Kristian Tveiten
Martin Sondov Hallan

Evolving video game opponents with NEAT and evaluating its impact on brain activity with fMRI

Master's thesis in Informatics (MSIT)
Supervisor: Ole Jakob Mengshoel
Co-supervisor: Asta Kristine Håberg and Hallvard Røe Evensmoen
July 2023

Kristian Tveiten
Martin Sondov Hallan

Evolving video game opponents with NEAT and evaluating its impact on brain activity with fMRI

Master’s thesis in Informatics (MSIT)
Supervisor: Ole Jakob Mengshoel
Co-supervisor: Asta Kristine Håberg and Hallvard Røe Evensmoen
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

ABSTRACT

Neuroevolution, spesifically Neuroevolution of Augmenting Topologies (NEAT), offers promising potential in the realm of video games towards evolving dynamic and engaging artificial opponents. Its impact on the gaming experience was investigated by utilizing Functional Magnetic Resonance Imaging (fMRI), to analyze the brain activity of 13 participants aged 21 to 26 (With an average self-rated gaming experience of 3.76/5) while playing a dodgeball video game. The participants played against three different opponents: one evolved with NEAT, another implemented as a traditional Finite State Machine (FSM), and the last one trained via Multi-Agent Posthumous Credit Assignment (MA-POCA). The self-reported, game performance and fMRI data revealed trending and significant differences between all the agents, in addition to elevated activation in brain regions amygdala and nucleus accumbens during play against the agent evolved with NEAT, when compared to the other agents. These brain regions are commonly associated with emotion processing, learning and reward processing. Furthermore, self-reported data through questionnaires and game performance data highlighted the elevated game experience against the NEAT agent in terms of rated 'unique strategy', 'balanced challenge', 'sense of mastery', 'frustration', and 'entertainment value'. The opponent evolved with NEAT also outperforms the other opponents in terms of metrics suggesting a elevated state of flow for participants. The findings in behavioral and neurological data suggest that implementing neuroevolution in the development of in-game opponents significantly enhances the experience of player-versus-agent gaming.

SAMMENDRAG

Neuroevolusjon, spesielt Neuroevolution of Augmenting Topologies (NEAT), har lovende potensial innen videospill for utvikling av dynamiske og engasjerende kunstige motstandere. Dens innvirkning på spill-opplevelsen ble undersøkt ved bruk av funksjonell magnetresonansavbildning (fMRI), for å analysere hjernens aktivitet hos 13 deltakere i alderen 21 til 26 år (med en gjennomsnittlig selvrapportert spillerfaring på 3,76/5) mens de spilte et dodgeball-videospill. Deltakerne spilte mot tre forskjellige motstandere: en utviklet med NEAT, en annen implementert som en tradisjonell Finite State Machine (FSM), og den siste trent via Multi-Agent Posthumous Credit Assignment (MA-POCA). Den selvrapporterte-, spill- og fMRI-dataen viste trendende og signifikante forskjeller mellom alle agentene, i tillegg til økt aktivering i hjerneregionene amygdala og nucleus accumbens under spill mot agenten utviklet med NEAT sammenlignet med de andre agentene. Disse hjerneområdene er vanligvis forbundet med følelsesprosessering, læring og belønningsprosessering. Videre fremhevet den selvrapporterte dataen gjennom spørreskjemaer og spill-dataen den forbedrede spillopplevelsen mot NEAT-agenten når det gjaldt deltakerene sin tilbakemelding på metrikkene 'unik strategi', 'balansert utfordring', 'mestringsfølelse', 'frustrasjon' og 'underholdningsverdi'. Agenten utviklet med NEAT overgikk også de andre agentene når det gjaldt metrikker som antyder en høyere tilstand av 'flow' for deltakerne. Funnene i atferds- og nevrologiske dataene antyder at bruk av neuroevolusjon under utviklingen av spill-motstandere betydelig forbedrer opplevelsen av spiller-mot-agent spill.

PREFACE

We are very thankful for the opportunity to cooperate with so many talented and passionate people throughout the course of this project. This project could not have been completed without valuable help and insights from experts of distinct fields of research. A special thanks to our co-supervisors and close collaborators Asta Kristine Håberg and Hallvard Røe Evensmoen for their engagement and expertise in a field of research very foreign to us. They also assisted us with the process of collecting and analyzing the brain data. A huge thanks to our supervisor Ole Jakob Mengshoel, for giving us valuable assistance with his knowledge. We are grateful to you for trusting us with combining two master-projects into one, as we believe this led to an amazing task for us to undertake. A great thanks to Jonathan Jørgensen, for assisting us with great insights, specifically within game development. We would also like to thank all of the participants who dedicated their time and effort to help us complete our study.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **ANN** Artificial Neural Network
- **CTRNN** Continuous Time Recurrent Neural Network
- **EA** Evolutionary Algorithm
- **EDF** EyeLink Data File
- **FSM** Finite State Machine
- **fMRI** Functional Magnetic Resonance Imaging
- **GA** Genetic Algorithm
- **HP** Hitpoints
- **HUD** Heads-up Display
- **LOC** Lateral Occipital Cortex
- **MA-POCA** Multi-Agent Posthumous Credit Assignment
- **MCTS** Monte Carlo Tree Search
- **MRI** Magnetic Resonance Imaging
- **NEAT** Neuro Evolution of Augmenting Topologies
- **NN** Neural Network
- **NPC** Non Playable Character
- **NTNU** Norwegian University of Science and Technology
- **PCA** Principal Component Analysis
- **PFC** Prefrontal Cortex
- **ROI** Region of Interest
- **SPC** Superior Parietal Cortex

- **Trainer** A python program which infers actions for an agent in a environment and learning by doing so.
- **TTL** Transistor-Transistor Logic: A signal sent from the fMRI scanner meant to synchronize the experiment with the start of the scan. Usually in the form of a "S" keypress.
- **TWEANN** Topology and Weight Evolving Artificial Neural Networks

CHAPTER

ONE

INTRODUCTION

Video games are incredibly diverse, with various genres showcasing distinct aspects of how a player can interact with the medium. Some games emphasize player-centric experiences, puzzle solving or interacting with an environment in some way. Others incorporate agents within the game, requiring the player to either collaborate or compete against them to succeed. Some games are centered around being a player-versus-player experience, focusing on engaging in competitive or cooperative gameplay versus another human. Other games focus more on the player experience of a single player. In these games, the opposing or collaborative non-playable character (NPC) is often referred to as an agent. Video games, therefore, stand as a unique medium that offers a vast range of experiences, from deep, immersive single-player narratives to dynamic multiplayer environments, highlighting the diverse ways of interaction that differentiate it from other traditional forms of entertainment.

