2007

Using machine learning techniques to create AI controlled players for video games

Bhuman Soni

Edith Cowan University

Recommended Citation


This Thesis is posted at Research Online.
https://ro.ecu.edu.au/theses_hons/1294
Edith Cowan University

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study.

The University does not authorize you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following:

- Copyright owners are entitled to take legal action against persons who infringe their copyright.

- A reproduction of material that is protected by copyright may be a copyright infringement. Where the reproduction of such material is done without attribution of authorship, with false attribution of authorship or the authorship is treated in a derogatory manner, this may be a breach of the author’s moral rights contained in Part IX of the Copyright Act 1968 (Cth).

- Courts have the power to impose a wide range of civil and criminal sanctions for infringement of copyright, infringement of moral rights and other offences under the Copyright Act 1968 (Cth). Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Using Machine Learning Techniques to Create AI Controlled Players for Video Games

Dissertation submitted in partial fulfilment of the requirements for the degree of

Bachelor of Computer Science (Honours)
Honours Thesis: Semester 2, 2007

By: Bhuman Soni
Student No: [Redacted]
Supervisor: Associate Professor Philip Hingston
Faculty: Computing, Health and Science
Institution: Edith Cowan University (ECU)
USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.
COPYRIGHT AND ACCESS DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

(i) incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) contain any material previously published or written by another person except where due reference is made in the text; or

(iii) contain any defamatory material.

Signed......

Date......31/01/2008......................
Abstract

This study aims to achieve higher replay and entertainment value in a game through human-like AI behaviour in computer controlled characters called bots. In order to achieve that, an artificial intelligence system capable of learning from observation of human player play was developed. The artificial intelligence system makes use of machine learning capabilities to control the state change mechanism of the bot. The implemented system was tested by an audience of gamers and compared against bots controlled by static scripts. The data collected was focused on qualitative aspects of replay and entertainment value of the game and subjected to quantitative analysis.
Acknowledgements

I have to start of by a big thanks to my supervisor Associate Professor Philip Hingston for putting up with me and being incredibly patient through my entire honours journey.

Another important person, who’s helped me throughout is my Colombian friend Gloria Arenas. Thanks for always being there for me, words cannot express how thankful I am for all that you have done.

Also thanks to my parents and sisters whose blessings and wishes have kept my hopes up at all time.

Thanks to all my friends and the people who participated in my experiment. I was running my experiment at a busy time of the semester, most of my participants had work commitments and uni commitments and yet they took the time for participation and I really appreciate it.

And last but far from the least I thank my god Lord Krishna for I truly believe in him and its because of his blessings I have been able to reach this stage.
# Table of contents

1. 1. ......................................................................................................... Introduction
    8

2. 2. Background to the study
    10
    2.1 AI used in games ............................................................... 10
    2.1.1 Artificial neural network ................................................. 10
    2.1.2 Evolutionary algorithms ............................................... 11
    2.1.3 Finite State Machines ................................................ 12
    2.1.4 Fuzzy State Machines (FuSM) ....................................... 12
    2.2 First person shooter genre .................................................. 13
    2.3 Initial Scope of the research .............................................. 14

3. 3. The significance of the study
    16
    3.1 Purpose of the Study ....................................................... 16
    3.2 Research questions ......................................................... 16

4. 4. Review of the Literature
    17
    4.1 AI and Games ................................................................. 18
    4.2 Machine Learning in Video Games ................................... 20
    4.3 Research similar to the proposed study ............................ 22

5. 5. Theoretical Framework
    25

6. 6. Implementation
    26
    6.1 The AI technique ............................................................. 26
    6.2 The learning algorithm .................................................... 27
    6.3 The game ........................................................................... 30
    6.3.1 Some key concepts of Unreal Tournament 2004 ............ 31
    6.4 Testing procedure ........................................................... 31
    6.4.1 Map selection ............................................................ 32

7. 7. The implemented system
    33
    7.1 FeedforwardBot ............................................................. 34
    7.2 RecurrentBot .................................................................. 36
7.3 UT2004 modifications for the study ..................................................... 38
7.4 The Recorder.................................................................................... 39
  7.4.1 The state changes of the data recorder .......................................... 39
  7.4.2 Determining the player action...................................................... 40
  7.4.3 Gathering Training data ............................................................... 41
7.5 Training the bots............................................................................... 42
  Equalizing the dataset.......................................................................... 42
7.6 The ANN state change mechanism ...................................................... 43
  7.6.1 The ANN bot............................................................................... 44
  7.6.2 The game and ANN communication process.................................... 44
8. 8. Implementation phases

  8.1 Preparation Phase ............................................................................. 47
  8.2 First Implementation Phase................................................................. 47
  8.3 Second Implementation Phase............................................................. 47
  8.4 Third Implementation Phase ............................................................... 47

9. 9. The questionnaires

  9.2 Bot specific questionnaire................................................................ 49
  9.3 Bot rating Questionnaire ................................................................. 49
  9.4 Data analysis techniques.................................................................... 50

10. 10. Data analysis: bot specific questionnaire

  10.1 Unpredictability in bot behavior......................................................... 52
  10.2 Human-like attributes of the bot......................................................... 54
  10.3 Bot’s skills and intelligence ............................................................... 55
  10.4 Bot difficulty level............................................................................. 57
  10.5 Bot combat skills.............................................................................. 58
  10.6 Entertainment value of the bot............................................................ 59
  10.7 Overall experience playing against the bot.......................................... 61

11. 11. Data Analysis: bot rating questionnaire

  11.1 Question 1 ...................................................................................... 63
  11.2 Question 2 ...................................................................................... 64
  11.3 Question 3 ...................................................................................... 65
  11.4 Question 4 ...................................................................................... 66
  11.5 Question 5 ...................................................................................... 67
Tables of figures

Figure 1: An illustration of a simple neuron (Source: Negnevitsky, 2002, p.166)
Figure 2: The state changes of a simple electric bulb
Figure 3: Player and game interaction
Figure 4: State machine example (Source: Meyer, 2003)
Figure 5: Overview of the implemented system
Figure 6: FeedforwardBot ANN structure
Figure 7: The structure of the RecurrentBot ANN
Figure 8: State changes during recording
Figure 9: Training data collection process
Figure 10: State changes of the bot during gameplay
Figure 11: Data flow between ANN and bot during gameplay
Figure 12: Unpredictability in bot behaviour plot
Figure 13: Human-like attributes plot
Figure 14: Bot skills and intelligence plot
Figure 15: Bot difficulty
Figure 16: Bot combat skills
Figure 17: Entertainment value of the bot
Figure 18: Overall experience of playing against the bot
Figure 19: Enjoyability
Figure 20: Least entertaining
Figure 21: Most human-like behaviour
Figure 22: Replayability
Figure 23: Time efficiency
1. Introduction

Artificial intelligence (AI) has been an active area of computer science research for the last five decades. Over the last few years, interactive computer games have gained substantial popularity among AI researchers. As Laird & Van lent (2001) suggests, games are

- a cheaper alternative to robotics;
- becoming increasingly realistic with the ever advancing graphics technology;
- flexible enough to give the researchers more time to experiment;

The success of most games however has depended largely on the game's graphics technology, and as Fairclough, Fagan, Namee, & Cunningham (2001) claim, AI was mostly an afterthought for most game developers. This however, is not the case anymore, as the graphics race seems to have run its course, with the developers searching for newer ways to make their games more engaging. One such way perceived by most game developers is AI (Laird & Van lent, 2000).

The increasing popularity of online games is a strong indication of an ever increasing need of human-like AI in computer games (Yannakakis & Hallam, 2005). Currently, as Sweetser (2002) states most developers apply rule based approaches such as Finite State Machines (FSM) and Fuzzy State Machines (FuSM), to model AI behaviour in their games. Despite the effectiveness of these approaches, they are often limited in their implementation, which results in somewhat predictable AI behaviour (Bakkes, Spronck & Postma, 2004).

In this study we suggest a quantitative approach to analysing qualitative data. We quantitatively investigate how qualitative aspects of entertainment and replayability are affected by game AI.

This study identifies the use of machine learning (ML) as an AI technique in a game to create more human-like AI behaviour, which in turn leads to a more enjoyable and replayable game. The implemented AI model learns by observation of expert player play and controls the state change mechanism of a bot during combat. This study only focuses on the combat aspect of the gameplay therefore
the trained AI model only controls the state change mechanism while the bot has engaged in combat.

Throughout this document, the term player refers to the human player and the term bot refers to the AI controlled character.
2. Background to the study

ML is a branch of AI that entails training a computer to perform tasks that would otherwise require human intelligence. ML can be achieved through techniques such as ANN and Evolutionary Algorithms (EA) (Bakkes, Spronck & Postma, 2004).

2.1 AI used in games

2.1.1 Artificial neural network

An artificial neural network (ANN) is a ML technique inspired by the structure of the human brain. Analogous to the human brain, an ANN is made up of processing elements called neurons that are interconnected through links. Each link has a numerical weight value associated with it which is the means of long-term memory of the ANN (Stergiou & Siganos, 1997). Refer to Figure 1.

A neuron in an ANN may contain more than one set incoming input connections, but only one outgoing output connection. To generate the output a neuron has to be activated using an appropriate activation function (Stergiou & Siganos, 1997).

![Figure 1: An illustration of a simple neuron](Source: Negnevitsky, 2002, p.166)

One of the activation functions is called sigmoid activation function, which takes an input value (which can be between 0 and infinity) and changes it to a number between 0 and 1 (Negnevitsky, 2002, p.166).
With the ability to learn, an ANN learns during its training phase, when it is presented with a set of training examples. During training, the ANN learns by updating the weight values associated with the connections. Once trained, an ANN can generalize an output corresponding to an input pattern based on what it has learned from the training examples (Stergiou & Siganos, 1997).

The ANN learning can either be supervised or unsupervised. With supervised learning, the correct output for a particular set of inputs is already known; therefore the AI is trained to achieve that output. With unsupervised learning, the ANN does not know the correct output, but carries out a trial and error process in order to achieve the desired output (Spronck, 2005).

The type of ANN learning used in this study is supervised offline learning. Learning can be either offline or online. Online learning (OL) sees the AI adapt in its environment in real-time, where the AI tries to adapt to the environment it is in. Offline learning entails the AI learning without any intervention and by itself (Ponsen, 2004).

Commercial application of an ANN can be seen in the commercial game Battlecruiser: 3000 AD, where virtually every in-game NPC is controlled by an ANN (Woodcock, 2007).

### 2.1.2 Evolutionary algorithms

Evolutionary algorithms (EA) refer to a branch of computational algorithms inspired by Charles Darwin's theory of biological evolution through natural selection and survival of the fittest. The algorithms include genetic algorithms (GA), evolution strategies and genetic programming, all of which evolve a given population using selection, mutation and reproduction (Negnevitsky, 2002, p. 217). According to Johnson and Wiles (2004), GA is the most commonly used type of EA in games.

