('Move Probability.csv', index = False, header=True)

        # Prints out the team for testing and display purposes
        print("Current Team: \n")
        counter = 0
        for lst in TeamList:
            print(lst[0])
            print(f"Moves: {lst[1]}, {lst[2]}, {lst[3]}, {lst[4]}")
            print(f"Stats: {defaultStatsList[counter][0][0]}: {defaultStatsList[counter][0][1]}", end = ", ")
            print(f"{defaultStatsList[counter][1][0]}: {defaultStatsList[counter][1][1]}", end = ", ")
            print(f"{defaultStatsList[counter][2][0]}: {defaultStatsList[counter][2][1]}", end = ", ")
            print(f"{defaultStatsList[counter][3][0]}: {defaultStatsList[counter][3][1]}", end = ", ")
            print(f"{defaultStatsList[counter][4][0]}: {defaultStatsList[counter][4][1]}\n")
            counter += 1

        onField = 1
        # Battle Loop starts here
        while True:
            print(f'Turn {Turn}\n')

            # Opens the Pokemon Selection menu
            pyautogui.moveTo(877, 728)
            pyautogui.click()

            # Collects the current stats of all pokemon to take into
            # account any stat changes
            statsList = []
            for i in range(6):
                pyautogui.moveTo(mousePosition[i][0], mousePosition[i][1])

                image = pyautogui.screenshot(region =
(HealthScreenshotPosition[i][0],HealthScreenshotPosition[i][1],300,23))
                health = imgConvert(image)

                try:
                    health = health[0]
                except:
                    health = ""

                if 'fainted' in health:
                    Fainted[i] = True
```

```
            image = pyautogui.screenshot(region =
(StatsScreenshotPosition[i][0],StatsScreenshotPosition[i][1],440,23))
            stats = imgConvert(image)
            try:
                stats = stats[0]
                if "//" in stats:
                    stats = stats.replace("//", "/")

                stats = stats.split(" / ")
                stat = []
                for i in stats:
                    temp = []
                    temp.append(i[:3])
                    temp.append(i[4:])
                    stat.append(temp)
            except:
                stat = defaultStatsList[i]

            statsList.append(stat)

        # Collects Opponent Name Data
        pyautogui.moveTo(643, 452)

        # Searches for the Opponents information
        width, height = pyautogui.size()
        imagePos = (width,0)
        for i in listdir("Type Finder"):
            try:
                newPos = pyautogui.locateCenterOnScreen("Type Finder/" + i, confidence = 0.9)
                if newPos[0] < imagePos[0]:
                    imagePos = newPos
            except:
                pass

        image = pyautogui.screenshot(region = (imagePos[0]-25,imagePos[1]-32,300,23))

        Opp = imgConvert(image)

        # Checks for transformed Pokemon
        if "Type changed" in Opp[0]:
            Opp[0] = TeamList[0][0]

        if "Wishi" in Opp[0]:
            Opp[0] = "Wishiwashi"
        # Checks for exceptions to conventional naming
        if ' '.join(Opp[0].split(" ", 2)[:2]) in SpaceNames:
            Opp[0] = ' '.join(Opp[0].split(" ", 2)[:2])
        elif 'Alola' in Opp[0]:
            Opp[0] = Opp[0].split(')', 1)[0] + ")"
        else:
```

```
    Opp[0] = Opp[0].split(' ', 1)[0]

# If the Opponent isn't found the first time try again
# with background noise removed
if Opp[0] not in Names_list:
    print(Opp[0], " has not been found")
    image = reduceImage(image, 200)
    Opp = imgConvert(image)

    if ' '.join(Opp[0].split(" ", 2)[:2]) in SpaceNames:
        Opp[0] = ' '.join(Opp[0].split(" ", 2)[:2])
    elif 'Alola' in Opp[0]:
        Opp[0] = Opp[0].split(')', 1)[0] + ")"
    else:
        Opp[0] = Opp[0].split(' ', 1)[0]

correct = ''
similarity = 0
if Opp[0] not in Names_list:
    print(arr[0], " has still not been found")
    for i in Names_list:
        if similarity < similar(arr[0], i):
            correct = i
            similarity = similar(arr[0], i)
    Opp[0] = correct
    print("Closest Name: ", arr[0])

Opp = Opp[0]

print("Opponent: ", Opp)

# Calculates the probabilty of move types
Opp_moves_df = pd.read_csv('Move Probability.csv')
Opp_moves_df = Opp_moves_df[Opp_moves_df['Name'] == Opp]
Opp_moves = Opp_moves_df.values.tolist()