Player versus agent games vary a lot in their implementation of the agents. This often depends on the computer controlled NPCs purpose in said game. If their purpose is to be a cooperative agents, they will assist players. On the contrary, if they serve as opponents they should provide challenges or competition for the player. A common feature among games is the implementation of algorithms to create a predefined playing style configured by the developer.

While the video game industry has had enormous improvements in aspects like graphical and auditory fidelity over the years, the use of artificial intelligence (AI) in video games has been not evolved similarly. Recently, graphics have become so great in video games that the technology has reached a point where it is almost indistinguishable from real life [1]. With these aspects reaching such a high standard, developers have been increasingly focused on developing more advanced NPCs by applying Artificial Intelligence [2][3]. Video games differ widely based on their genre and gameplay, and many different types of AI are implemented that embody unique strengths and weaknesses for a broad range of game scenarios. While there is nothing inherently wrong with the state of game AI today, new types of AI could provide novel and interesting situations in videogames.

A machine learning system with very interesting properties and capabilities is Neuroevolution. This subfield of artificial intelligence involves using genetic algorithms to evolve artificial neural networks (ANNs) through evolutionary processes manipulating the weights, topology, and configuration. A core strength of neuroevolution is its ability to explore the problem space efficiently while keeping track of and protecting novel solutions. Given these properties, agents developed with such methods could provide greater enjoyment, immersiveness, or novelty to players of commercial games. As previously stated, the field of implementing advanced AI in commercial games is still relatively unexplored, especially when it comes to learning models. Applying this type of AI in games can improve novel AI in games, and exchange large amounts of programming expenses for computational expenses.

The motivational force behind this project is to implement AI to evolve agents that players experience as interesting and rewarding to play against. Evaluating the player experience can be complex, and achieving objective results can be challenging due to biases of different evaluation methods. For this project, advanced methods eliminating subjectivity bias from the analysis of player experience during game play were applied. Neuroevolution was compared to two other methods of implementing AI in games, where the player experience for each agent were assessed by the means of questionnaires, game behaviour data, functional magnetic resonance imaging (fMRI), and eye-tracking. One of the methods implemented for comparison with the neuroevolution agent is a Finite State Machine (FSM). This is a highly traditional way to handle NPC behavior in games, relying on states and the transition between them to enact their behavior. The final method is a type of Reinforcement Learning (RL) with attention networks. RL is a very well developed type of AI with good results in general problem solving.

fMRI is a neuroimaging technique that allows us to examine brain activity by detecting changes in oxygenation levels in blood in small blood vessels in the brain. When a brain region is more active during a task, a feedforward mechanism leads to increased blood flow to that region. The increase in blood flow supersedes the oxygen needs of the region leading to a local increase in blood oxygenation level, i.e., blood oxygen dependent (BOLD fMRI). By using an MRI scan technique sensitive to differences in blood oxygenation level and scanning dynamically during task performance, it is possible to capture these changes, and thereby indirectly map brain activity related to different tasks. This makes fMRI a valuable tool for studying the neuronal correlates related to playing against different agent. We use this data to investigate if different agents engage different brain regions such as the frontal cortex, amygdala, and nucleus accumbens, but also to measure activity of the whole brain.

The model presented in Figure 1.0.1 is a highly abstract illustration of how this study was conducted. Participants were placed in an fMRI scanner with an eye tracking camera attached to the head coil where the participants head is placed. Also mounted in the head coil is a periscope, allowing the participant to see a screen located behind the fMRI scanner. This way, participants can perform tasks on the screen with hand held controllers.

This approach allows research beyond subjective self-reporting and performance data and gives a more nuanced picture of how players respond at a neuronal level to various game scenarios and AI behaviors. Our study provides insight from a new type of evaluation of the success of neuroevolution-based AI to engage players, and valuable insights into the nature of player-AI interaction in video games.

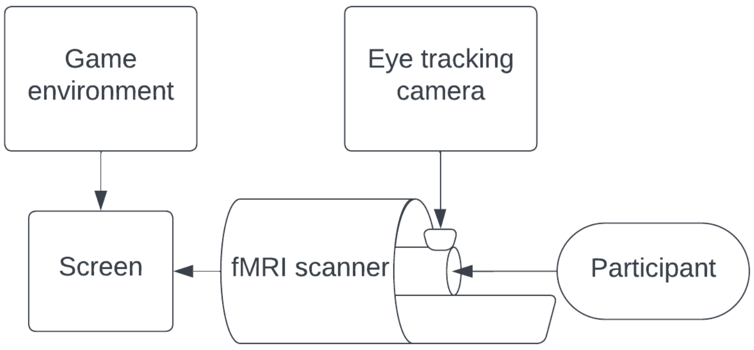


Figure 1.0.1: An abstract model of how the study was performed

1.1 Project statement and research questions

The research question for this thesis is as follows:

Problem Statement. *How does neuroevolution applied to opponents in video games impact the player's neuronal responses?*

The following research questions are explored to answer the problem statement:

Research Question 1. *What types of agents are suitable candidates for a comparative study between different video game agents?*

To answer this question we utilized a unity environment suited for both traditional gaming agents, and more advanced agents controlled through advanced AI techniques. We used a player versus agent environment to isolate the players perception of their opponent. To adequately compare agents with unique behaviors to each other, we thoroughly researched the field of gaming AI, to find agents we find suitable to compare. These agents were developed with the goal of exhibiting diverse and unique behavior, and to be evaluated through different metrics to see which agent received the best response.