An example of a game using GAs to model opponent behaviour is a real-time strategy game titled Cloak, Dagger and DNA (CCD). In the game both the player and the NPC have DNA strands which keep track of its performance in each battle. The game lets the user evolve DNA strands by placing them in battle against each other (Woodcock, 2007).
ANN and EA are still not widely used for commercial games, as most developers consider them 'experimental' and too 'risky' for their tight development schedules (Woodcock, 1998). Therefore some of the more commonly implemented AI techniques in games are FSM and FuSM.

### 2.1.3 Finite State Machines

A finite state machine (FSM) is a rule-based AI technique composed of states. A state in an FSM can change to another state, which is referred to as a state transition. A state transition is normally caused by an event, which serves as an input for the state transition. A state is composed of a set of actions, which are executed when the system is in that state (Meyer, 2003). As Fu & Houlette (2004) describe, an FSM is a concise, nonlinear description of how an object can change its state over time, possibly in response to events in its environment. The state change depends on the input and the state transition function. The diagram below represents a simple state transition diagram for an electric bulb:

![State Transition Diagram for an Electric Bulb](image)

**Figure 2: The state changes of a simple electric bulb**

With respect to games, each state in a FSM represents behaviour or a set of behaviours. A game object normally consists of several states and it survives in the game environment by changing its state according to the environment.

Successful implementation of FSM can be seen in the commercial first person shooter (FPS) Half-life released in 1999 by Vivendi universal. (Woodcock, 2007)

### 2.1.4 Fuzzy State Machines (FuSM)

Unlike Boolean logic, fuzzy logic deals with degrees of membership of truth rather than crisp membership i.e. with fuzzy logic something can be partially true or partially false. For example, if the height of a man is represented with Boolean logic he would either be tall or short, whereas if fuzzy Logic is used the man can
be slightly tall or slightly short (Negnevitsky, 2002, p. 89). FuSM is a combination of FSM and fuzzy logic, whereby the system can be partially in a state, as opposed to either being entirely in that state or not.

A FuSM is a combination of fuzzy Logic and FSM, resulting in fuzzy states instead of the crisp states of an FSM. FuSM has been successfully used in games such as Unreal Tournament and Civilisation: Call to Power (Johnson & Wiles, 2001).

2.2 First person shooter genre

The computer game used in this study is the popular FPS Unreal Tournament 2004 (UT2004) (Dawes & Hall, 2005).

FPS is a game genre where the player perceives the game environment from the first person perspective; the player can see the entire environment except its own character. Most FPSs have two primary game types, single-player and multi-player. In single-player, the player plays the game against computer controlled characters known as bots, which can be either friendly or hostile relative to the game. In single-player games, the players often advance through a story by completing a set of objectives that usually involve interaction with other bots. The multi-player mode of an FPS consists of several game modes all of which require human players competing alongside or against each other with or without the help of bots. The players participate in multi-player games via internet connection.

Of the several existing game modes in UT2004, the mode used in this study is Deathmatch. In a Deathmatch game, the objective is to score the highest number of points within the specified time by killing other bots or players. A player or bot character is killed when it looses all its allocated health points. Killing another player's character or a bot requires shooting at them. A Deathmatch game takes place on a map (which can also be referred to as a level), which is the environment where the Deathmatch takes place. There are useful items scattered throughout the map that can be used by the player to gain a competitive advantage. Some of these useful items are

- Weapons: there are ten different weapon types in the UT2004 map used for this study.
- Health pickups: used to increase the player health level.
- Ammo: weapon specific ammunition.
• Defence/offence power-ups: these items give the characters special abilities such as extra defence and strength boost.

Mod

A mod or modification is often a fan made enhancement to an existing game. Depending largely on the popularity of the game, developers often supply mod tools for their games. Often released on the internet, a mod can be an entirely new game or the existing game with new characters and levels (Jagger, 2004).

2.3 Initial Scope of the research

The initial scope of the study was to compare and contrast online learning with offline learning and to investigate its effects on the game’s replay and entertainment value. The objective was to create a bot that learns offline and another bot that learns during gameplay i.e. online learning. The purpose of using online learning was to create a bot that changes its strategy and displays unpredictable human-like behaviour.

Online learning was to be implemented through the means of using, the reinforcement learning algorithm Q-Learning.

Reinforcement learning is a process through which an agent learns the optimal action in a given environment by interacting with it. E.g. an agent in a given environment executes an action and the environment gives the agent a reward as feedback for the action executed. The reward is either positive or negative, the RL algorithm than establishes a policy, through which it aims to maximize the possibility of positive reward. The policy is known as an action selection policy, which controls the decisions made by the RL agent (Kaelbling, Littman & Moore, 1995, p. 1).

However during the second implementation phase of the research it was evident that it was not possible to achieve the initial scope within the given time frame. This was owing to the fact that the amount of time needed for the second implementation phase was significantly underestimated. Another factor was, given the nature of online learning, fifteen minutes of gameplay may not have been sufficient for the AI to learn and improvise. Therefore the scope was changed to comparing and contrasting offline learning by offline learning using randomised action selection. It was hypothesized that, the use of a randomised
action selection technique would add a level of unpredictability and randomness to the bot that would help achieve the study objectives.
3. The significance of the study

As Bauckhage, Thurau, Sagerer, Bernd, & Gerald (2003) assert, FSMs often appear artificial as they always cycle through a fixed set of actions, which results in repetitions that cause predictable AI behaviour. One of the limitations of FSM and FuSM is their rule-based nature which limits them to the situations anticipated by the programmer. Hence there is a high chance that a player may discover patterns in its behaviour, through repeated play. Once the players discover weaknesses in the AI behaviours, they can easily exploit them to their advantage. This makes the game too easy and thus hampers the gameplay experience for the player (Bakkes, Spronck & Postma, 2004).

If this study suggests that the use of ML technique such as ANN would result in high replay and entertainment value for the game, it would open a set of possibilities for future research, where a similar approach could be used to:

- Develop support characters in a Role-Playing-Game (RPG) scenario;
- try the proposed AI technique for games of different genres;
- A similar approach can be adopted by game developers for game AI.

3.1 Purpose of the Study

The purpose of this study is to investigate whether human-like bot behaviour achieved through the use of a ML technique can result in high replay and entertainment value for the game.

3.2 Research questions

Can bots controlled by an ML AI technique result in a more replayable and an entertaining game, when compared to bots controlled by static scripts?

Components of the above question are:

- A comparison between bots controlled by offline learning, offline learning using randomised action selection and static scripts, what results in the most human-like behaviour for the bot from the gamer's perspective?
- Does the proposed artificial intelligence technique increase the entertainment value for the gamer?
- Can the use of the proposed artificial intelligence technique increase the replay value for the gamer?
4. Review of the Literature

The bot AI technique in the game which controls the bots various actions...

The AI technique sends and receives sensory information from the bot in the game world throughout the game.

Figure 3: Player and game interaction

A bot is a computer controlled character that is part of the game. A bot is the name given to a computer controlled character, controlled by the game’s AI technique. The AI technique in a game acts as a controller for the bot’s behaviour. Throughout the game, the bot sends sensory information received from the game’s environment and sends it to the AI technique, which analyses it and directs the bot accordingly.

A simple example illustrating this would be a bot that can be turned on or off.
As evident from Figure 4, the states in a FSM are mutually exclusive.

Like the example above, the bot exists in different states in the game world. Each state of a bot represents a set of actions, hence when the bot is in that state it performs a variety of actions programmed for that state. An example would be when the bot is in a running state and while running the bot spots a weapon. At this point, an event occurs which would notify the bot’s state machine and cause a state transition to a state in which the bot can pickup the weapon (Meyer, 2003).

4.1 AI and Games

Early research into AI and video games involved the use of ML techniques to create computer controlled players for traditional board games (Laird, 2001). There are many experiments that have been carried out in the past on AI in games, but listing down the details of each and every experiment is beyond the scope of this document. Examples listed below give an indication of what has been done in the past with respect to ML and games.

The Checker’s playing program by Arthur Samuel was one of the first programs to demonstrate ML and adaptive behaviour in 1959 (Samuel, 2000). Samuel’s program used the number of features of positions in checkers which have been deemed important by human experts to construct the evaluation function. The program learned by playing against itself and adjusting weights when necessary (Harley, 2002).
However the success of the experiment was limited as Samuel's program only won one match against R.W. Nealey in 1962, who at the time, was one of the nation's foremost players. Nealey defeated Samuel's program in a re-match and Samuel's program never won a single match after that (Chellipilla, 2000). As Harley (2001) states, "The challenge that Samuel left open to future researchers was to design a program that could invent its own features, i.e., one that could learn the game from scratch without human advice".

This was later known as the "Samuel-Newell" challenge, which was taken up by David Fogel, who designed his experiment Blondie24 in terms of the challenge (Harley, 2002). The purpose of Fogel's experiment was to create a program that teaches itself how to play checkers, without being pre-programmed with any information about the game (Chellipilla & Fogel, 2001). Blondie24 was built on an evolutionary ANN that evolved its weights using an evolutionary strategy. The ANN in Blondie24 uses the board position as its input and outputs a value which is used in a mini-max search. (Kendell & Willdig, 2001).

AI research has been carried out with many board games, chess and checkers being just two. A card game, of interest to some AI researchers is poker. According to Barone & While (1999), poker is a game of imperfect information, that is, the game has information which is hidden.

In 1999, Barone and While, in an attempt to improve on their previous effort at creating an adaptive poker player, used an evolutionary algorithm to evolve a poker player which adapted to its opponent's play style. The poker player also learnt its opponent's weakness in order to exploit it to its own advantage (Barone & While, 1999). It was through the above experiment, that Barone and While showed how evolving players can outperform static players (Barone & While, 1999).

Following Barone and While's experiment Kendell & Willdig (2001) adopted a rule-based approach at creating an adaptive poker player. They programmed a unique set of rules for each play style.

The above section listed some of the work carried out in the field of ML with the focus on board games. Moving on to the next section, descriptions of some ML experiments carried out on commercial video games are listed.
4.2 Machine Learning in Video Games

The following section contains a brief account of previous work in this field which is similar to the proposed study. This section focuses on ML experiments that have been carried out on games of genres other than FPS.

In 2002, Peter Spronck and his team aimed to improve the opponent intelligence in games by using an evolutionary algorithm (EA) to evolve an ANN offline. They used both feedforward ANN and a recurrent ANN to evolve an AI player. They used the space strategy game PICOVERSE designed for palmtops, to investigate the effects of their proposed technique. Picoverse is a space strategy game, where the player owns a small spaceship, and engages in missions that span the entire galaxy. The player’s tasks involve upgrading the ships and trading goods between planets (Spronck, Postma, & Sprinkhuizen-Kuyper, 2002).

