Controlling Playable Characters with AI

Deliverable 1 Document

Name : Saarah Huda Wasif Naheel

ID : H00232155

Supervisor : Dr. Hani Ragab

Program : BSc. Computer Science (Hons) - 4th Year

Date : November 24, 2018

Declaration

I, Saarah Huda Wasif Naheel confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: *Saarah Huda Wasif Naheel*

Date: November 24, 2018

Abstract

In the field of Artificial Intelligence, research has been conducted on a wide range of applications in various fields. One common implementation of Artificial Intelligence is in the world of gaming. Artificial Intelligence is known to bring commercial value to games, and boost user experience. The project aims to develop an application which executes machine learning algorithms for games run within emulators. Optionally, the project explores the possibility of enhancing current machine learning algorithms applied to display artificially intelligent characteristics in games.

Key Words: Artificial Intelligence, Games, Emulators, Machine Learning Models

Contents

Chapter 1

Introduction

Research on artificial intelligence with games has been conducted for a long period of time. AI researchers aim to efficiently control playable and non-playable characters to perform in an intelligent manner to solve a variety of tasks. In the event that the researchers are able to achieve this goal, it can be witnessed how artificial intelligence is closer to achieving true artificial intelligence. True artificial intelligence is when an AI agent showcases human-like intelligence and characteristics. Games provide a platform for AI researchers to execute algorithms due to the player interaction with complex environments which are similar to human interactions with its surroundings.

1.1 Aims

The aim of the project is to develop and design an application to implement machine learning models on emulated games and analyse the results. At least three machine learning algorithms will be applied to two emulated games. Optionally, we will examine the possibility of proposing improvements on existing machine learning models. We will also optionally investigate the possibility to generate machine learning models.

1.2 Objectives

- To develop an application which executes a file containing the machine learning model script for the emulated game.
- To investigate and compare the performance of different machine learning models on games.
- Optionally, to examine the likely enhancements on the current existing machine learning algorithms applied to games.
- Optionally, to explore the possibility of creating a machine learning model script by taking the emulated game information and the specification of machine learning model as inputs.

1.3 Manuscript Overview

The manuscript consists of 6 chapters. The first chapter provides a brief introduction to the aims and objectives of the project. The second chapter outlines and critically compares the related works carried out in the gaming industry with AI. The third chapter identifies the requirements that need to be fulfilled for the application to run in the expected manner. The fourth chapter provides an overview of the system architecture and the initial graphical user interface design of the application. The fifth chapter describes an evaluation strategy to evaluate all the requirements mentioned in the third chapter. The final chapter comprises of the methodology used to successfully develop the project with an efficient project plan. All subsequent risks are included which may occur during the development of the application. Professional, ethical, social and legal issues are also mentioned briefly.

Chapter 2

State of the Art

Artificial Intelligence (AI) and gaming share a deep past. It started out with creating AI agents to play games and it did not matter if they had the ability to learn or not. Tic Tac Toe was the first game mastered by a program. Within a few years, more research was carried out in regard to AI learning and traditional board games and this went on for decades. Finally, the Chinook program and IBM's Deep Blue algorithm solved the roadblock of AI on checkers and chess respectively. The next AI benchmark was to learn the board game called Go, because of its greater complexity than Chess or Checkers. The Google Deepmind's AlphaGo recently became successful in learning to play Go and therefore overcame this above-mentioned benchmark. AlphaGo is based on a deep reinforcement learning approach. Atari 2600 video games were also learnt using Google Deepmind with raw pixel screenshots as inputs. Most of the AI research on this field concentrates on making the AI play like a human expert or at the very least, learn to play more efficiently. AI can also be used for content or level generation and to model players in games. Most of the recent research on AI and games is based on believable agents. Believable agents are those agents that pass the Turing test for games. [1, pp. 8–10]

To achieve true artificial intelligence, the program or software should be able to think and make decisions on its own given a diverse range of scenarios. Machine learning algorithms perform extremely well on tasks related to finding patterns from data. These algorithms learn through patterns on large amounts of data, and outputs conclusions and observations learnt from the data. Human senses work in a similar manner, but this is not intelligence. Humans do not watch things all day. Humans move around, interact with the environment, make decisions and learn to adapt to varying non-predictable environment [2]. Character interacting with objects in the game is the exactly the same as humans interacting with the environment. The game character learns from the environment and carries out a sequence of decisions to maximize rewards and learn from its failures similar to the way humans do.

AI is also beneficial to the game market. Including some component of AI in the game increases its value and therefore boosts user experience. AI component is included in the game, based on a simple algorithm to find the

next move or a complex algorithm to learn character behavior unless it achieves the above-mentioned purpose. AI can play the game with two different motives. The first motive is that a player plays to learn and optimize its performance. Game testing automation is an example of a significant field where such an AI is beneficial. The second motive is that a non-player that learns to play human like and believable leading to enhancement of player experience and increase in difficulty level of the game. [1, Ch. 1, pp. 23–24]

2.1 Background

This section covers the most common games used for artificial intelligence testing, and their corresponding emulators. Machine learning models which are most famous in controlling player characters to show intelligent behavior are also mentioned.

2.1.1 Games

The Atari 2600 games are most commonly used in AI research because of their balance between classic board games and current complex graphical interface games. AI in Atari games use planning strategies and interpret the game dynamics, which is identical to strategies used for board games. Atari games provide a platform for AI to learn from the actions occurring on the screen relative to recent video games. Atari 2600 encapsulates games of different genres and contains a standard interface design and controls [3].

Nintendo Entertainment System (NES) is one of the best rated gaming consoles and contains the classic popular games like Super Mario Bros and its variations, Bionic Commando, Bubble Bobble among many other games. Games belonging to this console are programmed to use memory resourcefully and primarily use 2048 bytes of RAM. Most of the game information such as the player's health, score and dimensions of the screen are usually present in static locations in the memory [4]. Manipulating these memory locations provide us with the necessary positions of the character, features (health, score, etc.) and dimensions of the game to run different AI algorithms effortlessly.

2.1.2 Emulator

Arcade Learning Environment (ALE) is a well-recognized and high-quality emulator to run AI algorithms on Atari 2600 games [3]. ALE is an open-source platform which focusses generally on planning and reinforcement learning. A variety of interfaces exists to interact with ALE on a wide range of programming languages. ALE provides extra functions like restoring the state and saving the contents stored in the registers, address counters, memory content of the game and so on [5].

NES games are supported by several emulators, but the most renowned emulators are FCEUX, BizHawk etc. Most of the NES emulators provide support to multiple platforms and languages. These emulators provide the same mechanism of saving the state and restoring it as ALE [4].

2.1.3 Machine Learning Algorithms

Basic common algorithms applied in the game domain are supervised learning, unsupervised learning, reinforcement learning and hybrid algorithms.

Supervised learning involves recognition of patterns and relationships from data which are labelled. Patterns and relationships are identified in order to predict the label for a data instance, which needs to be labelled. Artificial neural network (ANN) is a form of supervised learning algorithm which is used in AI game playing [1, Ch. 2, pp. 57–59].