Research Question 2. *Are there differences in brain activity and player perception related to the different agents?*

Comparing a players subjective impressions of the different agents behavior requires several forms of data acquisition. Differences in brain activity can be found through fMRI. Analyzing both whole brain activity and specific regions of the brain provides insights in how players subconsciously perceive differences in the agents. Through self-reporting, the player is allowed to provide their own impressions of the agents.

Research Question 3. *Can neuroevolution enhance gaming experience by generating novel behaviors in AI opponents?*

This question will be answered by judging the participant response by playing against the in-game opponent evolved with neuroevolution. Does the opponent evolved with neuroevolution create value to the game compared to other game AI methods? Is the behavior exhibited by agents evolved with neuroevolution notably different from other game AI methods?

1.2 Research method

The research method consisted of three main phases. First, a development phase where the necessary artefacts are developed, i.e., the games to be played and the agents proposed for play. The development phase was split in two, with a pre-pilot study performed halfway to assess points of improvement and guide further development towards the study goal. After development ended, the project shifted into the second phase to perform a research study in participants playing the game during fMRI. Finally, the third phase consisted mostly of analyzing the data from the second phase. The process are covered in this paper to elaborate the whole research study from how it was conducted, to discussing the results.

1.3 Preliminary process

In the fall of 2022, a month long research and planning project was conducted to explore and investigate the subject areas relevant in this project. Initially, the project started as a development project to investigate the capabilities of evolutionary/genetic algorithms in video games. Subsequently, a collaboration project with St. Olavs Hospital emerged, incorporating fMRI as the primary evaluation method in addition to other techniques such as eye tracking and questionnaires. Consequently, a significant portion of the time during the fall was dedicated to planning out the fMRI study and finding relevant and practical technologies to aid the project's progression.

Readers of both papers will find many differences in the projects, as much of this project changed even after the preliminary project had ended. One core difference is the transition from a 2D game environment to a 3D environment. The preliminary project investigated the use of ANNs as genes in evolutionary/genetic algorithms, but had not yet concluded on using neuroevolution as the selected investigative AI method. These selections, along with many others were made based on the research conducted through the preliminary process.

1.4 Thesis structure

This chapter covered the introduction and motivation behind the paper, as well as presenting the problem statement and research questions for the project. The background for this thesis are presented and explained thoroughly in chapter 2, elaborating on relevant technologies, methods, and concepts. Chapter 3 focuses on closely related work, providing additional insights into the specific research niche addressed in this thesis. Chapter 4 presents the methods and implementations made to carry out the study. In chapter 5 the results are presented and the findings are discussed in chapter 6. Finally, the findings are concluded in chapter 7.

CHAPTER
TWO

BACKGROUND & THEORY

In order to delve into the application of neuroevolution in the development of an AI for a video game, it is essential to have a thorough understanding of the components involved. To begin, we must first explore the field of video games. More specifically the engagement of player versus agent scenarios, where the development of captivating and engaging NPCs plays a vital role in creating a rewarding gameplay experience. This is followed by an exploration of evolutionary/genetic algorithms and its role in the current state of AI. The concept of neuroevolution holds significant potential in enhancing the effectiveness of AI in video games, making it an area worth investigating in detail. Finally, we will provide insight in the technology of fMRI, which allows for a non-invasive measurement of neural activity. We will explain how this can provide insight into how the brain processes information during gameplay. Relevant frameworks will be presented and some relations will be made between highly relevant papers and this study. In this chapter, we will provide a comprehensive overview of the concepts presented and their relevance to the research question of this thesis.

2.1 Video game theory

The development of video games has undergone significant changes over the past few decades. Initially, video games were primarily text-based and limited in their graphical capabilities. As technology progressed, particularly with the advent of more powerful microprocessors and graphics processing units, video games began to incorporate more advanced graphics, sound and gameplay mechanics. This has led to a wider range of genres and styles of games, as well as an increase in the level of realism and immersion that can be achieved.

Today, video games are a wide spread medium and the definition of a video game is becoming increasingly vague, as developers use more and more freedom to convey the gameplay and narrative for their game. This has developed to the point where different approaches to video game studies have formed, with two leading views. The first one is ludology, which seeks to study video games as a form of play, focusing on the gameplay mechanics and the rules that make up the game. Narratology on the other hand, studies video games as a form of storytelling. It focuses on the narrative elements of the game to convey a story, develop characters or create a theme for the game.

Given the nature of our study, a narratologist approach would not be a sensible way to develop the proposed game. We will be performing studies on subjects that do not have the luxury of time to develop relations to characters, nor will they have time to experience a story. Though this is proven to be a highly effective way to create immersion in video games[4], it does not fit very well for this study. We must also factor in that we are trying to observe interest in the behavior of an agent, not a fictional character. Having a narratologist approach would have a negative impact on these observations, as the subjects might be more or less invested because of their relation to the narrative presented.

A ludologist approach makes much more sense for this study, as both the player and the proposed agent will benefit from interacting with the gameplay mechanics and game rules. Having game rules that allow the player and agent to observe the rules in the same manner minimizes unwanted unfairness. For an intelligent agent, a rational decision is a good decision made with the available information. Since the agent receives information about the game state constantly, it is important that the player also receives important information so they also can make rational decisions. The proposed game should therefore have rules and visual representations that allow both player and agent to perceive the state of the game in the same manner.