Their evolved AI player successfully outperformed its scripted opponent and even discovered flaws in the script which could prove more useful than designing a new tactic (Spronck, Postma & Sprinkhuizen-Kuyper, 2002). In this experiment the evolved AI player was only tested for its fighting abilities and not for other aspects of the gameplay e.g. trading goods and upgrading spaceships.

According to Spronck, Postma, & Sprinkhuizen-Kuyper (2003), for unsupervised OL to be an effective game AI technique, the unsupervised online learning method has to be fast, effective, robust and efficient. They designed the experiment in terms of the four requirements and proposed the technique of Dynamic Scripting (DS). DS is an unsupervised online learning technique that mimics reinforcement learning for its learning process. DS maintains several rulebases for each opponent type in the game and each time an opponent is generated, the script governing its behaviour is created by selecting rules from the rulebase. The rules are selected based on a weight value associated with each of the rules (Spronck, Postma, & Sprinkhuizen-Kuyper, 2003).

To test the effectiveness of DS they implemented it in a Role-Playing-Game (RPG) simulation and in Neverwinter Nights by Bioware Corp (Belvings, n.d.). The results of both the above experiments indicate that the DS controlled characters outperform its static scripts controlled opponents (Spronck, Postma, & Sprinkhuizen-Kuyper, 2003). However DS occasionally took a long time to adapt which made it unacceptable as a game AI technique (Spronck, Sprinkhuizen-Kuyper & Postma, 2004).
After discovering the limitations of DS Spronck, Sprinkhuizen-Kuyper & Postma, (2004) implemented two methods for overcoming the problems of long learning times of DS. The two proposed methods were penalty balancing, whereby a better balance between penalty and rewards are ensured and history fallback, where DS shifts to a historic rulebase. The results of their experiments indicated both methods, when used in conjunction with each other significantly enhanced the performance of DS. Despite the effectiveness of the implemented improvements, DS still had the occasional, unacceptably long loading times (Spronck, Sprinkhuizen-Kuyper & Postma, 2004).

In 2004, Ponsen & Spronck applied DS to an RTS game scenario, to test how offline evolutionary mechanisms can improve the performance of adaptive game AI. To facilitate DS into an RTS, the game was divided into different states with DS rulebases representing each game state. The weight updates occur during the state transitions in the game and during the end of each game. The DS controlled team battled against opponents controlled by static scripts, each static script corresponded to a gameplay technique which according to Spronck & Ponsen (2004) is used by most human players while playing RTS games (Spronck & Ponsen, 2004).

The static scripts represented four tactics: two balanced tactics and two rush tactics. After the initial failure of DS to defeat the latter two techniques, they evolved the DS algorithm offline using an EA, which resulted in the DS succeeding in defeating all the four techniques (Spronck & Ponsen, 2004). Once again the experiment was based on evaluating the success of the proposed technique without taking into consideration the entertainment value which is a result of the AI technique.

Byeong, Sung, Yeong , & HA (2006) carried out their research into the use of ANN to control game characters by evolving an Intelligent Character (IC) in a fighting action game. They used a feedforward ANN trained with a reinforcement learning algorithm to control an IC. They proposed a scheme whereby their IC not only learns the game moves and rules from its opponents but also their action patterns and moving actions. They evaluated their proposed scheme in a custom made fighting game. The results of their experiment indicate that the opponent performs well against random party characters thus demonstrating the feasibility
of using slow techniques such as ANN, in fast action games (Byeong, Sung, Yeong & HA, 2006).

In contrast to the experiments mentioned above, Yannakakis & Hallam (2006) use ML to model player satisfaction (entertainment) in games. They used a fully connected multi-layered feedforward ANN and a fuzzy ANN to model the effect of challenge and curiosity levels to player entertainment. The feedforward ANN was trained using a GA and as indicated by their tests, the fittest ANN gets closer to the idea of human entertainment when compared to the fittest fuzzy ANN. through their experiment; they introduced a quantitative metrics of entertainment based primarily qualitative aspects (Yannakakis & Hallam, 2006).

Most of the experiments listed above, researchers have tried to create an intelligent character that outperforms its static scripts counterpart. However they have not investigated the effect of their AI technique on gameplay from an average player's perspective. The next section gives a brief account of some the ML experiments in the FPS genre.

4.3 Research similar to the proposed study

The Gamebots project started at the University of Southern California's Information Sciences Institute, aimed at turning the FPS Unreal Tournament into an AI test-bed for researchers. They successfully modified Unreal tournament to enable in-game characters being controlled via network sockets, by a program external to the game. The Gamebots project is open source and it's available for download through the Gamebots website (Adobbati et al., 2001).

Geisler (2002) successfully applied ML algorithms such as ANN, naïve bayes and decision trees to model player behaviour in an FPS. The algorithms learned a set of movement related combat behaviours by observation of expert player play. Geisler empirically evaluated the three ML techniques and concluded by recommending ANN for offline learning when compared naïve bayes and ID3.

Bauckhage, Thurau, Sagerer, Bernd, & Gerald (2003) identified bot programming as a learning task and therefore devised their experiment to promote their idea. One of their research goals was to show that learning by observation was possible by the means of an ANN. Using the game of Quake 2 they carried out many experiments with Self Organizing Maps (SOM) architecture. The results of those
experiments indicate the possibility of learning human-like behaviour using NN as a game AI technique (Bauckhage, Thurau, Sagerer, Bernd, & Gerald, 2003).

In 2004, Zanetti and Rhalibi proposed and applied an AI mechanism that uses ML techniques to achieve human-like behaviour in an FPS called Quake 3 Arena. They identified the three aspects of AI behaviour in an FPS, namely 'Move in fight', 'Route in the map' and 'Aim, shoot and Choose weapon' and implemented a ANN to learn each aspect. For the purpose of their study, they used a feedforward ANN trained using a GA. The result of their experiment however was an uncompetitive bot which was not fun to play against, due to the fact that the AI efficiently learned routing behaviours but failed to learn appropriate Aim, shoot and fight movement behaviours (Zanetti & Rhalibi, 2004).

Bakkes, Spronck & Postma (2004) proposed the TEAM adaptive mechanism to control behaviour of teams in team based FPS games. Their evolutionary algorithm based TEAM mechanism adapts online and coordinates the actions of an entire team rather than individual team members. The TEAM mechanism works by representing each game state as an FSM and creating an evolutionary algorithm for each state in the game. The evolutionary algorithm learns the optimal team behaviour for each state of the game (Bakkes, Spronck & Postma, 2004). In order to improve the performance of their TEAM mechanism they modified the original mechanism significantly. Unlike the original version, TEAM2 uses symbiotic learning, with best-response strategy, state based fitness function and a scaled roulette wheel selection function. In the experiment TEAM2 successfully outperformed its static opponents and won a comparative analysis with TEAM. However the learning performance of TEAM2 was somewhat slow and therefore Bakkes, Spronck & Postma concluded by stating the effectiveness of TEAM2 is relative to the game for which its being implemented (Bakkes, Spronck & Postma, 2005).

Vasta, Lee-Urban & Munoz-Avila (2006) implemented an online RL algorithm titled BLADE (Bounded Learning Algorithm for Domination Teams) for achieving winning policies in a Team FPS. BLADE is an online learning algorithm designed to control team actions and run continuously for multiple game instances. To prove the effectiveness of BLADE, they implemented it in Unreal tournament using the Gamebots distribution. They successfully demonstrate the effectiveness of BLADE at achieving winning policies and the ineffectiveness of discount rates common in reinforcement learning (Vasta, Lee-Urban & Munoz-Avila, 2006).
Dawes & Hall's (2005) initial attempt at creating a neural network within a FPS game ended in a result they deemed unsatisfactory. They identified the limitations inherent to the games scripting language as being the primary reason for the unsatisfactory outcome. Hence in their next experiment; they successfully devised an intermediary architecture independent of the games scripting language that enabled them to plug-in the bot's cognitive model into the game externally. Their goal was to create an easy to use test bed for AI researchers with which they can solely focus on programming the bots cognitive model, instead of game specific implementation details (Dawes & Hall, 2005).
5. Theoretical Framework

This study is based on the theory that game AI can impact its replay and entertainment value. A game such as Black & White proves that having an AI system that learns results in a highly entertaining game (Woodcock, 2007).

This study follows the hypothesis that a game where the AI behaviour is unpredictable and human-like can result in a more entertaining game with high replay value. In order to ensure that the above theory works in practice, the game AI must exhibit human-like unpredictable behaviour. Therefore learning by observation and using a randomised selection mechanism for action selection, the AI would exhibit unpredictable behaviour. In order to achieve learning, the proposed AI technique uses ML algorithms, thus making learning an important aspect of the research. The theory can be summarized as follows:

- The characteristics of human-like behaviour are unpredictability and randomness.
- By learning from a human, the AI would exhibit human-like behaviour.
- ML combined with randomised action selection would achieve human-like behaviour that makes the bot less predictable, which in turn achieves higher replay and entertainment value in a game.

We define the variables as follows:

- Predictability: the extent to which the player can or cannot make a correct guess at the bot's next move.
- Replay value: The desire to play the game again, as a result of enjoying it the first time.
- Entertainment value: The satisfaction gained through playing the game. The extent to which the player enjoys playing the game and the reason behind the enjoyment obtained of the game.
6. Implementation

The following implementation choices were needed:

**The ML Technique**
A suitable ML AI technique to achieve the study objectives was needed. Requirements for the AI technique include the ability to learn by observation of expert play and achieve human-like behaviour.

**The learning Algorithm**
A learning algorithm that could be used by the ML technique to learn the human players play style.

**The game**
Creating a game from ground-up for the study would be infeasible given the limited timeframe, therefore an existing game was needed that would provide a means of integrating the proposed AI technique.

**Testing procedure**
The tests should determine of how successfully the implemented AI technique meets the objectives of the study. The tests should involve a means of measuring an increase in the games replay and entertainment value.

6.1 The AI technique

There are several ML algorithms that can be applied to the problem, however after a careful consideration of all the algorithms and a review of the literature, ANN and decision trees were short listed. After a comparative analysis Giesler (2002) concluded that ANN’s are better suited for classification tasks which involve offline learning as compared to naïve bayes and decision trees. The ANN’s also had the following advantages;

- The ANN had better ability to generalize for the problem model of the study;
- ANN’s were better suited to complex problems;

Hence an ANN was better suited for the study.
The ANN used in this study is a fully connected multi-layer feed-forward ANN also known as the Multi-layer Perceptron (MLP). An MLP receives input through its input layers, processes the input in the hidden layer and returns the output through the output layer, given the nature of the Feedforward ANN, the information moves only in forward direction. The hidden layer of the ANN contains computational neurons that aid it in processing the input data (Stergiou & Siganos, 1997).