In contrast to supervised learning, unsupervised learning techniques encompass the recognition of patterns and relationships from data which are unlabeled. Clustering and frequent pattern mining are two types of unsupervised learning techniques. They are used to learn the behavior of a character or build upon existing game levels through the use of player data in order to balance out the game [1, Ch. 2, pp. 77–78].
The above-mentioned techniques are pattern recognizing algorithms and are not learning techniques, and thus do not show human like intelligence. They require huge amounts of data for supervised and unsupervised learning. In tree search algorithms, the AI does not learn from its actions, it only tries to follow an optimal sequence of actions.

Reinforcement Learning is another form of a machine learning technique which learns through continuous interactions between the agent and the environment. The agent at time t in a state s takes a possible action a and gets a reward r for carrying out the action. The agent needs to figure out best actions to take place which will maximize the sum of the rewards [1, Ch. 2, pp. 70–71] [6].
Deep reinforcement learning has been successful in the field of gaming. DRL is a combination of deep learning and reinforcement learning [7]. ***Mnih* et al.** [8] mention three notable challenges faced in the context of deep learning by reinforcement learning, which are stated as follows:

- Deep learning algorithms require huge amount of data to be labeled for training, but Reinforcement Learning algorithms learn through reward signals.
- Instances of data are considered to be independent in deep learning technique, in reinforcement learning we discover data which is likely to be related to each other.
- Deep learning requires the data distribution to be fixed and should not be altered when learning new behaviors, but for reinforcement learning data transforms as it learns.

The challenges above are attempted to be achieved through the use of a single recurrent neural network with raw pixel data, and controls as input, to discover control policies which should perform well in learning to play several games. The network is built with a modified version of a Q-learning algorithm, in which the weights are updated using stochastic gradient descent. To overcome data correlation and transformation that happens in Reinforcement Learning, we use a replay method to level the training data distribution by using random samples from previous conversions.

The parameters which approximate the value function are updated based on the sample of experience. The network learns from interacting with the environment. The experience sample for each time stamp is stored in a dataset. In the experience replay step, an experience sample is randomly chosen and is used to update the parameters above. The many advantages of applying this variation to Deep Q learning are:

- Weights are updated by experience which provide better data efficiency.
- Learning is done based on selecting random experience sample. This breaks the correlation between the data sample.
- This also smoothens the learning as its behavior is the mean behavior of the previous states.

Therefore, solves the three challenges faced in the deep Q learning approach to reinforcement learning [8].

Neuroevolution (NE) algorithms are hybrid techniques. One such popular Neuroevolution algorithm is called Neuroevolution of Augmenting Topologies (NEAT). NEAT evolves network structure and weights of the neural network through the use of genetic algorithm. The network information, which comprises of the node and connection genes, is contained inside a genome. Node genes represents the nodes in the input, hidden and output layer. The connection gene represents the linking of two node genes (output and input node), and it consists of a weight, flag value (dependent on whether its enabled or disabled) and an innovation number. In order to track the gene evolution history an innovation number is given to the gene. Each genome contains a random mutation rate for each type of mutation technique.

There are mainly four techniques of mutation in NEAT. A node can be mutated through randomly updating its weights. Node and connection genes can be mutated through changing the state, from disabled to enabled, or vice versa. The add node mutation inactivates a connection and creates a new node with one connection of weight which has the value of one. The other connection contains the weight of the previously disabled connection. Add link mutation is a technique which randomly adds a new connection between two nodes with weights between the range of negative two and two.

In NEAT, genome crossover is done in a meaningful way in order to protect genomes which are weak but contain useful topological information. These genomes are eliminated before their weights are optimized. In order to avoid the above-mentioned issues for genes, new node genes and connection genes are created during add node and add link mutation, followed by the incrementation of the global innovation number. The new innovation number

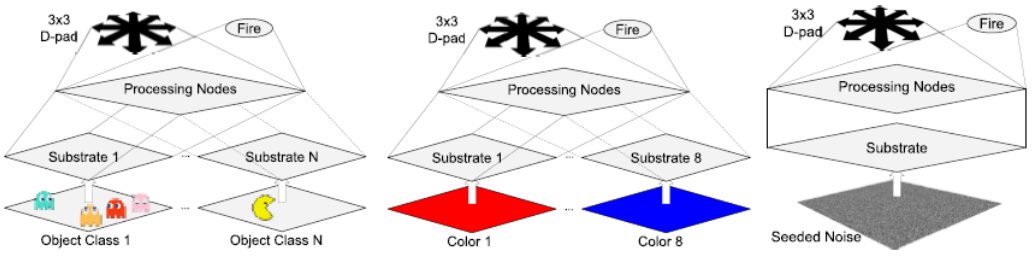
is assigned to the new gene created and thereby helps in the tracking of gene evolution. Two genes with same innovation number must represent same topology.

Furthermore, the genomes which are similar to each other are grouped together into species. They are split into species by calculating the sum of disjoint or excess genes in each genome and the difference in weights between the identical genes. If the sum is below a given threshold then it is placed into that species. We split genomes into species so that they can fight amongst the genomes present in the species and eliminate the weak ones instead of the entire genome pool. This helps to retain new topological innovation in genomes which do not have a high fitness yet, due to insufficient evolution of weights [9].

The crossover happens between two genomes to create a new genome which contains the best genes from the parent genomes. The genes from the parent genomes are ordered based on their innovation number. For each innovation number, the gene from the most fit parent is chosen and if both the genomes have equal fitness, then the genes are randomly picked from either of the parents, which are then placed into a child. If a particular innovation number of genes is present only in one parent, then this gene is considered as a disjoint or an excess gene and are also placed in the new genome.

2.2 Artificial Intelligence and Games

***Hausknecht* et al.** [3] assess neuroevolution algorithms on the basis of learning diverse Atari video games with minimal game domain knowledge. Four unique neuroevolution techniques were used for this purpose. Conventional Neuro-evolution (CNE) algorithm evolves weights of a fixed network topology artificial neural network (ANN). The next algorithm uses the Covariance Matrix Adaptation Evaluation Strategy (CMA-ES) to evolve weights in a fixed network topology ANN. NEAT algorithm evolves both weights and topology of an ANN. Lastly HyperNEAT evolves the topology and weights of an indirect encoding known as Compositional Pattern Producing Network (CPPN) using NEAT. To estimate the weights for the nodes in the layers adjacent to each other in the ANN, we use CPPN. The algorithms were applied using three different state input representation of the game, which were object, raw-pixel and noise-screen. The noise screen representation produces random activations. This is done to analyse if the algorithms are memorizing by assessing if the algorithms can figure out the correct actions from the input and ignore the random actions happening in the game.