Such states can be how to win the game, or how the game is lost. What actions lead to the winning state, and what actions puts the player closer to a loss. While this information is important, it is still crucial to keep a certain level of complexity in the game to prevent it from becoming monotonous. Both the player and the agent need a certain level of freedom in how they achieve their goals, as this is the space we will find the interesting behaviors on both ends.

2.2 Video game AI

Analyzing video game AI through a ludologist perspective provides an objective view of the behavior of an NPC, devoid of emotional attachments to the character itself. When viewing the behavior of an NPC in a video game, some questions are raised in relation to the entertainment provided by the NPC. At the root of these questions lies the biggest one: What makes an NPC interesting or fun to play against? This section will provide some background to this question, which will later be discussed in detail to provide some insight in this interesting topic.

The lowest level of autonomy found in video game AI is scripted AI. AI autonomy is an industry term that refers to all or groups of AI agents in a video game. While higher levels of autonomy imply higher levels of intelligence, lower levels of autonomy are more hard-coded by developers to perform a purpose. Scripted AI are normally not completely scripted, but switch to high levels of autonomy when needed [5].

Still a low autonomy system, but certainly higher than scripted AI are finite state machines (FSMs). These have been popular in games for decades, particularly for NPC behavior, because they provide a clear and easy-to-understand structure for managing an agent's behavior [5, 6, 7, 8, 9]. These are computational models that consist of finite states, transitions between those states, and actions that can be executed while in a particular state.

Suppose we have an agent with two states: 'Patrol' and 'Attack'. The FSM for this NPC could follow a predefined path until detecting a player within a certain range. This would cause the NPC to transition from the 'Patrol' state into the 'Attack' state. In this state, it will pursue the player and attempt to deal damage, until it no longer perceives the player in its range anymore. At this point it will return to the 'Patrol' state in an attempt to locate the player.

While this is a simple agent, the decision making of the agent grows in complexity together with increasing complexity of the environment, giving the agent more states and actions to perform. A more complex agent can perceive the environment to find cover, duck and hide from cover while attacking the player, relocating to new cover, fleeing away from the player and much more. Complex agents often use fuzzy logic to determine the transitions between states. Fuzzy logic is a form of multi-valued logic that allows for reasoning with imprecise or uncertain information by using degrees of truth rather than binary true/false values. In addition, many games use other AI techniques in combination with FSMs and fuzzy logic, such as behavior trees or utility systems to create more complex and realistic behavior[6, 10].

There are multiple factors contributing to the popularity of low autonomy systems in the field of video game AI. One significant reason is their ease of programming, making them accessible to developers with varying levels of expertise. Additionally, these systems are relatively simple to control, allowing developers to achieve specific desired behaviors for non-player characters (NPCs) in the game. Game development companies also tend to favor these systems due to them being more developed, having far more toolkits and frameworks for development[5]. Another factor is their trait of predictability. The ultimate goal of game design is to create enjoyable experiences for a wide range of players. What one player finds enjoyable may be very different from what another does. Many argue that predictability in video game AI puts the player in control of situations, providing a stable and entertaining experience[11].

While some argue that predictability in video game AI is good, others argue that the inferior intelligence of NPCs degrades the overall quality of the gameplay. Wang et al. [3] argue that low autonomy systems have two major weaknesses; *Loopholes* *and predictability.* Low autonomy systems incorporate these flaws in a way that allows players to exploit their faults, which can lead to negative game loops. Over extended periods of gameplay, NPCs might appear repetitive and uninteresting, negatively impacting game engagement and reducing overall playability.

Reddemann [12] proposed that *applying neural networks to games can make them more interesting and increase their playability.* This idea of more interesting individual NPCs with potentially academic AI controlling their behavior could be an interesting solution to problems addressed by Duarte et al.[2]. In their survey from 2020 they elaborate on the need for more complex AI in video games:

> *This growing interest has also been fueled by an increasing commercial interest by the gaming industry on finding solutions to create games that are more enjoyable, challenging and engaging, and that ultimately provide a more personalized experience to the human players, in the form of, for example, more sophisticated AI (i.e., non-player characters or NPCs) that exhibit more complex behaviors and skills and act more naturally and human-like.*

There have been some landmark developments in the gaming industry over the years. One significant contribution was the 2005 video game "F.E.A.R.". This game introduced Goal-Oriented Action Planning (GOAP) [6], a dynamic departure from traditional FSMs. GOAP provides AI characters with a set of possible actions and a goal, using a planning algorithm to determine an effective action sequence based on the current game state. This method facilitates complex, adaptable behaviors, significantly enhancing NPC intelligence and the overall gaming experience. Despite its computational demands and intricate implementation, GOAP's introduction in F.E.A.R. spurred a new era of innovation in video game AI, highlighting the potential for advanced NPC behavior beyond pre-defined state transitions. Many modern commercial games use planners to model the behavior of their NPCs, such as Tomb raider, Middle Earth: Shadow of Mordor and Dying Light.

While the planning capabilities of AI, as demonstrated by F.E.A.R.'s GOAP system, greatly enhance the adaptability and strategic depth of NPCs, another dimension of AI sophistication is represented by learning systems. These systems, rather than making decisions solely based on pre-programmed logic, incorporate behaviors learned from data. A compelling illustration of this concept is seen in the popular online game Dota 2, where the developer Valve Corporation implemented an AI developed by OpenAI, OpenAI Five, trained through reinforcement learning[13]. The AI played millions of games against itself offline over a 10-month training period, learning effective strategies and behaviors which were then deployed in real-time gameplay. This AI ended up winning 99.9% of the 7000 games played, even beating the world champion team, OG, in a tournament match.

Machine learning models represent a unique type of AI technique that is applied to large volumes of data. Through extensive training, these models can develop their own form of intelligence, which can be utilized to create intelligent in-game agents capable of solving complex problems. These agents, also known as 'learners,' can even continue to learn and improve their actions during gameplay. Despite the demonstrated success of learners in beating world-class players in both deterministic and stochastic game environments [14, 15, 16], their adoption in commercial games has been limited. This hesitancy might be attributed to the fact that many breakthroughs with learning AIs have resulted in superhuman levels of performance, which could be overwhelming and undesirable for players in games designed primarily for entertainment.