#Calculates Viability
eff_df = pd.read_csv('Pokemon.csv')
rslt_df = eff_df[eff_df['Name'] == Opp]
pointList = []
counter = 0
for lst in TeamList:
    # Calculates the weoght of each of the opponents moves
    TotalMoves = []
    UniqueMoves = []
    MoveAmount = []
    weight = 0
    weakness = eff_df[eff_df['Name'] == lst[0]]
    for i in Opp_moves:
        for j in range(1,5):
            MoveInfo_df = Moves_df[Moves_df['Name'] == i[j]]
```

```
                MoveInfo = MoveInfo_df.values.tolist()
                MoveInfo = MoveInfo[0]
                if MoveInfo[3] == '-':
                    MoveInfo[3] = 1
                if MoveInfo[4] == '-':
                    MoveInfo[4] = 1
                MoveDamage =
float(MoveInfo[3])*float(MoveInfo[4])*(weakness[MoveInfo[1]].values.tolist())[0]
                TotalMoves.append([MoveInfo[0],MoveDamage])

            for i in TotalMoves:
                if i not in UniqueMoves:
                    UniqueMoves.append(i)
            for i in UniqueMoves:
                total = (TotalMoves.count(i))/len(TotalMoves)
                MoveAmount.append(total)

            for i in range(0,len(UniqueMoves)):
                weight += UniqueMoves[i][1]*round(MoveAmount[i], 2)

            # Calculates the value of each move to decide what to do
            points = []
            PokeStats = statsList[counter]
            currentPokemon = pd.read_csv('pokemon.csv')
            currentPokemon = currentPokemon[currentPokemon['Name'] == lst[0]]
            currentPokemon = currentPokemon.values.tolist()
            # Checks if it has the right stats
            if len(PokeStats) != 5:
                PokeStats = defaultStatsList[counter]
            for i in range (1,5):
                STAB = 1
                Mv_df = Moves_df[Moves_df['Name'] == lst[i]]
                move = Mv_df.values.tolist()
                moveType = move[0][1]
                moveStat = move[0][2]
                moveDamage = move[0][3]
                moveAccuracy = move[0][4]

                if moveDamage == '-':
                    moveDamage = 1

                # Considers using stat moves based on how long that
                # Pokemon has been on the field for
                if moveStat == 'Status' and move[0][0] not in HealMoves:
                    if counter == 0:
                        moveDamage = 100/onField
                    else:
                        moveDamage = 40

                # Considers using Entry Hazards if the max amount of
                # that hazard isnt on stage yet
```

```
            for i in EntryHazardMoves:
                if i[0] == move[0][0] and i[1] != 0:
                    moveDamage = 120
                    i[1] -= 1

            # Checks if the current pokemon needs healing
            if pyautogui.pixel(212,453) != (0, 187, 81) and move[0][0] in HealMoves:
                    moveDamage = 100

            # Considers the stat that governs the strength of the
            # move
            atkStat = 0
            if moveStat == 'Physical':
                atkStat = PokeStats[0][1]
            if moveStat == 'Special':
                atkStat = PokeStats[2][1]
            if moveStat == 'Status':
                atkStat = PokeStats[4][1]

            if moveAccuracy == '-':
                moveAccuracy = 1

            if moveType == currentPokemon[0][2] or moveType == currentPokemon[0][3]:
                STAB = 1.5

            try:
                Speed = float(PokeStats[4][1])
            except:
                Speed = float(defaultStatsList[counter][4][1])

            # Calculates total value of each move

            totalpoints = float(((rslt_df[moveType]*float(moveDamage)*STAB))-
weight)*float(atkStat)*float(moveAccuracy)*Speed

            # if that pokemon is not already on the field halve
            # the value to simulate a turn being lost switching
            # that pokemon in
            if counter != 0:
                totalpoints = totalpoints/2

            # Sets negative values to 1 to make even a extremely poor
            # choice better than trying to use a Pokemon that is no
            # longer useable
            if totalpoints<=0:
                totalpoints = 1

            # If a Pokemon is no longer useable set all it's value to
            # zero so the AI doesn't try to use them
            if Fainted[counter] == True:
                totalpoints = totalpoints*0
```

```
            points.append(totalpoints)
        pointList.append(points)

        counter += 1

    # Prints the probability of an Opponent having a move
    print("Moves: ", end="")
    for i in range(0,len(UniqueMoves)):
        print(f"{UniqueMoves[i][0]} - {round(MoveAmount[i]*100, 2)}%", end = "   ")
    print("")