(a) Object Architecture (b) Raw-Pixel Architecture (c) Noise-Screen Architecture

Figure 2.1: Architectures

Figure 2.1 presents a graphical representation of the system architecture for different input representation. A total of 55 Atari games with different genres were chosen. A policy for each game was generated separately. This policy consisted of 150 generations and 100 individuals per generation. 15k episodes play per each game and an episode ran for a game interval of 13 mins. The game score was the fitness score of the individual at the end of each episode. The fittest individual at the end of 150 generations is considered as the score for the neuro evolution agent. Their technique is tested by using z-score for score normalization. This is done since each game has its own different game scores. Using this normalization, we know how many standard deviations a score is from the mean score.

The performance of the algorithms was analyzed based on Analysis of Variance (ANOVA) statistical test with a Turkey's HSD significance test. From conducting the above experiment, they found out that for object representation, NEAT and HyperNEAT outperformed the fixed topology networks. For noise representation, NEAT performed better than the rest of the algorithms. For raw pixel input representation, only HyperNEAT was able to learn due to the indirect encoding in HyerNEAT. Even though HyperNEAT algorithm is more complex and contains indirect encoding, it did not seem to perform better than NEAT as long as the inputs were object or noise. Learning algorithms (NEAT and HyperNEAT) were compared with planning algorithm, random play and human high scores by experts. The planning algorithm gives better performance than the learning but both of them perform better than random play. Neuro evolution methods gave a better score than the high scores for three of the games but on average the score of the algorithms were beaten by the human experts.

The performance of the algorithm reduces depending on the generalization of the input representation. As Atari games are deterministic and do not require the history of the character actions to predict the next move and therefore all the algorithms implemented are feed forward networks. To apply the algorithm to non-deterministic games, ***Hausknecht* et al.** mentioned the possible implementation of recurrent networks. The objects were recognized manually for each Atari game, this is not very efficient for a large number of games or a game which contain many objects. The manual recognition raises the possibility of error in regards to identifying objects. Time consumption is another factor which could be taken into account for the manual recognition of objects for each game.

***Mnih* et al.** [8] presents an alternative to HyperNEAT for Atari games. The paper incorporates deep reinforcement learning with Atari games. This algorithm contains a Deep Q-Network with experience replay.
This system was evaluated by calculating the mean of the maximum predicted Q-values of a set of randomly selected states. The rewards are discounted when the agent follows the policy on the particular state.
This deep reinforcement learning algorithm was compared to other Reinforcement learning algorithms such as the Sarsa(λ) algorithm [3] and the contingency algorithm [4], both of which are considered as top performing models. Both of these algorithms had more prior information of the game by having each of the 128 colours as a separate channel and conducted background subtraction in order to understand the given visual problem in a game. The

deep reinforcement learning algorithm on the other hand learnt by RGB screenshots, which are converted into raw gray scale screenshots, and the detection of game objects is its own task to do. This algorithm performs better than the other reinforcement learning algorithms even with less knowledge about the game. The algorithm beat human players in almost half of the games. The ones it scored less than the human scores were primarily strategy games which require a lot of time to learn and play. The wide variety of possible moves in a full-fledged strategy game would cause the algorithm to take a significant amount of time to learn due to the complexity of each move, considering an evaluated decision of good and bad moves must be taken.

McPartland **et al.** [6] primarily aim to apply Reinforcement Learning in First-Person Shooter games. A complex AI bot is used which learns through navigation and gathering items in an user created three-dimensional environment similar to a maze. The AI bot also learns through combat scenarios. In order to learn the bot controllers, Reinforcement Learning uses Sarsa(λ) algorithm. The secondary aim brought forward by the author is to analyse a way to effectively create generalized autonomous bots through the use of combining Reinforcement Learning controllers.

The system was evaluated through three tasks; Navigation Task, FPS Combat Task and General Purpose Task. The navigation task was aimed towards testing how efficiently a bot could traverse a maze-type setting while gathering items. The aim of the FPS combat task was based on the ability of how well a bot could fight when trained against an opponent which was controlled by a state machine. This state machine bot was encoded to fire a shot whenever an enemy was within sight, while ensuring that the weapon was ready to fire. The state machine controlled bot keeps enemies within range until elimination. The final task, general purpose task, was setup with four types of bots. One was a Reinforcement Learning bot, another was a state machine bot, and the remaining were non-reacting bots. The state machine controlled bot was encoded with aggressive behavior, which explores the playing field, gathers items within range, and hunts down enemies when sighted. The non-reacting bots were placed to give the Reinforcement Learning bot targets to practice on and learn a generalized combat strategy. Three types of Reinforcement Learning models were used HeirarchicalRL, RuleBasedRL and Reinforcement Learning. All the bots were compared to provide average, lowest and highest results. The conclusions show that the hierarchical reinforcement learning and the rule based reinforcement learning models presented higher performance in combat and navigation is comparison to the flat reinforcement learning model.

The algorithms were run and tested within a user created game environment in which all bot spawn points, and item spawn points were the same throughout all the trials. The weapons have unlimited ammunition, with a 1 second delay rate between shots fired. We can say that the models were over-fitting since the game environment was static throughout all the trials, which decreases the chances of the AI actually learning. If any of these bots were placed into an actual game environment with dynamic respawn points and limited ammunition capacities with varying reload times from different weapons, the accuracy may possibly drop significantly.

***Lample* et al.** [10] present an AI-agent which was used for playing FPS games, in a game mode which involved achieving the maximum number of kills by the AI-agent. The agent used Deep Recurrent Q-Network (DRQN) with Long Short-Term Memory (LSTM), and was trained in two phases, one being navigation, and other being shooting. The training was done through reward shaping and additional supervision from ViZDoom platform. The VizDoom platform is used for the game Doom. The internal configuration and ground-truth knowledge of enemies provided by ViZDoom is used during training.

The agent was evaluated against game bots which are built into the game, with and without using navigation. The evaluation was run for 15 minutes on each map, while for full death matches the evaluation was run with an average of 10 training maps and 3 test maps.

The evaluations cover the navigation aspect of DRQN, comparing scores to human counterparts, and game features aspect of DRQN. With navigation enhancements, the agent showed significant improvements to the number of items picked up and the kill/death ratio in comparison to having no navigation enhancements. The agent also outperformed humans, whose scores were averaged over 20 individuals. Using game features for training improved performance significantly with notable differences in the kill/death ratio.

The modified DRQN model, and an alternative model presented by ***Dosovitskiy* et al.** [11] known as direct future prediction, took part in a competition called the Doom AI Competition. The alternate model only took part in the full Deathmatch track and beat the DRQN model by over 50%. Direct future prediction was a simple model and did not require the additional supervision which is presented within the modified DRQN model.

***Nielsen* et al.** [12] aim to create agents which can effectively play a range of general strategy games. Six games are played, which are two player turn-based games on a two dimensional grid where the victor is the one which removes the pieces put forth by the opponent. Strategy Game Description Language (SGDL) is a game engine which is used to assert a variety of strategy games. Six architectures are used for agents, along with their variations. All architectures based on learning algorithms were put through substantial training for each agent. All architectures were given equal time for training.