Despite advancements with systems like GOAP and OpenAI Five, the potential for more dynamic, adaptable, and engaging NPC behavior remains largely untapped. The potential for resource saving and the creation of novel and engaging AIs through the use of learning systems in video games is significant, opening up exciting possibilities for the future of game design.

2.3 Reinforcement learning

Reinforcement learning (RL) is a machine learning approach to find an optimal solution in a search space by maximizing the notion of cumulative reward gathered in a given environment. Agents train their intelligence by continuously interacting with the environment, while receiving feedback in the form of rewards. These rewards help the RL agent adapt its policy, which makes the agent learn to make better decisions to maximize the reward. It is completely unsupervised without the need for labeled data. An example of how the process of interacting with the environment and receiving rewards can be seen in Figure 2.3.1.

Reinforcement learning is a common term for many different approaches. The policy in RL can be model-based or model-free, where model-free methods include algorithms like Q-learning and actor-critic. Model-based methods like monte carlo tree search (MCTS) employ an approach where the agent learns and forms an internal model of the environment to make decisions. Algorithms like MCTS are often used in games with deterministic environments, such as chess and go, but are also used to create intelligent NPCs.

However, for stochastic environments, model-free reinforcement learning is a more suitable choice. Stochastic environments often become too complex to create an internal model of, instead requiring a model free approach where decisions are made solely on the experience and observations within the environment. One of the most popularly applied model-free deep reinforcement learning methods is Deep Q-Network (DQN). DQN utilizes a neural network to approximate the Q-value function, which maps a pair of state and action to a predicted cumulative reward. The policy, which decides which action to take in a given state, is derived from this Q-value function. The learning occurs through interacting with the environment, with positive or negative rewards received based on the outcomes of actions. These rewards influence the future probability for selection of actions in given states. However, the last action performed in a state is not always solely responsible for obtaining a reward. In a Markov Decision Process-based environment, the state-action mappings that led up to the final state-action which obtained the reward are also updated, reinforcing the sequence of decisions that led to the reward.

One disadvantage of reinforcement learning is it can be quite challenging to design the reward function responsible for defining what the agent is trying to maximize. RL is suitable for environments where there is not one static optimal solution where the environment is dynamic and uncertain. The RL algorithm with a suitable policy implementation can learn to solve complex tasks and can capture a wide variety of state to action mappings. However, RL can have trouble with training stability and applying learnt policies to new scenarios.

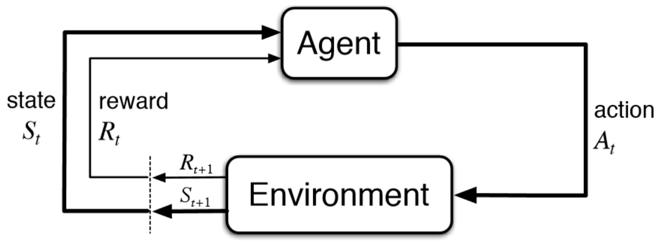


Figure 2.3.1: *The figure shows how the reinforcement learning process works, where an agent interacts with the environment and receives a reward [17]*

2.4 Genetic algorithms

Genetic Algorithms (GAs) are a class of optimization and search algorithms within the broader field of Evolutionary Algorithms (EAs). GAs leverage mechanisms inspired by the principles of biological evolution such as reproduction, mutation, and selection to efficiently explore solution spaces. Genetic algorithms are usually inspired by Charles Darwin's theory of evolution by natural selection, though some EAs are also inspired by Lamarckism, such as the previously mentioned Alphastar[16]. The key difference between Darwinism and Lamarckism, is the Lamarckism theory of individuals acquiring traits during a lifetime and passing it down to offspring rather than from natural selection supported by Darwinism. Lamackism EAs often optimize the solutions using RL alongside the selection, crossover and mutation operators.

Genetic algorithms are population-based metaheuristic optimization algorithms. This means they work with a population of individuals which represents potential solutions to a problem, and use the mechanisms inspired by evolutionary biology to iteratively refine these solutions. The core idea behind EAs comes from Darwin's theory of natural selection, which can be summarized as "survival of the fittest". The fittest individuals from a population are selected for reproduction, giving rise to offspring with enhanced traits in successive generations.

Each individual in a population is encoded as a data structure which mimics a biological chromosome. These data structures are often referred to as genotypes, and can be represented in many different ways. As long as the chosen data structure supports the operations which EAs require like crossover and mutation, any data structure can be used for evolution.

The instance of a genotype is called a gene, and is the medium of where the optimization happens. The genetic encoding can be applied to binary strings, real-valued encoding, permutation encoding, tree encoding and graph encoding. Multiple genes form a genome that gives rise to physical traits in the task at hand, which are collectively referred to as the phenotype.

Just like evolution in the real world, the population of individuals are set out into an environment with a given task to be solved. Where only the strongest individuals are allowed to generate offspring based on their genes. The performance of solving the task is evaluated by a pre-defined **fitness function**.

The fitness function denotes the objective of which the individuals should aim to achieve. Different implementations of the fitness function can have great impact on the resulting individuals from evolution. The fitness function should be as general as possible to avoid solutions exploiting unwanted behaviour to trigger specific rewards. At the end of the generation after every individual has been evaluated, a set of the fittest individuals are selected during the **selection** phase. The selection method is task independent and can be implemented in many ways. However, the most prominent selection method is fitness proportionate selection. Genomes are picked by chance, where the genomes are assigned a probability of being selected proportional to their fitness. The selected genomes are then subjected to genetic combination or **crossover**. During the crossover stage, the parent individuals are combined in a implementation specific manner which forms the genetic material of the descendant. The crossover is usually performed with pairs of individuals. The crossover algorithm can be manipulated in many ways for different results. Crossover can happen by passing down alternate genes from each parent, or passing down the first half from the first parent then other half from the other parent.