### 6.2 The learning algorithm

Back-propagation algorithm is a supervised learning technique commonly used for training a feed-forward ANN. It examines a given input pattern, generates the output and compares it to the actual output. If there is a difference between the generated output and the desired output, it calculates the error value which is used to adjust connection weights of the ANN. The term back-propagation is derived owing to the fact that the error value is sent backwards i.e. from the output layer to the input layer (Sweetser, 2003, p.619). Negnevitsky (2002, p.177) illustrates the back-propagation algorithm as follows. In this description we use these notations:

- \( F_i \) is the total number of inputs of neuron \( i \) in the network;
- \( y_i \) is the activity level of the \( j^{th} \) unit in a layer and \( W_{ij} \) is the weight of the connection between the \( i^{th} \) in a layer and the \( j^{th} \) unit in the next layer;
- \( \theta_i \) is the bias on unit \( i \);
- \( \text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \);
- \( \alpha \) is the learning rate.

**Step 1:** initialise the network by setting all the weights and threshold levels to random numbers, one neuron at a time.

\[
\begin{pmatrix}
-2.4 & 2.4 \\
F_i & \theta_i \\
\end{pmatrix}
\]

**Step 2:** Activation

The network is activated by applying the inputs and the desired output
a) Calculate the actual output of the neurons in the hidden layer, using the sigmoid activation function

\[ y_j(p) = \text{sigmoid} \left[ \sum_{i=1}^{n} x_i(p) \cdot w_{ij}(p) - \theta_j \right], \]

\( n \) is the number of inputs of neuron \( j \) in the hidden layer.

b) similar to the step above calculate the actual output of the neurons in the output layer

\[ y_k(p) = \text{sigmoid} \left[ \sum_{j=1}^{m} x_{jk}(p) \cdot w_{jk}(p) - \theta_k \right], \]

\( m \) represents the number of inputs of neuron \( k \) in the output layer.

**Step 3: Training**

Update the weights propagating backwards, the error value associated with the output neurons

a) calculate the error gradient for the neurons in the output layer

\[ \delta_k(p) = y_k(p) \cdot [1 - y_k(p)] \cdot e_k(p) \]

where

\[ e_k(p) = y_{d,k}(p) - y_k(p) \]

calculate the weight corrections:

\[ \Delta w_{jk}(p) = \alpha \cdot y_j(p) \cdot \delta_k(p) \]

Update the weights at the output neuron

\[ w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \]

b) calculate the error gradient for the neurons in the hidden layer

\[ \delta_j(p) = y_j(p) \cdot [1 - y_j(p)] \cdot \sum_{k=1}^{m} \delta_k(p) \cdot w_{jk}(p) \]

Calculate the weight corrections
\[ \Delta w_{ij}(p) = \alpha \cdot x_j(p) \cdot \delta_i(p) \]

Update the weights at the hidden neurons:

\[ w_{jk}(p + 1) = w_{jk}(p) + \Delta w_{jk}(p) \]

Step 4: iteration

Increase \( p \) by one and go back to step 2 and repeat the process until the selected error criterion is satisfied, which in this study was set to 0.005 (Negnevitsky, 2002, p.177).
6.3 The game

The game used in this study was a commercial FPS, Unreal Tournament 2004 (UT2004) published by Epic games Inc. UT2004 is the second sequel to the original and highly popular Unreal Tournament released in 1999. Like its predecessors UT2004 has a large global fan base and has spawned online communities of gamers contributing to the game via custom made mods. Epic has also released the complete source code for the game along with Unreal Development Environment (UDE), an integrated development environment for Unreal Script (Unreal Tournament.com, 2004).

The advantages and disadvantages of using UT2004 were as follows

Advantages

- UnrealScript tutorials provided by the game developers.
- Forums and websites with fan made mods available for download, along with several sites dedicated to creating mods in Unreal games.
- Access to the full UnrealScript source code for the game.
- Easy to modify bot behaviours and access to bots cognitive model (Dawes & Hall, 2006).
- The possibility of connecting an external program to the game and establishing communication though UnrealScript’s message passing mechanism.
- Projects such as Gamebots (Adobbati et al., 2001) and Dawes & Hall’s experiment (2006) prove the possibility of modifying UT2004 for the purpose of the study.

Disadvantages

- UnrealScript has a steep learning curve as compared to programming languages such as Java or C++, mainly due to the lack of a well documented API.
- Game installation consumes a large amount of space on the hard drive, approximately five and a half gigabytes.

The advantages outweigh the disadvantages and therefore UT2004 was used for this study.
6.3.1 Some key concepts of Unreal Tournament 2004

Unreal Engine
Unreal engine is a commercial game engine built by Epic games Inc. used to create UT2004. The Unreal engine, divides game execution time into events and ticks. The events are handled by the engine’s scripting language UnrealScript and the ticks are handled natively i.e. by the engine source code written in C++ (Sweeney, 1997). In the Unreal architecture all objects are known as actors and all in-game mobile characters are known as pawns. A pawn is assigned to a controller which, as its name implies, initiates the pawns various actions. The controllers can either be AI directed, in case of a computer controlled bot or player directed in case of a human player controlled character.

Tick
The UT2004 game engine manages time by dividing the game time into ticks which is the smallest unit of time typically between 1/10th and 1/100th of a second. The game engine updates the actors (characters, objects etc) in a level at every tick (Sweeney, 1997).

Event
An event occurs in UnrealScript when an actor sends a message and the actors that are set to listen to it respond. For example, if a player’s character is about to fall from a ledge, falling down is an event which calls the MyFall event of the player pawn class (Sweeney, 1997).

6.4 Testing procedure
Given the research questions, it was determined that the effectiveness of the implemented AI technique would be best judged by an audience of gamers. Therefore a population of twenty three gamers was subjected to a gaming session against several bots. The order of the games played was randomised to minimise order effect. The participants were twenty two males and one female of different nationalities between the age of eighteen and fifty. They were allowed to participate irrespective of their gender or race and they all had prior experience at playing a FPS game.

The participation procedure was as follows
- To familiarize with the UT2004 controls and rules of a Deathmatch game mode, each participant was subjected to an in-game tutorial.
• Once familiar with the UT2004 controls, the participant would play three games against bots controlled by static scripts and bots controlled by the implemented AI technique. Each game was of type Deathmatch, with the duration of fifteen minutes and a maximum score of twenty five.
• After each game the participants were asked to answer a questionnaire that would enable them to reflect their opinions on the bot they played against.
• Once they finished playing all three games, they were given a final questionnaire to get their opinion on the overall experience.

The game could be shortened by one of the players reaching the winning score before the pre-determined game time runs out. To prevent bias, the participants were unaware of the bots against which they were playing.

6.4.1 Map selection

To get the players perception about the implemented AI bots, it was important to have a balance between exploration and combat on the chosen map. Combat refers to the player having sufficient combat encounters with the bot, while exploration refers to the player being able to explore the map for useful items. There exist several maps in UT2004, however the chosen map titled “CRASH” was large enough to provide exploration and small enough to have frequent bot encounters.
7. The implemented system

The implemented AI model controls the state change mechanism of the bot in combat situations. Below is description of the components of the implemented system.

To use the game for the study, custom game types called mods were created and integrated into the game. Integrating an ANN internally in the game required thorough knowledge of UnrealScript and it was not possible to accumulate that knowledge given the time frame for the study. Therefore an external ANN independent of the games scripting language was created that provided the necessary flexibility in programming the ANN. The ANN connects to the game through the mods designed for the game. The mods start the game on a local port establishing a server, to which the ANN can connect to by means of a network socket. Once connected, it establishes a client-server relationship between the ANN and the game server, after which they start exchanging messages. See Figure 5

**Overview of the implemented System**

![Diagram](attachment:image.png)

The game environment after the game has been started. The spawned bot sends sensory information to the ANN module.

The AI model can be either the recorder, FeedforwardBot or the RecurrentBot.

This study follows the Adobbati et al. (2001) approach, where communication between the game and AI model occurs via network sockets.

**Figure 5: Overview of the implemented system**

There were two types of external ANNs created, *FeedforwardBot* and *RecurrentBot* that control the state change mechanism of a bot in the game.
7.1 FeedforwardBot

The FeedforwardBot ANN as its name implies consists of a feedforward MLP with ten hidden neurons in its hidden layer. Its output selection mechanism consists of returning the output with the maximum activation value associated with it.

The structure of the FeedforwardBot ANN is as shown in Figure 6 below:

![FeedforwardBot ANN structure](image)

- The coloured lines represent the links between the neurons in the network.
- The square boxes and circles represent the neurons in the network.
• An input parameter with a range of non-numeric values such as player and enemy weapon, has each value in its range represented by a neuron. A dark black neuron represents an activated neuron.

The inputs to the ANN are:
• Enemy distance: the distance between the bot and its opponent.
  Range= 0.1 (very close) to 3.73 (very far).
• Bot Health: The health level of the bot.
  Range= from 0 to 199
• Bot Shield: The shield power of the bot.
  Range= from 0 to 50.
• Bot weapon: The equipped weapon.
  Range=Shield gun, Assault rifle, Bio-Rifle, Minigun, Shock Rifle, Link Gun, Flak cannon, Rocket launcher and the Lightning gun.
• Enemy weapon: The equipped weapon of the opponent.
  Range=Shield gun, Assault rifle, Bio-Rifle, Minigun, Shock Rifle, Link Gun, Flak cannon, Rocket launcher and the Lightning gun.
• Ammo: The ammo left in the equipped b weapon.
  Range= Relative to the weapon.
• Enemy Firing: A value indicating whether the opponent is firing his equipped weapon.
  Range= 1 (not firing) to 2 (Firing).

The output value is called the next action, which can be one of the following actions:
• Hunting: in this action the bot follows its opponent once it is out of sight.
• Ranged Attack: When the bot fire’s the equipped weapon at its opponent from a long distance.
• Charging: Charging results in the bot charging at its opponent while firing the equipped weapon.
• Shield Self: The bot obtains a defensive stance and fire its equipped weapon at the opponent.
• Tactical Shoot: Tactical Shoot sees the bot moving in random directions while shooting at the opponent in order to dodge hostile fire and confuse the opponent.
7.2 RecurrentBot

The RecurrentBot uses a feedforward ANN of type MLP, with ten hidden neurons. One of primary objectives of the recurrent bot was to create an AI control mechanism with more randomness and unpredictability as compared to the FeedforwardBot. Therefore to achieve randomness, the RecurrentBot ANN uses a randomised action selection technique.