The agents went through two types of evaluation, namely competitive evaluation and user interaction evaluation. In competitive evaluation, each agent was pit against another agent on a wide range of models. This helped analyse the general performance of all the trained agents. In user evaluations, human participants were asked to play against a minimum of two agents which use different game models, and then provide answers in relations to their preference between the two agents. This helped to provide data on which agent proved to be the best against humans, and which agents had the most satisfying interactions with the participants.

Many agents would prove to fare poorly against strategy games which have high branching factors. Branching factors are the number of potential moves per turn. High branching factors can prove to be very challenging for AI systems to overcome. Certain games like Civilization could prove to be very computationally expensive due to its sheer complexity. The paper mentions that this is a topic of research.

***Kunanusont* et al.** [13] create a learning agent which has the ability to learn from video game screenshots through the use of Deep Q-Network within General Video Game AI framework (GVG-AI). The Deep Q-Network method was implemented with experience replay on six games grouped into two categories, namely shooting games, and exit-finding games. The screen block size, dimensions and level dimensions were used as inputs to the algorithm. There is no restriction on time, which means that the algorithm terminates once the game ends. The algorithm is compared in performance by evaluating it against Monte Carlo Tree Search planning algorithm.

The authors did not train many games from the known 140 GVG-AI games due to the long amount of time taken to train each game. The comparison is biased since the Monte Carlo Tree Search had a time limit, while the algorithm used by the author has no time constraints. The Convolution Neural Network (CNN) is pre-trained before applying it to a game. In the event that the game is too large, the CNN work better if it could be expanded. When using General Video Games AI, the learning agent should be able to use its previous experience from other games, into a new game. This would be more efficient since the AI does not have to learn from the very beginning for each game, which is similar to human behavior.

A similar approach is taken by ***Woof* et al.** [14] who describe an alternative state representation of General Video Game AI in which the input is sent as a game object to the Deep Q-Network. The authors compare state representation with two others, pixel representation and feature representation. Object state representation was shown to have a higher average score over the other two in games where the environmental ground-truth was available for extraction. The ground-truth in this context refers to the mechanics of the game and the relation between the objects and their domain.

Chapter 3

Requirements Analysis

This chapter outlines the system and user requirements necessary for the development of the system. These requirements describe the functional and non-functional features of the application.

3.1 System Requirements

The requirements which formulate the functionalities which the system has are as follows:

3.1.1 Functional Requirements

S-FR- 1 The system shall interact with the emulator

Priority: Must Have

The system will open the emulator that supports the selected game.

S-FR- 2 The system shall insert the ROM of a selected game on an emulator

Priority: Must Have

The system will open the ROM of a user selected game on an emulator based on the console that the game was originally developed for.

S-FR- 3 The system shall process the selected machine learning model on the game.

Priority: Must Have

The system will process the user selected machine learning model on the user selected game using the game supported emulator.

S-FR- 4 The system shall allow users to modify settings of the algorithm.

Priority: Can Have

The system will provide users options to alter the parameters of the algorithm.

S-FR- 5 The system shall generate a machine learning model.

Priority: Want to Have

The system will generate a machine learning model script file which takes the game information and the specification of the ML model as inputs.

3.1.2 Non-Functional Requirements

S-NFR-1 Functional Scalability

Priority: Can Have

The system shall have the ability to add new functionality while minimizing effort.

3.2 User Requirements

The requirements which formulate the functionalities which the user must have are as follows:

3.2.1 Functional Requirements

U-FR-1 The user shall be able to select a game on which he wants to run a machine learning algorithm.

Priority: Must Have

The user will be able to choose a game from a list of games on which he wants to execute a machine learning algorithm.

U-FR-2 The user shall be able to select machine learning algorithm to execute on the game.

Priority: Must Have

The user will be able to choose a machine learning algorithm from a list of ML models to execute on a selected game.

U-FR-3 The user shall be able to modify settings of the ML algorithm.

Priority: Can Have

The user will be able to change the parameters of the selected ML algorithm.

3.2.2 Non-Functional Requirements

U-NFR-1 Interactivity

Priority: Must Have

The user shall be able to interact with the system through a Graphical User Interface.

U-NFR-2 Usability

The user shall have a user friendly and easy to use user interface.

U-NFR-2.1 **Interface Design**

Priority: Must Have

The user shall have an intuitive and efficiently designed user interface.

U-NFR-2.2 **Help Section**

Priority: Want to Have

The user shall have access to a guide which shall help improve the learnability of the system.

U-NFR-3 Quality

Priority: Must Have

The user shall have a user interface as per the intended purpose, while satisfying user prospects.

Chapter 4

Design

This chapter outlines the user interface design and system architecture. The system architecture is split into two designs. The primary design outlines the primary functionalities of the system, while the optional design outlines the functionalities of a system given that a user is able to upload their own configurations.

4.1 Graphical User Interface Design

The user interface design is an essential part of user experience. A good design has the ability to attract people and keep them engaged in a simplistic manner. The graphical user interface for this project is aimed towards simplicity.

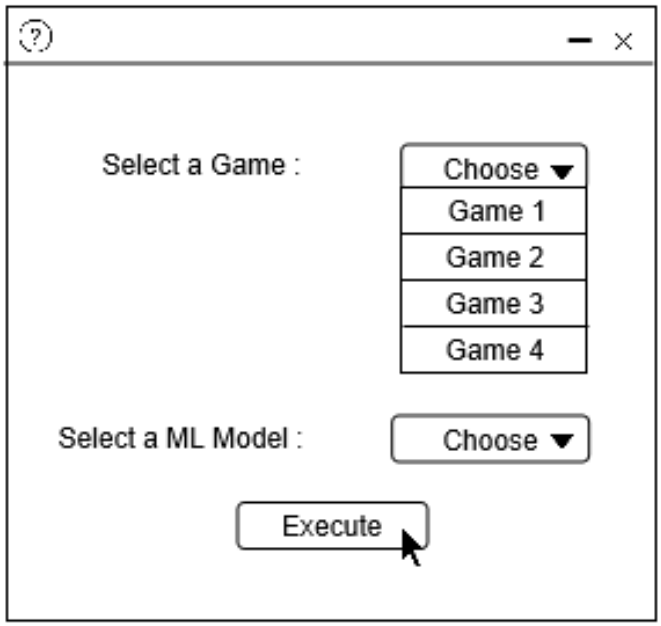


Figure 4.1: Graphical User Interface Design

The figure 4.1 describes a basic and simple straight forward GUI interface of the application. Users can select a game and a machine learning model from the drop-down menu containing the list of different games and ML models. Once the selection is made, the user clicks on the execute button. The rest of the process is carried out by the system. Therefore, the initial design of the application is assumed to be user friendly and easy to navigate and fulfil its purpose.

4.2 System Architecture

The system architecture outlines the structure of the system, and how each component interacts within the system. Designs are created for the primary functionality of the system, and the optional functionality of the system.