With regular crossover, a problem can occur where the genes stagnate due to lack of variation in the population at a given time. To remedy this, *mutation* is used. Mutation is a genetic operator which randomly changes genes to explore the search space and possibly find new optimal solutions. Mutation can be set to happen at each crossover, following some criteria or in a random manner. An illustration of crossover and mutation is presented in Figure 2.4.1.

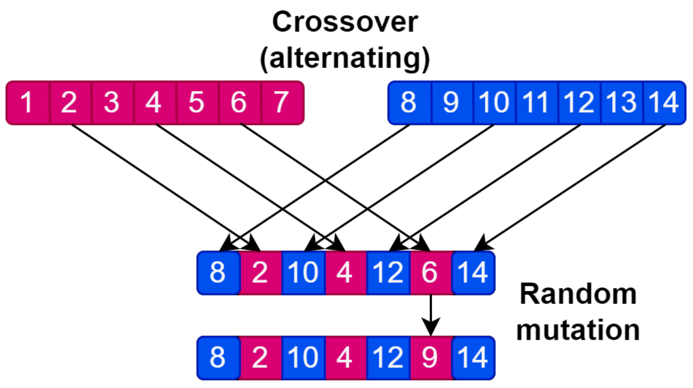


Figure 2.4.1: *Simple crossover example with alternating genes passed down from each parent. Gene number 6 of the descendant is mutated.*

The individuals with the best fitness score can also be carried over to the next iteration without manipulation of the genes by a process called **elitism**. Elitism is a technique in genetic algorithms where the best performing individuals, or the "elite," are carried over to the next generation without alteration. This ensures optimal solutions are preserved and speeds up convergence by counteracting the shuffling and stochasticity of crossover and mutation. However, excessive use can lead to premature convergence to sub-optimal solutions, thus it's balanced with crossover and mutation for population diversity.

With selection, crossover and mutation, GAs can find high-performing solutions quicker than traditional methods like reinforcement learning [18]. This is due to the stochasticity of mutation (and possibly crossover if a stochastic algorithm is chosen). Desired traits can quickly be found by luck. Mutation can of course also deteriorate the traits of a given descendant solution from being the optimum.

2.5 Neuroevolution

The most prominent application of genetic algorithms has been to evolve genes that are readily interpretable to humans, as elaborated in Section 2.4. This is useful in a variety of areas, such as game rules, mechanics, or decision-making. For complex applications where the traditional gene encodings are restrictive, the field of evolutionary computation has given rise to an innovative approach - neuroevolution. Neuroevolution integrates genetic algorithms with the complexities of artificial neural networks, facilitating search optimization in intricate and high-dimensional problem spaces, even though the resulting solutions may be less immediately understandable to humans.

Neuroevolution applies genetic algorithms to develop ANNs by manipulating the network weights. Many neuroevolution techniques operate with a fixed topology, which is pre-determined by the developer before launching the evolutionary training process. An illustration of how neuroevolution can occur is presented in Figure 2.5.1.

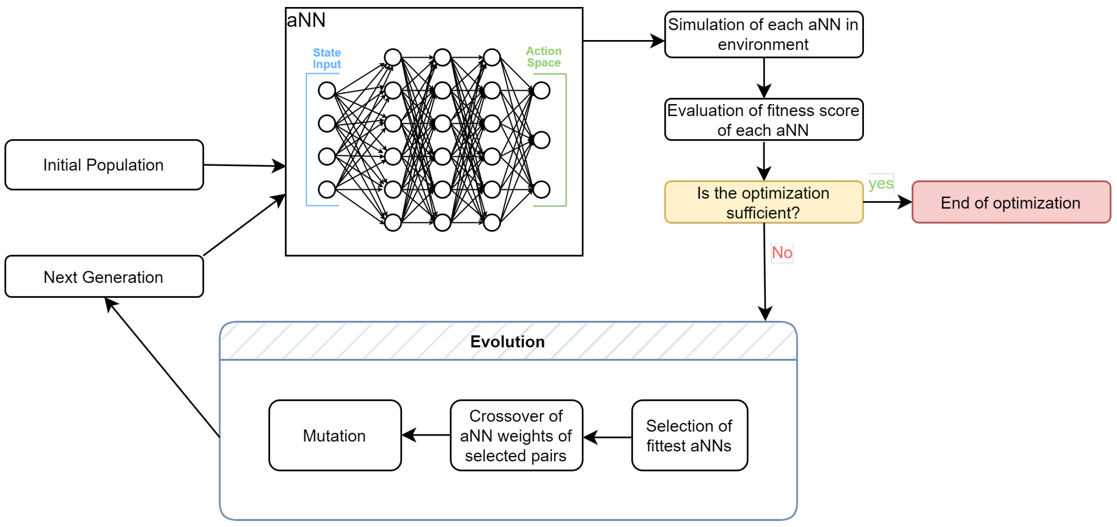

aNN
State Input
Action Space
Initial Population
Next Generation
Simulation of each aNN in environment
Evaluation of fitness score of each aNN
Is the optimization sufficient?
yes
End of optimization
No
Evolution
Mutation
Crossover of aNN weights of selected pairs
Selection of fittest aNNs

Figure 2.5.1: *The genetic algorithm process of generating a population of artificial neural networks and evolving them by their weights*

As with many other machine learning techniques applied to neural networks, deciding the network size and layout can be challenging. Too small of a network can diminish the network's ability of capture the patterns and relationships in the data necessary to solve the task, resulting in underfitting. Underfitting occurs when the model is too simple and fails to generalize well to unseen data. On the other hand, overly large networks may have too much capacity and can overfit the training data, meaning they become overly sensitive to noise or random fluctuations in the data. Lastly, one of the worst aspects of a network size too big in the context of neuroevolution, is increased computational requirements. This can lead to training sessions taking a substantial amount of time, in addition to large networks hindering the performance of games of which its applied to. Striking the right balance between model complexity and generalization is essential to achieve optimal performance.