The randomised action selection technique used in this study selects an output from a group of outputs based on its fitness value. The probability of selection is the fitness value of the output divided by the total fitness of the population. The fitness value in this case is the activation value associated with the output.

The use of a randomised action selection technique, while effective at causing random behaviour may result in a bot that is too random and hence ineffective. Therefore to solve the problem of over randomness, an extra input parameter was added to the RecurrentBot ANN which kept track of the previous action executed. With knowledge of the previous action, the network has a form of memory through which the previous action had a higher probability of being selected.
The structure of the RecurrentBot ANN is as shown in Figure 7:

![Diagram of RecurrentBot ANN]

**Figure 7: The structure of the RecurrentBot ANN**

Legend
- The coloured lines represent the links between the neurons in the network
- The square boxes and circles represent the neurons in the network
- An input parameter with a range of non-numeric values such as player and enemy weapon, have each value in its range represented by a neuron. A dark black neuron represents an activated neuron.

The RecurrentBot ANN has the same inputs and outputs as the FeedforwardBot ANN, with an additional input parameter that is described below:
Previous Action: The last action performed by the bot.
Range: Hunting, Charging, Ranged Attack, Shield Self and Tactical shoot.

7.3 UT2004 modifications for the study

There were two categories of mods designed for UT2004, training mods and gameplay mods.

Training mods
These mods were designed to collect training data, through observation of expert play.

• **Feedforward Training:** This mod is used to collect training data for the FeedforwardBot. Once the mod is started the Recorder is connected to the game to start collecting training data.

• **Recurrent Training:** This mod is used to collect training data for the RecurrentBot. The recorder is connected to the game to collect training data after the mod is started.

Gameplay mods
These mods were designed for the ANN bots to interact with the in-game bot

• **Feedforward Player:** As soon as the FeedforwardBot establishes connection to the game, a bot is spawned in the game.

• **Recurrent Player:** The custom game mod that spawns a bot controlled by the RecurrentBot. It is the same as the Feedforward player, but it is given a unique name to avoid confusion.
7.4 The Recorder

The recorder was designed to gather training data for the FeedforwardBot and RecurrentBot, through observation of expert play.

7.4.1 The state changes of the data recorder

![Diagram of state changes during recording]

The Recorder system exists in two states Recording and Waiting to record. In the Waiting to record state the player moves around the map collecting useful items and looking for the opponent. The state changes to recording once the opponent comes in sight. As soon as the state change occurs, the Recorder begins storing input information and calculates the next action and continues storing data until the next state change occurs. The system reverts back to the waiting to record state, if the opponent is out of sight or the opponent is killed by the player.
7.4.2 Determining the player action

Since there was no definite way of determining the player action, assumptions about the player's action were made. Training samples are recorded every tick, that the opponent is in sight of the player in a twenty minute Deathmatch where the expert faces a static scripts AI bot.

Legend

- Distance: distance refers to the distance between the player and the opponent.
- Moving: the direction in which the player is moving from the opponent.
- Enemy in sight: if the player's opponent is in sight.
- Action: the player action determined by the system.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Moving</th>
<th>Enemy In sight</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far</td>
<td>Toward</td>
<td>Yes</td>
<td>Ranged Attack</td>
</tr>
<tr>
<td>Far</td>
<td>Backward</td>
<td>Yes</td>
<td>Shield Self</td>
</tr>
<tr>
<td>Far</td>
<td>Sideways</td>
<td>Yes</td>
<td>Tactical move</td>
</tr>
<tr>
<td>Medium</td>
<td>Toward</td>
<td>Yes</td>
<td>Charging</td>
</tr>
<tr>
<td>Medium</td>
<td>Backward</td>
<td>Yes</td>
<td>Tactical move</td>
</tr>
<tr>
<td>Medium</td>
<td>Sideways</td>
<td>Yes</td>
<td>Tactical move</td>
</tr>
<tr>
<td>Close</td>
<td>Toward</td>
<td>Yes</td>
<td>Charging</td>
</tr>
<tr>
<td>Close</td>
<td>Backward</td>
<td>Yes</td>
<td>Tactical move</td>
</tr>
<tr>
<td>Close</td>
<td>Sideways</td>
<td>Yes</td>
<td>Tactical move</td>
</tr>
<tr>
<td>Far/Medium or Close</td>
<td>Toward</td>
<td>No</td>
<td>Hunting</td>
</tr>
</tbody>
</table>

Table 1: Table of assumptions
7.4.3 Gathering Training data

An illustration of how the data flows through the game and the Recorder, while training data is collected.

![Diagram of training data collection process]

**Figure 9: Training data collection process**

Note: the numbers in the brackets in Figure 9 denote the sequence in which the steps are carried out.

Step by step description of Figure 9: Training data collection process.

- Step 1: Launch UT2004;
- Step 2: Load the custom gametype;
- Step 3: The Recorder connects to the game at post game start up;
- Step 4: The player starts playing the game;
Steps 5 and 6: while the game is in progress the player action is determined and training examples are sent to the recorder;

Steps 7: The recorder stores the training example in a temporary data store;

Steps 8, 9 and 10: As soon as the game ends, the recorder stores the training examples from the temporary data store to a permanent data store as a dataset file;

Steps 5 to 7 are repeated as long as the player is playing the game. The training dataset is the final set of examples permanently stored from the temporary data store. For the RecurrentBot each training example has the value of the previous action, which was the primary difference between the datasets used to train both RecurrentBot and FeedforwardBot. Given the fact that this study only focuses on the combat aspects of the gameplay, the game sends the data only when the enemy is in sight of the human player. The stored dataset is later used to train one of the bot ANN.

7.5 Training the bots

To train the ANN bots their respective datasets are loaded and training is initiated. Once the training finishes the trained weights are saved and loaded into the ANN bots for gameplay. Training of the FeedforwardBot bot resulted in 72% and the recurrent bot training resulted in 94% accuracy (using 10-fold cross-validation).

Equalizing the dataset

An early version of the trained bots resulted in a bot that learned only one action and therefore the bot kept doing the same action. After pondering over the problem it was discovered that the training data collected was through expert player play on a single map which resulted in an over trained bot. This was in contrast to the study objective as the trained bot lacked the ability to generalize and hence kept repeating the same action. Therefore the algorithm for equalising the dataset was implemented.

The training examples collected through expert play, were inconsistent due to the expert performing a particular action more frequently than the others. Hence the algorithm for equalising the training dataset was written. The algorithm was as follows:

- Get the number of training examples for each action
• get the action that has the highest number of training examples \( maxExamples \)
• Compute a threshold value \( \mu \)
  \[
  \mu = maxExamples \times 0.75
  \]
• for each action that has less corresponding training examples than \( \mu \)
  
  o sort the training examples for the action in descending order
  
  o Starting from the first training example
  
  o begin replicating training examples sequentially
• until the number of training examples is equal to \( \mu \)
• save the new modified dataset

### 7.6 The ANN state change mechanism

The following sections describe the interaction between the ANN bots and the game as they control the state change mechanism of the in-game bot.
7.6.1 The ANN bot

Waiting is the default state for the bot, during which the bot moves around the map collecting various items and weapons. Once the bot spots the opponent i.e. the human player, it enters the combat mode where it changes its state as instructed by the ANN.

**Figure 10: State changes of the bot during gameplay**

Waiting is the default state for the bot, during which the bot moves around the map collecting various items and weapons. Once the bot spots the opponent i.e. the human player, it enters the combat mode where it changes its state as instructed by the ANN.

7.6.2 The game and ANN communication process

Given the similarity in the ANN architecture of the two bots a common term bot ANN will be used to refer to both the RecurrentBot and FeedforwardBot in the diagram below. The diagram below illustrates the data flow process through the bot ANN and the game during gameplay. This diagram also shows how the bots interact with the game using the request response paradigm.
Figure 11: data flow between ANN and bot during gameplay

Note: the numbers in the brackets in Figure 11 denote the sequence in which the steps are carried out.

Step by step description of Figure 11: data flow between ANN and bot during gameplay.

- Step 1: Launch UT2004.

[Diagram description]

Figure 11: data flow between ANN and bot during gameplay

Note: the numbers in the brackets in Figure 11 denote the sequence in which the steps are carried out.

Step by step description of Figure 11: data flow between ANN and bot during gameplay.

- Step 1: Launch UT2004.
• Step 2: Load the relevant game type i.e. Feedforward Player or Recurrent Player.
• Step 3: the bot ANN connects to the game. It could be either one of the two bots i.e. RecurrentBot or FeedforwardBot.
• Step 4: The incoming connection spawns a character in the game
• Step 5: As soon as the bot sees an opponent in the game, it sends the information to the bot ANN.
• Step 6: The bot ANN processes the input.
• Step 7: The bot ANN stores the incoming bot information and game information in a temporary data store.
• Step 8: The bot ANN sends the output to the in-game bot.
• Step 9: Once the game ends, it sends a notification of game ended to the ANN module.
• Step 10 and 11: The bot ANN retrieves the game and bot information and saves them to in their respective files.

As mentioned earlier, the bot ANN controls the state change mechanism of the in-game bot during combat. In order to do so, the bot and ANN communicate though a request-response paradigm, whereby the bot sends the ANN a request in the form of a message composed with the input pattern requesting for the new state to change to. The ANN replies to the request by processing the input pattern and sending the new state as a message. The received message triggers an event in UnrealScript that causes the bot to change its state to a new state. The bot state remains unchanged if the new state sent by the ANN is the same as its existing state.
8. Implementation phases

8.1 Preparation Phase

The preparation phase involved determining the network architecture i.e. the input and output parameters of the ANN. This was the most crucial aspect of the ANN design, owing to the fact that the game takes place in an open environment where the choice of the input and output parameters had a significant impact on the ANN’s performance. The outcome of this phase was the ANN design with the input and output parameters.

8.2 First Implementation Phase

The Recorder was developed during the first implementation phase along with a mod for UT2004. The logic for determining the player action was determined and implemented in the Feedforward learner and Recurrent learner mods. The outcome of this phase was the recorder ready to collect training data from the game through the training mods.

8.3 Second Implementation Phase

This phase was divided into four stages as follows, where in the first stage

- Stage 1: The FeedforwardBot was developed.
- Stage 2: Training data was collected.
- Stage 3: The FeedforwardBot was trained.
- Stage 4: Develop the Feedforward player mod

The outcome of this phase was a fully trained FeedforwardBot ready to play its first game against a human player.