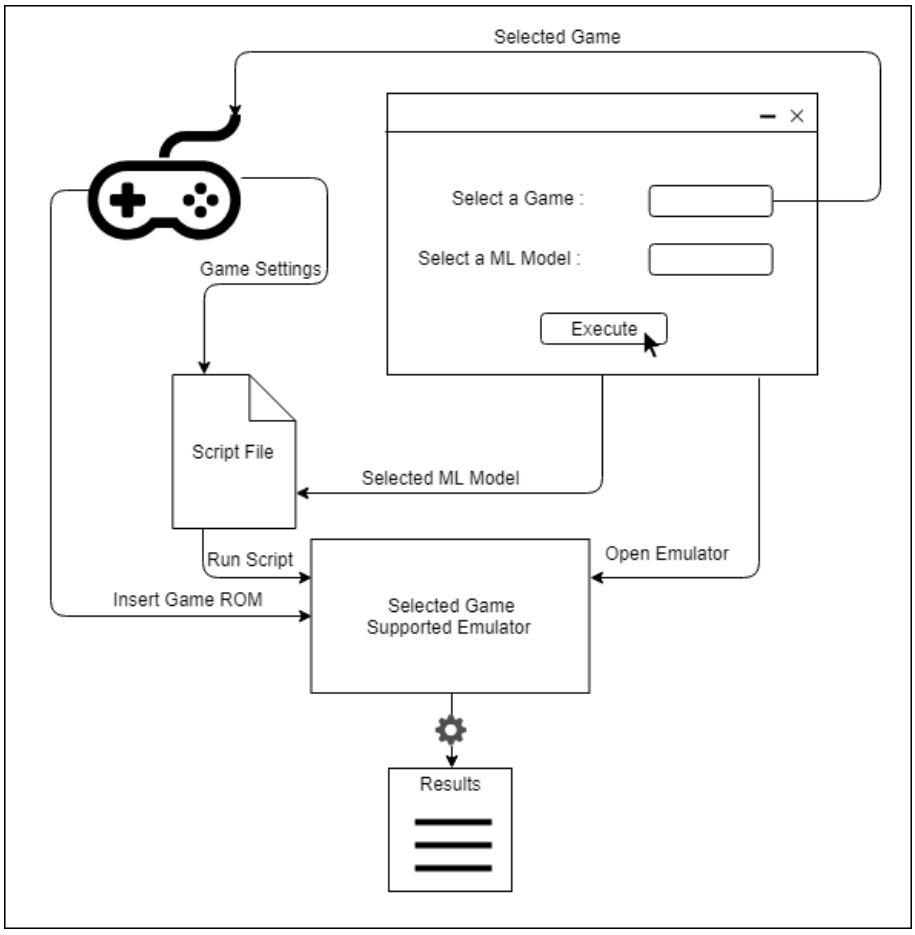


Figure 4.2: System Architecture for the Primary Functionality of the Application

The figure 4.2 represents the functionality of the project and how the different components interact with each other. When the user selects a game and machine learning model of his choice, he must click the execute button which will trigger the loading of a script. The script contains the game information and selected ML model, which is then executed on an emulator which supports the selected game. The emulator is opened, then the ROM of the selected game is opened and finally the script is run. After executing the model on the game, the system shall generate the result of the process.

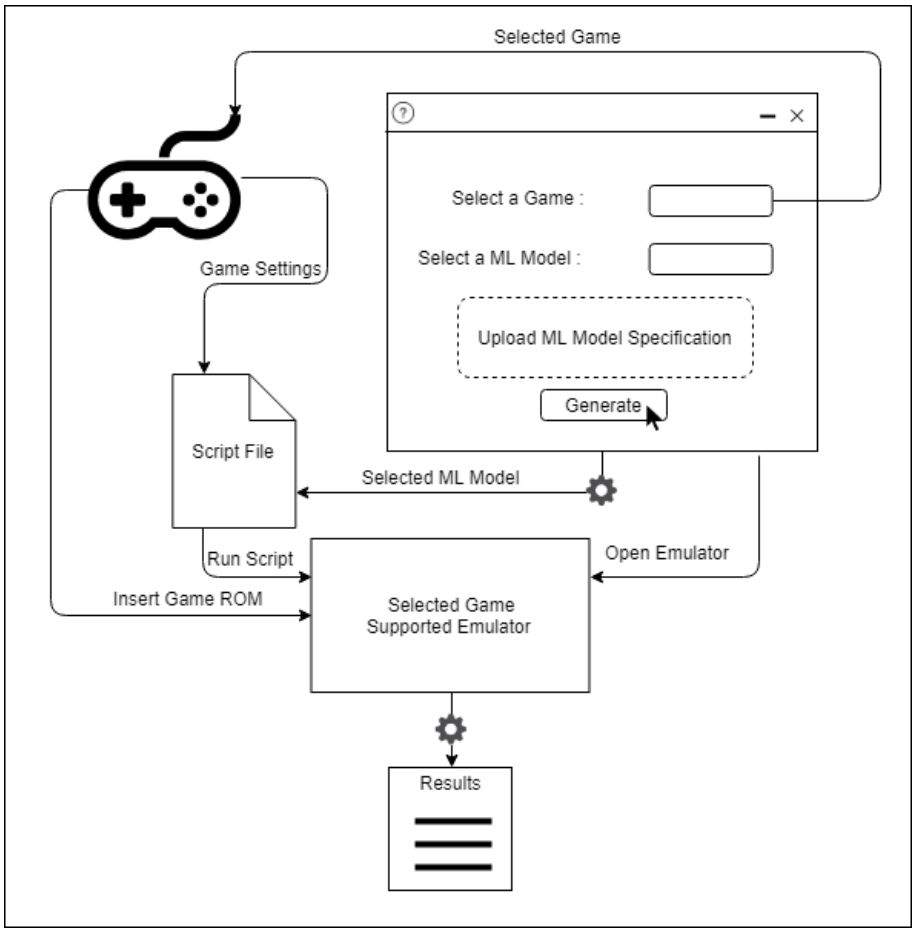


Figure 4.3: System Architecture for the Optional Functionality of the Application

Figure 4.3 outlines the process of the system working with the optional functionality to generate a ML model script. There is an option to upload a user defined ML Model Specification. The application creates a template which contains the game information, the model and the user defined ML model parameter. This template is then utilized to generate a script based on the parameters within it. This script is processed on the emulator with the specified game and ultimately provides users with the final results.

Chapter 5

Evaluation Strategy

This chapter describes methods to evaluate the functional and non-functional system and user requirements stated in chapter 3. This chapter is divided into two parts, the usability assessment and system assessment. The usability assessment provides evaluation techniques for the evaluation of user requirements. The system assessment outlines the evaluation techniques for the evaluation of system requirements.

5.1 Usability Assessment

The usability assessment will be conducted on a wide range of users who have a general knowledge about artificial intelligence. Gamers and non-gamers would be chosen for this assessment. Through the use of this, we will be able to evaluate the working of all functional and non-functional user requirements. The usability assessment will be based on the results of three survey forms, per person. All results will be anonymized. Each participant shall be allowed to withdraw from the survey at any given time, and their results will be discarded. A consent form is given before the evaluation begins which informs the user of the purpose of the evaluation, how the evaluation will be conducted and usage of results after the evaluation is completed. The surveys are divided as follows:

- **Pre-Survey**

  Acquire information about participants, primarily whether or not the participant is a gamer and if the participant has any knowledge or interest in artificial intelligence.