2.6 NEAT

In contrast to traditional neuroevolution, NeuroEvolution of Augmenting Topologies (NEAT) adopts a unique strategy. Rather than solely encoding the weights of the neurons in the networks as the genomes, NEAT also represents the actual topology of the network within the genome. As a result, NEAT encompasses two types of genomic components: nodes and connections. The node genes signify the presence of nodes in the network, while the connection genes define the weight of a connection and whether it is enabled or not [19].

2.6.1 Evolving both network topology and weights

An artificial neural network is a computational model inspired by the functioning of nerve cells in the brain. It consists of several layers, including an input layer, hidden layers, and output layers as seen in Figure 2.6.1. The nodes in the input layer are the entry points to the neural network, while the output nodes represent the outputs from the network. The hidden nodes receive neural activations from other nodes and send activation signals to others. Each edge or connection in the network has a positive or negative weight, a scalar value that modifies the neural activation. The equation that calculates the output of each node can be seen in Equation 2.1. This equation calculates the output value y using the inputs to that particular node x_j, weights for each of the incoming connections w_j, activation function f and bias b. This is summed over m inputs. These weights can be tuned intricately to learn complex patterns in data.

$$y = f\left(\sum_{j=1}^{m} w_j x_j + b\right) \tag{2.1}$$

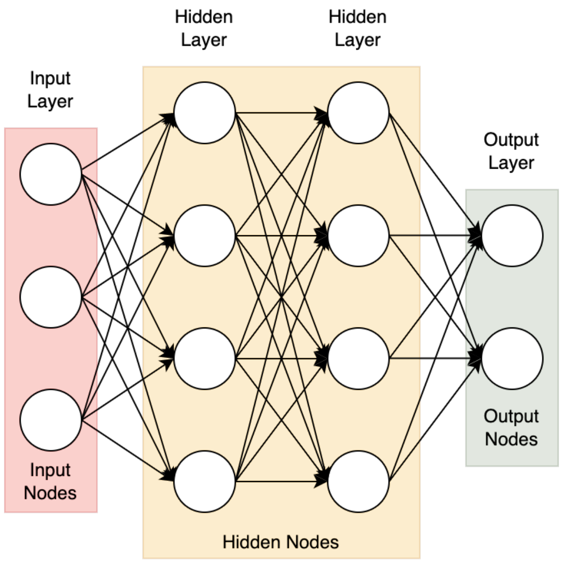


Figure 2.6.1: Example of a artificial neural network structure

Regular neuroevolution methods only use the weights as genomes and subject for evolving. NEAT treats both the neural net weights and the actual network structure as genomes. Meaning following selection, crossover and mutation, both the best weight values and network structure is inherited. This leads to the task being solved with a small network initially, but growing more complex with more complex behavior after prolonged training. It ensures that the network size is appropriately sized in order to solve the task. Other methods like Topology and Weight Evolving Artificial Neural Networks (TWEANN) generates random initial neural networks, where networks are larger initially than NEAT with no justification behind the initial complex network structure.

NEAT does not employ the concept of layers, instead the order is denoted by inputs and outputs in relation to that given node. An example of the way NEAT structures the neural network can be seen in Figure 2.6.2.

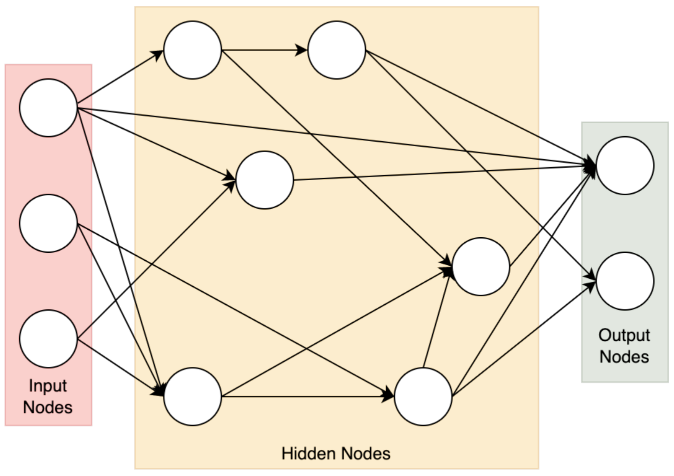


Figure 2.6.2: Example of a artificial neural network structure evolved with NEAT. Notice the lack of layers and partly direct input to output connections.