    # Prints the value of each decision
    for i in range (0,6):
        print(TeamList[i][0])
        print(TeamList[i][1] + ": " + str(pointList[i][0]))
        print(TeamList[i][2] + ": " + str(pointList[i][1]))
        print(TeamList[i][3] + ": " + str(pointList[i][2]))
        print(TeamList[i][4] + ": " + str(pointList[i][3]))
        print("Best: " + str(max(pointList[i])) + '\n')

    # Finds the order of best choices in case something happens
    # that makes the first choice no longer useable
    maxList = []
    for i in pointList:
        maxList.append(max(i))
    swap = maxList.index(max(maxList))
    faintSwap = maxList.index(sorted(maxList, reverse=True)[1])

    # if the best choice doesnt require swapping in
    if swap == 0:
        # increases how long the Pokemon has been on the field for
        onField += 1
        counter = 1
        # Opens the Attack menu
        pyautogui.moveTo(86, 726)
        pyautogui.click()
        # Picks the best option
        index = pointList[swap].index(max(pointList[swap]))
        pyautogui.moveTo(movesPosition[index][0], movesPosition[index][1])
        pyautogui.click()
        time.sleep(2)
        # if that option is not available pick the next best option
        # until an option is available
        attacking = (0,0)
        while attacking != None:
            attacking = pyautogui.locateCenterOnScreen("CheckAttack.png")
            if attacking != None:
                index = maxList.index(sorted(maxList, reverse=True)[counter])
                pyautogui.moveTo(movesPosition[index][0], movesPosition[index][1])
                pyautogui.click()
```

```
            pyautogui.moveTo(attacking[0]+50,attacking[1])
        counter += 1

# if the best move is to swap out
else:
    onField = 1
    counter = 1
    attemptSwap = True
    while attemptSwap:
        pyautogui.moveTo(mousePosition[swap][0], mousePosition[swap][1])
        pyautogui.click()

        # if the pokemon can not be swapped out take the best
        # option with the pokemon already on field
        trapped = pyautogui.locateCenterOnScreen("Trapped.png", confidence = 0.9)
        if trapped != None:
            pyautogui.moveTo(86, 726)
            pyautogui.click()
            pyautogui.moveTo(movesPosition[index][0], movesPosition[index][1])
            pyautogui.click()

            attacking = (0,0)
            while attacking != None:
                attacking = pyautogui.locateCenterOnScreen("CheckAttack.png")
                if attacking != None:
                    index = maxList.index(sorted(maxList, reverse=True)[counter])
                    pyautogui.moveTo(movesPosition[index][0], movesPosition[index][1])
                    pyautogui.click()
                    pyautogui.moveTo(attacking[0]+50,attacking[1])
                counter += 1
        # check if the Pokemon has fainted and the AI is unaware
        fainted = pyautogui.locateCenterOnScreen("FaintCheck.png")
        if fainted == None:
            attemptSwap = False
        else:
            Fainted[swap] = True
            swap = maxList.index(sorted(maxList, reverse=True)[counter])
        counter += 1

    TeamList[0], TeamList[swap] = TeamList[swap], TeamList[0]
    Fainted[0], Fainted[swap] = Fainted[swap], Fainted[0]

# Checks if the next turn has started
counter = 2
attacking = pyautogui.locateCenterOnScreen("CheckAttack.png", confidence = 0.9)
while attacking == None:
    # sends out the second best pokemon in case of faint
    pyautogui.moveTo(mousePosition[0][0], mousePosition[0][1])
    pyautogui.click()
    fainted = pyautogui.locateCenterOnScreen("FaintCheck.png", confidence = 0.9)
    print("fainted: ", fainted)
```

```
            if fainted != None:
                pyautogui.click()
                pyautogui.moveTo(mousePosition[faintSwap][0], mousePosition[faintSwap][1])
                pyautogui.click()
                Fainted[faintSwap] = True
                TeamList[0], TeamList[faintSwap] = TeamList[faintSwap], TeamList[0]
                faintSwap = maxList.index(sorted(maxList, reverse=True)[counter])
                counter += 1
            attacking = pyautogui.locateCenterOnScreen("CheckAttack.png", confidence = 0.75)
            print("attacking ", attacking)
            time.sleep(3)

        Turn += 1
```

## Appendix G: Data sets

Provided in the extra submissions area of Moodle

## Appendix H: Executables

Provided in the extra submissions area of Moodle

## Appendix I: Websites

https://play.pokemonshowdown.com/