- **Questionnaire**

  The participant will be asked to conduct a series of tasks on the user interface of the application. The tasks can be selecting a game, a machine learning model and so on. We will record the difficulty of carrying out a task and the time taken to complete a task. The results will assist in the evaluation of the user functional requirements.

- **Post-Survey**

The participant shall be asked about their experience with the application. The questions will be based on the interactivity, usability and the quality of the user interface of the application. The results retrieved will contribute towards the evaluation of the user non-functional requirements.

5.2 System Evaluation

Each mandatory functional requirement will be evaluated using test cases in order to assess the fulfillment of the following aspects:

Completeness – The requirement must be successfully implemented.
Soundness – The requirement showcases expected performance.
Correctness – The requirement appropriately implemented.

The optional requirement to generate machine learning model script will be evaluated by comparing the performance of the generated script with the manual script for atleast two games.

There is one non-functional system requirement called functional scalability. Functional scalability, will be assessed through adding extra functionalities such as importing configurations and exporting results in a CSV file format. Such features should be implemented with ease.

Chapter 6

Project Management

This chapter covers the methodology used for the project. A Gantt chart is included which describes the schedule of important tasks throughout the provided project completion period. The risk analysis section outlines risks and possible avoidance, minimization and mitigation strategies for each risk.The final section gives a brief overview of possible professional, ethical, social and legal issues related to the project.

6.1 Project Methodology

In the initial stage, an efficient project methodology must be elected for smooth and efficient development of the project. An Agile Methodology known as Scrum shall be utilized to develop this project until completion.
Scrum teams can be divided into two primary roles. First being a Scrum Master, whose role is similar to that of a coach or mentor. Second is a Product Owner, whose role is to help build a feasible and well planned product [15]. In regards to this project, the two roles have been assigned as follows:

- **Scrum Master** - Supervisor

  The supervisor takes on the role of a Scrum Master by verifying that the developer/researcher is free of any doubts and distractions which may directly affect the progress of the current task.

- **Product Owner** - Supervisor and Users

  The supervisor adopts the role of a product owner by ascertaining the successful completion of required functionalities. The users are also considered as a part of this role since they play a direct role to the development of the user interface through user experience feedback.

The scrum process has three primary artifacts. They are namely the product backlog, the sprint backlog, and the burndown charts. The product backlog holds a list of functionalities to be added to the system. The product backlogs are prioritized to be able to work on the most critical features first. One of the most efficient ways to create a sprint backlog is to store user stories. User stories are brief functional descriptions from a users perspective. A

sprint backlog is created during the planning phase of meetings, which encompasses tasks which need to be done in order to successfully deliver the committed functionality for the current sprint. Burndown charts are used to visualize the amount of work remaining in a release or a sprint. This can prove to be effective in order to view that the plan is being executed as per schedule.

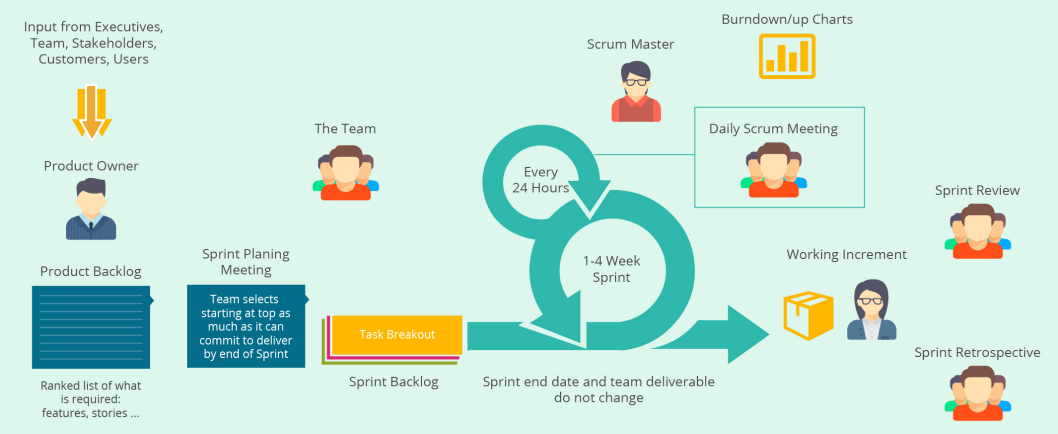


Figure 6.1: Scrum Process [16]

The tasks are split into a small set of deliverables. Each task is to be delivered iteratively within a fixed time frame. These are also known as timeboxed iterations or sprints [16]. Each delivered task has a potential of being shipped out as a working increment. Meetings are to be held on a weekly basis with the Scrum Master to make sure that development is being carried out as planned. A great example of a Scrum process is shown in figure 6.1. Upon the release of the first working prototype, users shall be asked to provide feedback on changes expected, if any. These changes are taken into account, and another iteration of the prototype must be released to show the effective implementation of the changes.

6.2 Project Plan

The main objectives are broken down into simpler and smaller tasks which are represented in two Gantt Charts. The first chart (figure 6.2) is related to the first deliverable, while the second chart (figure 6.3) is related to the final deliverable.

	Task Name	Duration	Start	Finish
1	**Project Plan for Application to analyze ML in Games**	228d	09/18/18	05/03/19
2	**Deliverable 1**	66d	09/18/18	11/22/18
3	Gather Topic Related Research Papers	3d	09/18/18	09/20/18
4	Summarize Papers	17d	09/21/18	10/07/18
5	Finalize Aim and Objectives	1d	10/08/18	10/08/18
6	Generate Requirements	7d	10/09/18	10/15/18
7	Produce an Evaluation Strategy	7d	10/16/18	10/22/18
8	Devise a Risk Management Plan	7d	10/23/18	10/29/18
9	Draft Deliverable 1 Documentation	15d	10/30/18	11/13/18
10	1st Draft Review by Supervisor	1d	11/14/18	11/14/18
11	Ethical Form Submission	1d	11/15/18	11/15/18
12	Modify Documentation based on Feedback	5d	11/16/18	11/20/18
13	2nd Draft Review by Supervisor	1d	11/21/18	11/21/18
14	Submission of Deliverable 1	1d	11/22/18	11/22/18
15	**Preparation of Semester 1 Exams**	18d	11/23/18	12/10/18

Figure 6.2: Project Gantt Chart (First Half)

The first stage of the Gantt Chart as shown in figure 6.2 covers all aspects of deliverable 1. Soon after deliverable 1, the Gantt chart shows a break for exams, followed by development of the User Interface for the application in figure 6.3.