8.4 Third Implementation Phase

Similar to the second implementation phase this phase can be divided into four stages

- Stage 1: The RecurrentBot was developed.
- Stage 2: Training data was collected.
- Stage 3: The RecurrentBot was trained.
- Stage 4: Develop the Recurrent player mod.
The outcome of this phase was a RecurrentBot ready for its first game against a human player.
9. The questionnaires

There were two types of questionnaires given to the participants, bot specific questionnaires and a bot rating questionnaire. Given the objective of the research it was important to get the participants attitude on the behavior of the bot. In order to accommodate that, closed Likert scale questions, multiple-choice questions and some open questions were used (Waddington, 2000). The Likert scale questions had the following options

- Strongly disagree : SD
- Disagree: D
- Neutral: N
- Agree: A
- Strongly Agree: SA

The values of the multiple-choice questions were relative to the question. The following contains a description of the bot specific questionnaire

9.2 Bot specific questionnaire

A bot specific questionnaire was given to the participant after each gameplay session. The purpose of this questionnaire was to obtain feedback on their experience of playing against the bot. Data on the behavioural aspects of the bot such as unpredictability and human-like attributes was collected using Likert scale questions. To obtain feedback on the entertainment value and combat skills of the bot Likert scale questions were used. The participants were given a bot specific questionnaire after each gaming session. The questions were arranged in their categories to minimise confusion for the participant.

9.3 Bot rating Questionnaire

This was the final questionnaire answered by participants once they finished playing against all the three bots. This section was composed of multiple-choice questions, whereby the participants overall views and opinions on the bot of their choice was determined. This questionnaire was designed to get the overall feeling from the participant of the bot that affected the game’s entertainment and replay value the most. This questionnaire can be referred to as having a voting scheme as the participants voted for the bot of their choice for each question.
The votes were totalled and represented on a column chart.

9.4 Data analysis techniques

For the purpose of a comparative analysis the participant responses to each question for the three bots were compared against each other. The were three statistical data analysis techniques applied, were

- Paired t-test: used for the bot specific questionnaire to compare the participant responses for each question for the three bots.
- ZTEST: used for the bot specific questionnaire to determine the rejection or failure of rejection of the null hypothesis at 5% significance level for each question for each bot (The Mathworks, Inc. 2007).
- CHITEST: the CHITEST is used in the bot rating questionnaire to validate the results displayed by each plot.

Microsoft Excel 2003 was used in this study for data analysis and hence the process of performing the paired t-test was automated.
10. Data analysis: bot specific questionnaire

Each category of questions is represented in this section by a column chart, followed by a table for paired t-tests, a table for ZTEST values and finally deductions on the category.

For the purpose of data analysis, the average response value per question was calculated for the three bots and plotted on a column chart i.e. represented using a column chart. Each plot was based on a group of questions categorised according to the different aspects of the bots.

The x-axis of the chart represents the questions, the y-axis represents the range and the columns represent the bots. The range with respect to the bot specific questionnaire was from -2 to +2, with an interval of 1. Each question in the bot specific questionnaire had five options represented by a unique value in the range as follows:

- Strongly disagree : -2
- Disagree: -1
- Neutral: 0
- Agree: 1
- Strongly Agree: 2

The above options are common to all the questions in the bot specific questionnaire unless stated otherwise.

The paired t-test values for each question are calculated and presented on the table, where the columns represent the question and the row represents the bots, with the paired t-test value in the cell.

In the ZTEST values table, the rows represent the question and the columns represent the bots. The values in the cells indicate the calculated ZTEST values. For each ZTEST table, the values in bold indicate failure to reject the null hypothesis. The values for the Static scripts bot are subtracted by 1, as the alternative hypothesis (that the subject disagrees) for Static scripts bot is the opposite of the alternative hypothesis for the FeedforwardBot and RecurrentBot (that the subject agrees).

Example:
For the question, “The bots movement was often unpredictable?” the alternative hypothesis for RecurrentBot and FeedforwardBot would be the bot movement was unpredictable. As per the Static scripts bot, the alternative hypothesis would be the bot movement was no unpredictable. The ZTEST values are Table 3: ZTest values for unpredictibility in bot behaviour.

- RecurrentBot = 0.0008
- FeedforwardBot = 0.0000284
- Static scripts bot = 0.081

Therefore the null hypothesis is rejected for the recurrent and FeedforwardBot as both 0.0008 and 0.0000284 are less than 0.05; however it is accepted for the Static scripts bot. Hence the ZTEST values for the bots would be in bold i.e.

| 0.0008 | 0.0000284 | 0.081 |

### 10.1 Unpredictability in bot behavior

To get determine the unpredictability of the bot behaviour, the following questions were asked:

- Question 1: The bot managed to surprise you with its unpredictable combat strategies?
- Question 2: Due to the bot’s unpredictable combat behaviour, you had to re-think your combat strategies from time to time.
- Question 3: The bots movement was often unpredictable?
To refer to the plot for this section see Figure 12 and for the paired t-test value see Table 2. From the data presented above it can be determined that

- From the plot and the paired t-test values for Question 1 both the RecurrentBot and FeedforwardBot successfully managed to surprise the player with their unpredictable combat strategy, when compared to a bot controlled by Static scripts.
• Both the plot and paired t-test values for Question 2 indicate that the FeedforwardBot managed to make the player re-think his/her combat strategy from time to time more effectively than the Static scripts bot.
• The movement of both FeedforwardBot and RecurrentBot was unpredictable when compared to the Static scripts bot.

10.2 Human-like attributes of the bot

To determine the human-like attributes of the bot, the following questions were asked

• Question 1: The bot's combat skills made it appear more human-like?
• Question 2: The bot displays human-like dodging skills?
• Question 3: The bot Displays human like movement?
• Question 4: The bot displayed human-like behavior?
• Question 5: The bot appeared as if a human player was controlling it?

The five questions are referred to as Q1, Q2, Q3, Q4 and Q5 in the plot below.

![Human-like attributes of the bot](image)

**Figure 13: Human-like attributes plot**
Bots

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecurrentBot and FeedforwardBot</td>
<td>0.096</td>
<td>1</td>
<td>0.754</td>
<td>0.462</td>
<td>0.396</td>
</tr>
<tr>
<td>RecurrentBot and Static scripts bot</td>
<td>0.320</td>
<td>0.004</td>
<td>0.034</td>
<td>0.023</td>
<td>0.258</td>
</tr>
<tr>
<td>FeedforwardBot and Static scripts bot</td>
<td>0.824</td>
<td>0.009</td>
<td>0.048</td>
<td>0.116</td>
<td>0.732</td>
</tr>
</tbody>
</table>

Table 4: paired t-test values for Human-like attributes

<table>
<thead>
<tr>
<th>FeedforwardBot</th>
<th>RecurrentBot</th>
<th>Static scripts bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 0.330</td>
<td>0.0008</td>
<td>0.7337</td>
</tr>
<tr>
<td>Q2 0.001</td>
<td>0.002</td>
<td>0.046</td>
</tr>
<tr>
<td>Q3 0.008</td>
<td>0.018</td>
<td>0.187</td>
</tr>
<tr>
<td>Q4 0.029</td>
<td>0.0007</td>
<td>0.419</td>
</tr>
<tr>
<td>Q5 0.058</td>
<td>0.002</td>
<td>0.823</td>
</tr>
</tbody>
</table>

Table 5: Z-test values for Human-like attributes

To refer to the plot for this section see Figure 13 and for the paired t-test value see Table 4. From the data presented above it can be determined that

- From the plot of Q1 it appears that the RecurrentBot has the most human-like combat skills of all the bots. However the paired t-test values for Q1 indicate that all the bots possess combat skills that make them appear equally human-like.
- Both the plot for Q2 and Q3 and the paired t-test values for Q2 and Q3 indicate that the FeedforwardBot and the RecurrentBot display human-like movement and dodging skills when compared to Static scripts bot.
- The RecurrentBot displays more human-like behaviour when compared to Static scripts bot. This is evident from the paired t-test value for Q4.
- Despite minor differences in the plot for Q5 the paired t-test values for Q5 indicate that all the bots appear equally as if the human was controlling it.

10.3 Bot’s skills and intelligence

To determine the skills and intelligence of the bot the following questions were asked

- Question 1: The bot was effective at all sorts of combat (close, ranged, medium range combat)?
- Question 2: The bots combat skills make it an interesting opponent?
- Question 3: The bot demonstrated intelligent behaviour?
Figure 14: Bot skills and intelligence plot

<table>
<thead>
<tr>
<th>Bots</th>
<th>Question 1</th>
<th>Question 2</th>
<th>Question 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecurrentBot and FeedforwardBot</td>
<td>0.1865</td>
<td>0.347</td>
<td>0.445</td>
</tr>
<tr>
<td>RecurrentBot and Static scripts bot</td>
<td>0.211</td>
<td><strong>0.006</strong></td>
<td><strong>4.15E-06</strong></td>
</tr>
<tr>
<td>FeedforwardBot and Static scripts bot</td>
<td><strong>0.003</strong></td>
<td><strong>0.005</strong></td>
<td><strong>0.007</strong></td>
</tr>
</tbody>
</table>

Table 6: paired t-test values for bot skills and intelligence

<table>
<thead>
<tr>
<th>Question</th>
<th>FeedforwardBot</th>
<th>RecurrentBot</th>
<th>Static scripts bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td><strong>2.05E-05</strong></td>
<td>0.081</td>
<td>0.309</td>
</tr>
<tr>
<td>Question 2</td>
<td><strong>3.98E-09</strong></td>
<td><strong>9.35E-05</strong></td>
<td>0.360</td>
</tr>
<tr>
<td>Question 3</td>
<td><strong>1.13E-09</strong></td>
<td><strong>1.60E-05</strong></td>
<td>0.360</td>
</tr>
</tbody>
</table>

Table 7: ZTEST and average values for bot skills and intelligence

To refer to the plot for the questions see Figure 14 and for the paired t-test values see Table 6.

- The plot for Question 1 and the paired t-test values for Question 1 in the
  able indicate that FeedforwardBot is more effective in all sorts of combat
  when compared to the Static scripts bot.
- Both the plot for Question 2 and the paired t-test values prove that the
  combat skills of the RecurrentBot and FeedforwardBot make them more
  interesting opponents than the Static scripts bot.
• Both the plot and the paired t-test values for Question 2 prove that the RecurrentBot and FeedforwardBot demonstrate more intelligent behavior when compared to Static scripts bot.

10.4 Bot difficulty level

To get feedback on the participant’s feelings about the bot’s combat and difficulty level the following multiple-choice questions were asked.

To determine the challenge level for each bot according to the participant the following question was asked

Question 1: listed on the x-axis of the plot

Options: the following options below represent the y-axis and are given the following unique numerical values for representing it on the chart

- Very hard: -2
- Hard: -1
- Average: 0
- Easy: 1
- Very easy: 2

![Graph showing bot difficulty levels](image)

**Figure 15: Bot difficulty**
Bots | Question 1
--- | ---
RecurrentBot and FeedforwardBot | 1
RecurrentBot and Static scripts bot | 3.98E-05
FeedforwardBot and Static scripts bot | 0.001

Table 8: paired t-test values for bot difficulty

<table>
<thead>
<tr>
<th></th>
<th>FeedforwardBot</th>
<th>RecurrentBot</th>
<th>Static scripts bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>0.052</td>
<td>0.111</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Table 9: ZTEST value for bot difficulty

For the plot see Figure 15 and for the paired t-test values see Table 8.

From the plot and the paired t-test values it is evident that the Static scripts bot is the easiest of the three bots when compared to RecurrentBot and FeedforwardBot. The difficulty level of both FeedforwardBot and RecurrentBot can be determined as being in between average to hard.

10.5 Bot combat skills

This question was asked to determine the participants’ overall opinion on the bot’s combat skills.

Question: listed on the x-axis of the chart

Options: the following options below represent the y-axis and are given the following unique numerical values for representing it on the chart

- Very strong = -2
- Strong = -1
- Average = 0
- Weak = 1
- Very weak = 2
The bot's combat skills were?

<table>
<thead>
<tr>
<th>Bots</th>
<th>Question 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecurrentBot and FeedforwardBot</td>
<td>0.365</td>
</tr>
<tr>
<td>RecurrentBot and Static scripts bot</td>
<td><strong>0.003</strong></td>
</tr>
<tr>
<td>FeedforwardBot and Static scripts bot</td>
<td><strong>0.0003</strong></td>
</tr>
</tbody>
</table>

Table 10: Paired t-test values for bot combat skills

<table>
<thead>
<tr>
<th>Question</th>
<th>FeedforwardBot</th>
<th>RecurrentBot</th>
<th>Static scripts bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>0.00001</strong></td>
<td>0.021</td>
<td>0.101</td>
</tr>
</tbody>
</table>

Table 11: ZTEST value for bot combat skills

From the plot and the paired t-test values it is evident that the Static scripts bot has the weakest combat skills of all the bots falling into the range of being average to very easy. The RecurrentBot and the FeedforwardBot fall into the category of being average to strong.

### 10.6 Entertainment value of the bot

To determine the entertainment value of the game owing to the bot behaviour, the following questions were asked.

- Question 1: You enjoyed the game owing to the bot's combat skills?
- Question 2: The bot was fun to play against?
Figure 17: Entertainment value of the bot

<table>
<thead>
<tr>
<th>Bots</th>
<th>Question 1</th>
<th>Question 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecurrentBot and FeedforwardBot</td>
<td>0.295</td>
<td>0.1347</td>
</tr>
<tr>
<td>RecurrentBot and Static scripts bot</td>
<td>0.058</td>
<td><strong>0.004</strong></td>
</tr>
<tr>
<td>FeedforwardBot and Static scripts bot</td>
<td>0.423</td>
<td>0.148</td>
</tr>
</tbody>
</table>

Table 12: Paired t-test values for bot entertainment value

<table>
<thead>
<tr>
<th></th>
<th>FeedforwardBot</th>
<th>RecurrentBot</th>
<th>Static scripts bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td><strong>5.57633E-10</strong></td>
<td><strong>8.88178E-16</strong></td>
<td>0.998</td>
</tr>
<tr>
<td>Question 2</td>
<td>0</td>
<td>0</td>
<td>0.99996</td>
</tr>
</tbody>
</table>

Table 13: ZTEST values for bot entertainment value

For the plot see Figure 17 and for the paired t-test values see Table 12.

- The plot for Question 1 shows minor differences in the entertainment value provided through the bot’s combat skills. The paired t-test values for Question 1 indicate that all the three bots provided the same level of entertainment owing to their combat skills.
- The plot for Question 2 shows the RecurrentBot and the FeedforwardBot being more fun to play against. The paired t-test values for Question 2 indicates, the RecurrentBot being more fun to play against when compared to Static scripts bot.
10.7 Overall experience playing against the bot

Question 1: listed on the x-axis of the plot.

Options: the following options below represent the y-axis and are given the following unique numerical values for representing it on the chart:

- Not Enjoyable: -2
- Less Enjoyable: -1
- Neutral: 0
- Enjoyable: 1
- Highly Enjoyable: 2

![Bar chart showing overall gaming experience](chart.png)

Figure 18: Overall experience of playing against the bot

<table>
<thead>
<tr>
<th>Bots</th>
<th>Question 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecurrentBot and FeedforwardBot</td>
<td>0.003</td>
</tr>
<tr>
<td>RecurrentBot and Static scripts bot</td>
<td>0.004</td>
</tr>
<tr>
<td>FeedforwardBot and Static scripts bot</td>
<td>0.365</td>
</tr>
</tbody>
</table>

Table 14: paired t-test values for overall gaming experience

<table>
<thead>
<tr>
<th>Question</th>
<th>FeedforwardBot</th>
<th>RecurrentBot</th>
<th>Static scripts bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1685E-10</td>
<td>0</td>
<td>0.999995</td>
<td></td>
</tr>
</tbody>
</table>

Table 15: ZTEST value for overall gaming experience
For the plot see Figure 18 and for the paired t-test values see Table 14.

It is evident from both the plot and the paired t-test values that the most RecurrentBot is more enjoyable than FeedforwardBot and Static scripts bot.
11. Data Analysis: bot rating questionnaire

There were five questions in this section and the number of votes for each bot was totalled and represented using a column chart. Each question in this questionnaire is represented using a plot of the column chart.

The x and y axis for the column chart represent the following:
- Y-axis: Number of participants
- X-axis: the question asked

The columns in the chart are the three bots, and the number at its peak represents the number of votes.

11.1 Question 1

![Most enjoyable bot](image)

**Figure 19: Enjoyability**

As seen on the plot, RecurrentBot has the highest number of participant votes for being the most enjoyable bot to play against. However, a CHITEST value of 0.296 for this plot indicates that the difference is not statistically significant (possibly due to the low number of subjects).
11.2 Question 2

![Least enjoyable bot](image)

**Figure 20: Least entertaining**

As indicated on the plot, the Static scripts bot received the maximum number of votes and therefore it's the least entertaining of all. However, again, the CHITEST value of 0.337 is not statistically significant.
11.3 Question 3

The bot with the most human-like behaviour

- FeedforwardBot
- RecurrentBot
- Static scripts bot

Which bot displayed the most human-like behaviour?

Figure 21: Most human-like behaviour

This plot shows a close call as to the number of participants who find the RecurrentBot to be the most human-like of all the bots. Once again the CHITEST value of 0.840 shows no significant difference.
11.4 Question 4

Figure 22: Replayability

This question investigates one of the key aspects of the study, replayability. As observed from the plot, the RecurrentBot has the maximum number of votes for being the most replayable bot. The CHITEST value for this question is 0.070, which is not quite significant.
11.5 Question 5

Do you feel that fifteen minutes time was sufficient to differentiate between the three bots and their behaviours?

Figure 23: Time efficiency

Time was an important factor for the study as had there been insufficient time for the participants to distinguish between the bots the experimental results may not have been very clear. However as evident from the plot and the CHITEST value of 0.0002 it's evident that 15 minutes was sufficient to distinguish between the bots.
12. Discussion

Through observation and the data presented above, it is clear that the three bots have different play styles. They have the same aiming accuracy and speed of movement i.e. they all have the skill level of the in-built UT2004 bot "Experienced", but what sets them apart is their state change mechanism. It is the state change mechanism that determines the bot's strategy in combat situations. Hence what contributes to the player enjoyment and replayability is the strategy adopted by the bot in these situations.

The CHITEST values obtained are a result of a smaller sample population size. Given the consistency of the results, it seems likely that a higher population size would result in a smaller CHITEST value in favour of the implemented AI controlled bots.

Static scripts bot was the weakest of the three bots (see Figure 15). Its combat skills ranged from weak to average (see Figure 16) and it displayed predictable behaviour when compared to the other bots. However this bot was fun to play against and provided substantial entertainment value to the gamer. Overall the Static scripts bot was inferior to the RecurrentBot and the FeedforwardBot.

The FeedforwardBot was a well trained bot effective at all sort of combat (see Figure 14). The FeedforwardBot featured an unpredictable combat strategy (see Figure 11) coupled with human-like dodging skills (see Figure 13) and it was effective at all types of combat (see Figure 14) when compared to the other two bots. This proved to be a challenge for advanced level participants and they enjoyed it thoroughly. However it caused some frustration amongst intermediate to novice level players, as they found it too challenging and difficult.

As seen in Figure 21 and Figure 13, it is evident that RecurrentBot is the most human-like bot of the three bots and therefore has a slightly higher entertainment and replay value. Despite having a similar training mechanism to the FeedforwardBot, the RecurrentBot mimics the expert play style more accurately than the FeedforwardBot. When compared to the other two bots, some of the traits of the RecurrentBot are that it

- combat skills are average to strong and hence more human-like (see Figure 12)
- displays slightly more intelligent behaviour see Figure 14
- has average to strong combat skills see Figure 16
- is average to hard to defeat it see Figure 15

The above traits of the bot make it more fun to play against (see Figure 17) and results in a gaming experience that is more enjoyable than the FeedforwardBot and the Static scripts bot (see Figure 18).

This gives an indication that players prefer a challenging AI opponent that is human-like and maintains a balance between challenge and entertainment. The players do not want to be intimidated by an unrealistically accurate opponent that kills them every time. Players also do not want an opponent who they can kill with ease, every time they spot them. If the AI opponent is too challenging (FeedforwardBot) or too easy (Static scripts bot) (see Figure 20), it will reduce the entertainment level for some players and may also lead to frustration. Therefore an AI character that exhibits a degree of randomness and unpredictability often surprises the player, which maintains the entertainment and replay value of the game at a high level.
13. Limitations

Smaller Sample Population
There are no demographic questions in the questionnaire, owing to the size of the population. A larger sample population would provide an opportunity for demographic questions that could lead to more informative results.

Support Character
This study only investigates the effect of the implemented AI model strictly from a computer game opponent perspective; however an investigation could be conducted on the AI model to control a support character that aids the player. This would help deduce the effectiveness of the implemented AI model as a support characters in the games.

Limited to one game mode
This study is solely focused on testing the implemented AI model in a Deathmatch game. Team games such as Capture-the-flag and domination require the competing teams to strategize their actions in order to gain victory. Hence testing the implemented AI model in a team environment would give an indication of its co-ordination and planning skills.