	Task Name	Duration	Start	Finish
16	**Develop User Interface of Application**	22d	12/11/18	01/01/19
17	Explore Programming Languages for GUI	5d	12/11/18	12/15/18
18	Learn the GUI Part of the PL	3d	12/16/18	12/18/18
19	Design Application interface	4d	12/19/18	12/22/18
20	Develop the Interface	10d	12/23/18	01/01/19
21	**Develop Functionality of Application**	34d	01/02/19	02/04/19
22	Learn to Integrate Script with UI Language	6d	01/02/19	01/07/19
23	Generate a Script Snippet to Open Emulator	7d	01/08/19	01/14/19
24	Generate a Script Snippet to Open Game ROM	7d	01/15/19	01/21/19
25	Generate a script snippet to run ML algorithm	10d	01/22/19	01/31/19
26	Add User Settings to Script	4d	02/01/19	02/04/19
27	**Implementations of Extra Functionalities**	30d	02/05/19	03/06/19
28	Develop Generic Machine Learning Model	15d	02/05/19	02/19/19
29	Research on Improvements to existing ML for games	15d	02/20/19	03/06/19
30	**Usability Evaluation and Document**	15d	03/07/19	03/21/19
31	**Documentation for Deliverable 2**	33d	03/22/19	04/23/19
32	Drafting documentation	19d	03/22/19	04/09/19
33	1st Document Review	1d	04/10/19	04/10/19
34	Update Document	6d	04/11/19	04/16/19
35	2nd Document Review	1d	04/17/19	04/17/19
36	Final Document Changes	5d	04/18/19	04/22/19
37	Submission of Deliverable 2	1d	04/23/19	04/23/19
38	**Preparation of Semester 2 Exams**	7d	04/24/19	04/30/19
39	**Preparation for Poster Session**	10d	04/24/19	05/03/19
40	Creating and Designing Poster	8d	04/24/19	05/01/19
41	Poster Session	1d	05/02/19	05/02/19
42	Submission of Poster	1d	05/03/19	05/03/19

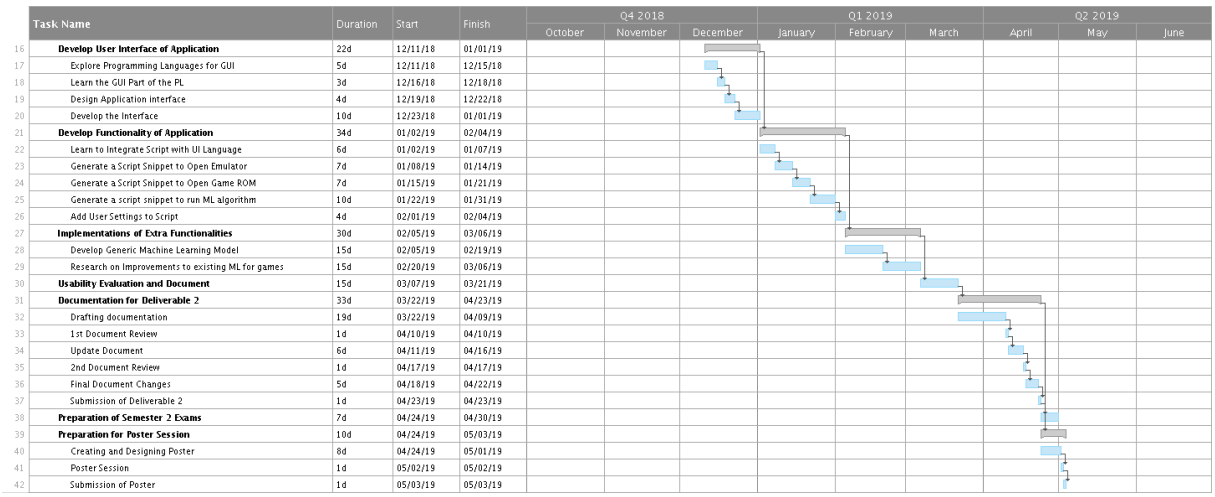


Figure 6.3: Project Gantt Chart (Second Half)

Once the User Interface is developed, focus is given to attaining functionalities of the application, as shown in figure 6.3. Optional implementations of extra functionalities are approached given that there is no delay with the development of the User Interface and the functionality.

Once the functionalities have been implemented, user evaluation is done on the User Interface of the application. This is then followed by drafting the documentation for deliverable 2, and finally concluded with designing a poster for the poster session.

6.3 Risk Management

Project development involves risks which must be taken into consideration during the initial stages of project planning. Risks tend to affect the time it takes for a developer to complete a certain task which could prove to be troublesome. The following subsections clearly identify possible risks for this project, and provide suitable ways to avoid and minimize such risks.

6.3.1 Risk Analysis

The first stage of the risk management process is the identification of risks. We categorize our risks under six pre-defined categories. The categories are as follows:

- **Technology**

  These risks are caused by consequent software or hardware issues raised during development of a system.

- **People**

  These risks are relevant to individuals whose actions may have a significant impact to the system.

- **Organizational**

  These risks are stated in regards to the environment in which the project/software is being employed.

- **Tools**

  These risks are associated with tools used to support the development of a system.

- **Requirements**

  These risks are brought forward through changes in user demands and expectations, followed by updates to the modified requirements.

- **Estimation**

  These risk emerge from estimation of resources needed to create a system.

Upon the identification of risks, the next step is to find the possibility of the risks occurring, and to identify the probable impact on the development of the system if any were to occur.

Risks are separated into the three different levels of priority:

Risk Priority	Color
High	
Moderate	
Low	

Table 6.1: Risk Priority

Using the above table 6.1, we can identify and rank risks as follows:

Risk No.	Risk	Risk Type	Impact	Probability	Description
1	Loss of important project associated data and files	Technology	Catastrophic	Low	Loss of files due to data corruption or accidental deletion.
2	Underestimating the time required to complete the project	Estimation	Catastrophic	Moderate	The estimated project completion time cannot be attained.
3	Conflicting coursework deadlines	Organizational	Catastrophic	High	Project deadline collides with coursework deadlines.
4	Insufficient online resources	Tools	Serious	High	Scarce documentation of supplementary material.
5	Unidentifiable memory locations	Technology	Catastrophic	High	Cannot identify game information in the memory.
6	Unfamiliarity of programming languages	Tools	Tolerable	Moderate	Lack of knowledge in the key/core programming language.
7	Future modifications of requirements	Requirements	Tolerable	Moderate	Requirements are susceptible to changes and/or updates at a later stage.
8	Lack of communication	People	Serious	Moderate	Miscommunication between Supervisor and Student.
9	Conflicting schedules with supervisor	Organizational	Tolerable	Moderate	Supervisor unavailable to provide feedback at crucial stages of project deliverable.
10	Lack of memory capacity of host system	Technology	Tolerable	Moderate	Not enough memory to run the algorithm on the game.

11	Emulators do not support script(s)	Tools	Tolerable	Low	Unable to run script in a specific emulator.
12	Buggy and confusing User Interface	Tools	Tolerable	Low	User struggling to use the API

Table 6.2: Risk Analysis Table

6.3.2 Risk Planning

In the following subsection, every risk identified and ordered (based on priority) in table 6.2 are analyzed for avoidance and minimization. Necessary measures are listed for each risk to reduce the effect of impact.