Limiting the initial scope
Successful implementation of online learning would have added an extra degree of unpredictability and human-like behaviour in the bot. However whether fifteen minutes of game time is sufficient for the bot to adapt is debatable (see 2.3 Initial Scope of the research).
14. Conclusion

The aim of the study was to achieve high level behaviour in a bot through the use of machine learning to increase the game’s entertainment and replay value. Hence the implemented AI model used ML to learn from expert human player play and it was used to control the state change mechanism of a bot in UT2004. To test the effectiveness of the implemented system was evaluated by a group of human players who played against the bots controlled by static scripts and the AI model individually. Feedback on the player’s opinions on the bots was collected through questionnaires. The questionnaires were then subjected to statistical analysis, which prove that the bots controlled by the implemented AI technique are more entertaining and replayable than the bots controlled by static scripts.

An extension of the experiment could be integrating multiple ANNs into the AI model where an ANN could be used to control different aspects of the bots behaviour e.g. movement and weapons. Further research in this area could be carried out at creating an AI controlled bot that adapts online. If implemented efficiently, the AI may learn from the human player play style and improve upon it after repeated attempts. This would result in a bot that is constantly changing its strategies to counter the human player play style.

In conclusion this study shows that the use of an ML technique such as an ANN can result in a more human-like AI opponent when compared to Static scripts controlled AI opponent.
15. References


Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E. (2002). Evolving Improved Opponent Intelligence. 3rd International Conference on Intelligent Games and Simulation (pp. 94-98). SCS Europe: Bvba


### 16. Glossary

<table>
<thead>
<tr>
<th>AI Techniques and General Terms</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Intelligence (AI)</td>
<td>The sciences of making machines perform tasks that require human intelligence, through the use of programs.</td>
<td>(Negnevitsky, 2002)</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>The process by which a machine improves its performance over time, based on its past experiences.</td>
<td>(Negnevitsky, 2002)</td>
</tr>
<tr>
<td>Supervised learning</td>
<td>Learning that takes place while the game is being played by a gamer.</td>
<td>(Spronck, 2005)</td>
</tr>
<tr>
<td>Online Learning</td>
<td>The process by which the AI adapts during gameplay.</td>
<td>(Ponsen, 2004)</td>
</tr>
<tr>
<td>Offline Learning</td>
<td>The process by which the AI adapts by self-play and without human intervention.</td>
<td>(Ponsen, 2004)</td>
</tr>
<tr>
<td>Reinforcement Learning (RL)</td>
<td>Reinforcement Learning is used to train an agent to exhibit specific behaviour by rewarding and penalising agent action coupled to states.</td>
<td>(Spronck, 2005)</td>
</tr>
</tbody>
</table>

### 16.1 Techniques

<table>
<thead>
<tr>
<th>AI Techniques</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Neural Networks (ANN)</td>
<td>An ANN is an electronic simulation based on a simplified human brain.</td>
<td>(Sweetser, 2002)</td>
</tr>
<tr>
<td>Finite State Machines (FSM)</td>
<td>FSM is an AI technique that divides game objects behaviour in to logical states, so that the object has a behaviour for each different type of behaviour it exhibits.</td>
<td>(Sweetser, 2002)</td>
</tr>
<tr>
<td>Genetic Algorithms (GA)</td>
<td>An AI technique for optimization and ML that uses ideas from evolution and natural selection to evolve a solution to a problem</td>
<td>(Sweetser, 2002)</td>
</tr>
</tbody>
</table>
Fuzzy State Machines (FuSM) are Finite State Machines (FSM), also known as Finite State Automation (FSA), at their simplest, are models of the behaviours of a system or a complex object, with a limited number of defined conditions or modes, where mode transitions change with circumstance.

### 16.2 Game Genres

<table>
<thead>
<tr>
<th>Game Genres</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First-Person-Shooter (FPS)</strong></td>
<td>FPS is a game played from the first person perspective. The player is embodied in the character that they control i.e. Half Life.</td>
<td>(Jagger, 2004)</td>
</tr>
<tr>
<td><strong>Role Playing Game (RPG)</strong></td>
<td>RPG’s let players choose from a variety of roles. The player goes on quests, collects and sells items, fights monsters, and expands the capabilities of their character (such as strength, magic, quickness, etc.), all in an extended virtual World.</td>
<td>(Laird &amp; Van lent, 2001)</td>
</tr>
<tr>
<td><strong>Massively Multiplayer Online (MMO)</strong></td>
<td>Massively Multi-player On-Line (MMO) games are played on the internet with many people i.e. the MMORPG Everquest.</td>
<td>(Jagger, 2004)</td>
</tr>
<tr>
<td><strong>Adventure</strong></td>
<td>Gameplay involves the player moving around a restricted locale, solving puzzles and interacting with characters in an attempt to further a story line.</td>
<td>(Fairclough, 2001)</td>
</tr>
<tr>
<td><strong>God Games</strong></td>
<td>God games give the player god-like control over a simulated world.</td>
<td>(Laird &amp; Van lent, 2001)</td>
</tr>
<tr>
<td><strong>Real time strategy (RTS)</strong></td>
<td>Military Simulations where the player controls armies, made up of different types of units, with the aim of defeating all opposing forces.</td>
<td>(Ponsen &amp; Spronck, 2004)</td>
</tr>
</tbody>
</table>
## 16.3 Gaming Terms

<table>
<thead>
<tr>
<th>Gaming terms</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamer</td>
<td>A human player who plays games.</td>
<td>(none)</td>
</tr>
<tr>
<td>Bot</td>
<td>Often used in FPS games, a bot is a computer controlled character in a game.</td>
<td>(none)</td>
</tr>
<tr>
<td>Gameplay</td>
<td>The manner in which a game plays</td>
<td>(none)</td>
</tr>
<tr>
<td>Platform</td>
<td>The term platform refers to the kind of machine needed to play the game.</td>
<td>(Jagger, 2004)</td>
</tr>
</tbody>
</table>
17. Appendix

17.1 Questionnaires

17.1.1 Bot specific Questionnaire

For the purpose of hiding the identity of the bots the participants were playing against, all the bots were given code names such as

- Master Chief for RecurrentBot
- Cortana for FeedforwardBot
- Arbiter for the Static scripts bot

Your thoughts on the bot you played against

Bot name: (Relevant to the code name given to the bot they played against)

Date:

Answer key

SA: Strongly Agree
A: Agree
D: Disagree
SD: Strongly Disagree
N: Neutral

The bot behavior

Questions

Circle one of the below

1. The bot was effective at all sorts of combat (close, ranged, medium range combat)
   SD  D  N  A  SA

2. The bot managed to surprise you with its unpredictable combat strategies
   SD  D  N  A  SA

3. Due to the bot’s unpredictable combat behavior, you had to re-think your combat strategies from time to time
   SD  D  N  A  SA

4. The bot’s combat skills made it appear more human-like
   SD  D  N  A  SA

5. The bots combat skills make it an interesting opponent
   SD  D  N  A  SA

6. You enjoyed the game owing to the bot’s combat skills
   SD  D  N  A  SA

Movement
Questions
1. The bots movement was often unpredictable
2. The bot displays human-like dodging skills
3. The bot Displays human like movement

Overall Performance
Questions
1. displayed human-like behaviour?
2. was fun to play against?
3. demonstrated intelligent behaviour?

Put a circle around the option of your choice
The challenge level at defeating the bot? (Choose one)
1. Very hard
2. Hard
3. Average
4. Easy
5. Very easy
The bots combat skills were (Choose one)
1. Very strong
2. Strong
3. Average
4. Weak
5. Very Weak

What was your gaming experience like? (Choose one)
1. Not Enjoyable
2. Less Enjoyable
3. Neutral
4. Enjoyable
5. Highly Enjoyable

Any comments on the bots combat skills?
17.1.2 Bot rating questionnaire

Your overall thoughts

After playing 3 games of UT2004, titled Master Chief, Cortana and Arbiter, answer the questions below. Answer the questions by putting circle around your choice i.e. follow the example shown on the right

For example:
Q.) you enjoyed playing against the most?
1)Master Chief 2)Cortana 3)Arbiter

Circle your choice

Q.1) Which bot did you enjoy playing against the most? (Choose one)
1) Master Chief 2) Arbiter 3) Cortana

Q.2) Which bot did you enjoy playing against the least? (Choose one)
1) Master Chief 2) Arbiter 3) Cortana

Q.3) Which bot displayed the most human-like behaviour? (Choose one)
1) Master Chief 2) Arbiter 3) Cortana

Q.4) Which bot would you want to play against again?
1) Master Chief 2) Arbiter 3) Cortana

Q.5.) Do you feel that fifteen minutes gameplay time was sufficient to differentiate between the three bots and their behaviours?
1 Yes 2 No 3 Undecided

Q.6) In a few words, describe what you felt about the experiment?
17.1.2 Feedback from participants

This section contains the participants exact responses to Question 6 of the bot rating questionnaire

As mentioned above
Master chief = RecurentBot
Cortana= FeedforwardBot
Arbiter= Static scripts bot

Q.6) In a few words, describe what you felt about the experiment?

"I enjoyed it. it was interesting to see a computer play in three different styles".

"Really fun a bit difficult to have definite opinion about three different bots".

"I felt the bots should have been more aggressive, at least one should have attacked head on. It was often the case of how quick I can follow this bot, so the more accurate I was the easier, whereas an aggressive bot would hunt me down, bash me over the head and then walk over me".

"It definitely showed which bots were different, Arbiter was probably the normal Cortana was adapted
Master Chief I wasn't sure, could be a harder level or adapted."

"Well set up provided a chance to rate each bot effectively".

"Would like to play the first one again, cause I don't know if he was the hardest one due to that. He was the first, I haven't been playing for a while".

"Interesting".

"Great way to experiment by involving a game it's a good way of demonstrating that games are fun can form a great part of education".

"It is interesting to see how a bot can have a human like gaming attitude (Arbiter) and due to their skilled ultra knowledge it forces the game player to re-think strategies and learn more".
“Three interesting bots, cortana was able to chase me and fight and win, master chief was very pleasing and to fight with and challenge. Arbiter was easy. overall I felt interested to play again and beat Cortana”.

“The combats were not much fun because the bot kept doing the same moves as strategy. Again it was fun when the difficulty level was easier for master chief and arbiter. overall I would prefer playing against human because it unpredictable, where as bots seemed to be base on same strategy. It would have been more fun if the bots had few stragegies and it was changing strategies randomly”.

“Good gameplay intresting concept would be good to have more bots ranging from novice/unskilled to highly skilled to determine difference in gameplay”.

“Very good, very fun but I feel I adapted more and got better as I played so that may have affected the results perhaps”.

“Intresting”.

“All three bots had a different style f combat/play and one certainly was harder to defeat because of its unpredictability”.

“Good mixture of bot playing styles”.

“Very entertaining: could easily tell the difference between the 3 bots. They displayed a variety of styles, close combat, long range and a combination of both”.

“I thought it was lots of fun and spend some time trying to guess which of the bot was the best to play against”.

Bhuman Soni
Page 85
22/01/2008