There are three strategies which are taken into account:

- **Avoidance:** Necessary measure to be taken to prevent a risk from ever occurring.
- **Minimization:** Decreasing the affects of a risk to a feasible level.
- **Contingency:** Taking into account the worst possible scenario of a risk occurring, and developing an executable backup plan.

Risk 1: Loss of important project associated data and files

- **Avoidance**

  Use Version Control environments similar to Github to backup files and settings.

- **Minimization**

  Perform regular backups to local and remote systems.

- **Contingency**

  Redownload latest possible version of the file from backup sources.

Risk 2: Underestimating the time required to complete the project

- **Avoidance**

  Be up to par with the project plan and efficiently follow the scrum process through the delivery of an iterated prototype every week.

- **Minimization**

  Work on the project regularly and attend weekly meetings with the supervisor.

- **Contingency**

  Allocate more time for the project by modifying the initial project plan.

Risk 3: Conflicting coursework deadlines

- **Avoidance**

  Resourcefully allocate time between project and courseworks.

- **Minimization**

  Inform supervisor of possible conflict of coursework deadlines.

- **Contingency**

  Allocate time on all courseworks based on difficulty.

Risk 4: Insufficient online resources

- **Avoidance**

  Research on relevant tools which have active online support.

- **Minimization**

  Use tools which have relevant documentation, and an active developer community.

- **Contingency**

  Inform supervisor and find alternate solutions with bare minimum documentation to replace/support currently implemented tools.

Risk 5: Unidentifiable memory locations

- **Avoidance**

  Find relevant documentation in regards to object addresses of games in memory.

- **Minimization**

  Use tools to identify memory addresses of objects of relevant games.

- **Contingency**

  Only use games which have a well documented range of objects linked to memory addresses.

Risk 6: Unfamiliarity of programming languages

- **Avoidance**

  Identify languages to be used, and get acquainted with them through examples.

- **Minimization**

  Reduce the amount of time required to learn the relevant languages.

- **Contingency**

  Learn the relevant programming languages while developing the project.

Risk 7: Future modifications of requirements

- **Avoidance**

  Complete the primary requirements of the project at early stages in order to accommodate changes to requirements.

- **Minimization**

  Present an iteration of the project on a regular basis, and retrieve feedback from the supervisor.

- **Contingency**

  Notify the supervisor of changes which need to be made, and incorporate new requirements to the project plan.

Risk 8: Lack of communication

- **Avoidance**

  Book 30-60 minutes of weekly meetings with the supervisor.

- **Minimization**

  Ensure that meetings happen as per schedule, and that the supervisor is aware of every phase that the project is in.

- **Contingency**

  The student must raise concerns with the supervisor, and have a discussion about what to do next.

Risk 9: Conflicting schedules with supervisor

- **Avoidance**

  Rearrange meetings as per the convenience of student and supervisor.

- **Minimization**

  Regularly attend meetings.

- **Contingency**

  Ask questions over email, and provide a draft of the report for feedback via email if supervisor is unavailable.

Risk 10: Lack of memory capacity of host system

- **Avoidance**

  Use a system which has enough RAM to run the machine learning algorithms.

- **Minimization**

  Reduce the number of running background applications, and run algorithm in batches.

- **Contingency**

  Use either the University desktops, or request access from supervisor to use the supercomputer.

Risk 11: Emulators do not support script(s)

- **Avoidance**

  Conduct preliminary research on the scripts supported by each emulator, and vice versa.

- **Minimization**

  Use emulators which have extensive online support.

- **Contingency**

  Inform the supervisor of the issue and search for alternative emulators.

Risk 12: Buggy and confusing User Interface

- **Avoidance**

  Conduct test to avoid a buggy interface and user experience. Gather feedback on initial design of the interface.

- **Minimization**

  Base design decisions on user feedback.

- **Contingency**

  Find alternative methods to resolve bugs, and provide a detailed user manual for the interface.

6.4 Professional, Ethical, Social and Legal Issues

During the process of developing the project, we could encounter specific professional, ethical, social and legal issues. These issues must be determined and avoided in order to ensure success of the project. This section identifies and describes these issues.

Professional Issues

Participants who fail to show professional behavior during the evaluation surveys could prove to provide inaccurate results. This professional issue needs to be addressed by ensuring that all the participants display professional behavior while the survey is being conducted.

Ethical Issues

Before surveys are conducted, users are asked to sign a consent form which states how the data is going to be used. The data attained from the surveys and questionnaires would be anonymous, and would only be used to draw results and conclusions based on average feedback.

Social Issues

One notable social issue is similar to the above stated ethical issue. This issue is in concern with privacy. We eliminate such issue by making sure no names or personal features are recorded throughout the usability assessment without participant consent.

Legal Issues

Legal issues would arise if tools (e.g. emulators) are utilized or modified for which exclusive permission is not given. This is avoided in this project since the emulators primarily use the MIT, GNU General Public License (GPL) and Lesser General Public License (LGPL) licensing standards.

Bibliography

[1] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, Jan. 2018, ISBN: 9783319635187.

[2] J. Togelius, "Ai researchers, video games are your friends!" In *Computational Intelligence: International Joint Conference (IJCCI)*, vol. 669, 2016. arXiv: 1612.01608 [cs.AI].

[3] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, "A neuroevolution approach to general atari game playing," *IEEE Transactions on Computational Intelligence and AI in Games*, 2013.

[4] T. Murphy VII, *The first level of super mario bros. is easy with lexicographic orderings and time travel... after that it gets a little tricky.* 2013.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *CoRR*, 2012. arXiv: 1207.4708.

[6] M. McPartland and M. Gallagher, "Reinforcement learning in first person shooter games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 1, 2011. DOI: https://doi.org/10.1109/TCIAIG.2010.2100395.

[7] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. P. Liebana, "Deep reinforcement learning for general video game ai," Jun. 2018.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, 2013. arXiv: 1312.5602.

[9] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02.

[10] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," *CoRR*, 2016. arXiv: 1609.05521.

[11] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *CoRR*, 2016. arXiv: 1611.01779.

[12] J. L. Nielsen, B. F. Jensen, T. Mahlmann, J. Togelius, and G. Yannakakis, "Ai for general strategy game playing," *Handbook of Digital Games*, Mar. 2014. DOI: https://doi.org/10.1002/9781118796443.ch10.

[13] K. Kunanusont, S. M. Lucas, and D. P. Liebana, "General video game ai: Learning from screen capture," *CoRR*, 2017. arXiv: 1704.06945.

[14] W. Woof and K. Chen, "Learning to play general video-games via an object embedding network," *IEEE Computational Intelligence and Games Conference*, Jun. 2018. DOI: https://doi.org/10.1109/CIG.2018.8490438.

[15] *Scrum*, Mountain Goat Software. [Online]. Available: http://www.mountaingoatsoftware.com/agile/scrum (visited on 11/19/2018).

[16] *Scrum and kanban – are they that different after all?* Perfectial. [Online]. Available: https://perfectial.com/blog/scrum-and-kanban-are-they-different/ (visited on 11/19/2018).