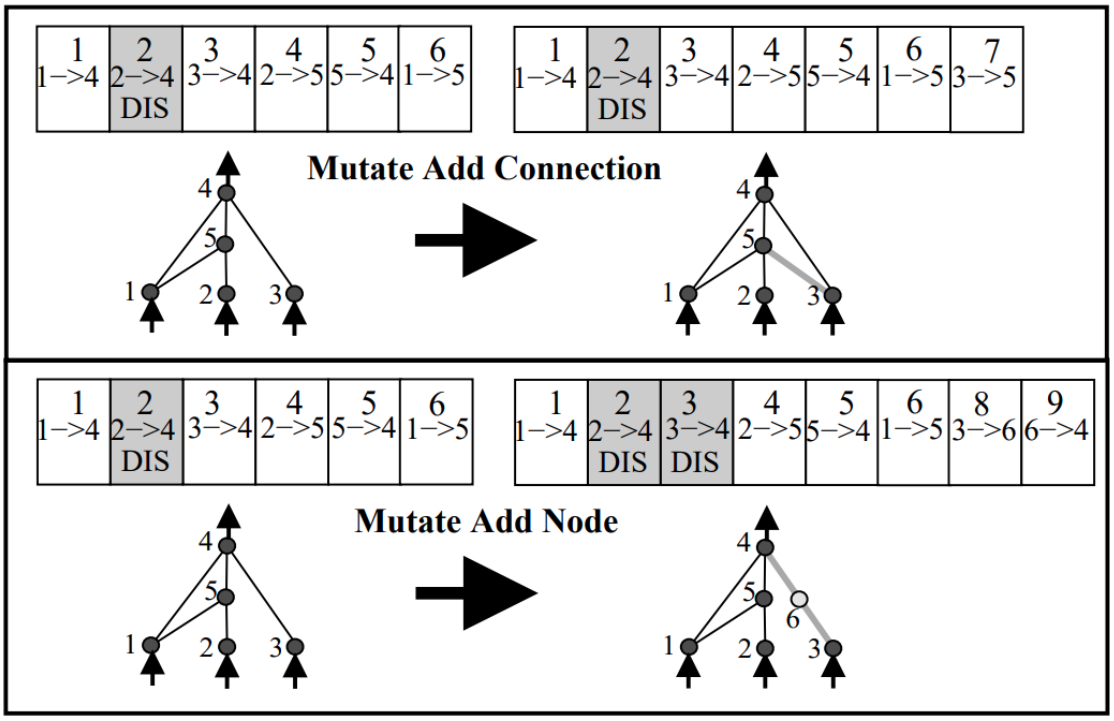


Figure 2.6.3: Structural mutation in NEAT, adding nodes and connections between nodes. The connection genes are shown above their phenotypes. The top number denotes the order in which each node was added to the network (called *innovation number*. When a new connection happens, the connection gene that is being split is disabled, and two new genes for each part of the connection is then added in addition to the new node which caused the split [19].

Network topology grows by mutation where new nodes are added to existing connections and new connections between existing nodes as seen in Figure 2.6.3. NEAT also keeps a innovation number on each connection gene, identifying the connection across multiple genomes. When a node is added to a existing connection, the old connection gene is disabled, and two new connection genes forming the split connection are created. Each of the new connection genes also receive an incrementing new innovation number denoting the order of creation.

2.6.2 Genetic encoding

The genetic encoding in NEAT is designed to easily align when pairs of genomes mate. The genotype in NEAT consists of genes for both nodes and connections, with the connection gene being the most significant due to the information contained. It keeps information identifying the input node at the start of the edge and the receiving output node at the end of the edge. In addition to the connection's weight or strength, the connection genes also maintain information about their enabled or disabled state. Lastly, the innovation number denotes the sequence in which nodes were added, allowing the algorithm to locate corresponding genes for crossover by tracking history.

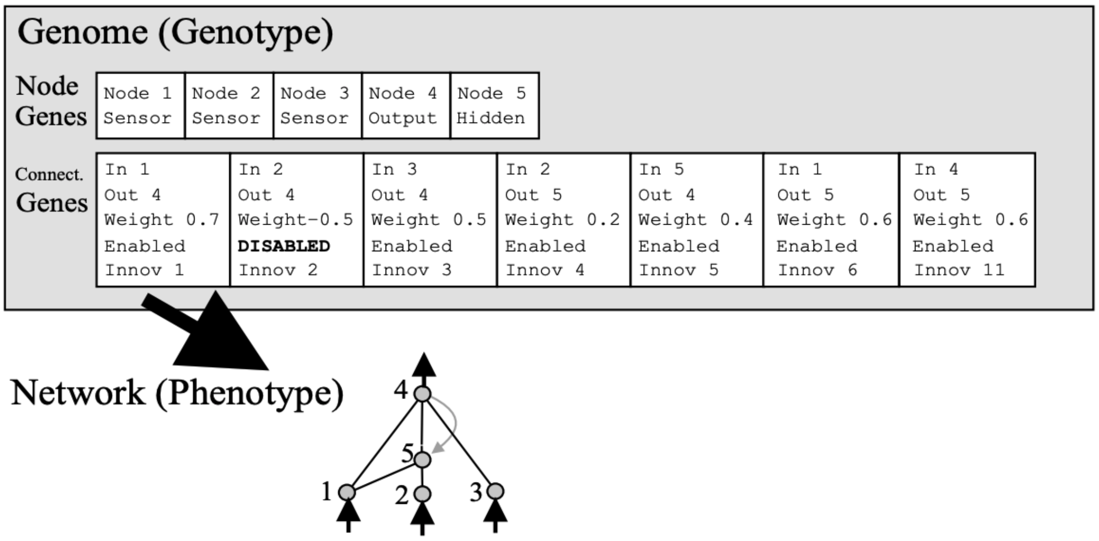

Genome (Genotype)
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Output
Node 5 Hidden
Connect. Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight-0.5 DISABLED Innov 2
In 3 Out 4 Weight 0.5 Enabled Innov 3
In 2 Out 5 Weight 0.2 Enabled Innov 4
In 5 Out 4 Weight 0.4 Enabled Innov 5
In 1 Out 5 Weight 0.6 Enabled Innov 6
In 4 Out 5 Weight 0.6 Enabled Innov 11
Network (Phenotype)
4
5
1
2
3

Figure 2.6.4: Genetic encoding in NEAT illustrating the difference between node and connection genes. Figure from Stanley et al. [19].

2.6.3 Crossover

Performing crossover on continually evolved network topologies seems like a daunting task. If the network topologies do not match entirely, it becomes crucial to determine how to effectively combine them. NEAT offers a clever solution to this problem. The algorithm keeps track of which nodes that are similar in structure by keeping the historical origins of each node. As previously mentioned in Section 2.6.2 this history is called *innovation number*. This is a number that denotes the order of each node acting as identification. These innovation numbers are never changed, and are inherited by offspring. The numbers are also incremented for each new connection gene that is added.

By analyzing these innovation numbers, NEAT can assess which connections are the same structure and then knows exactly which genes match up to perform genetic crossover. For disjoint genes (genes that do not match in the middle of the network) and excess genes (genes that do not match in the end of the network), they are inherited to the descendant from the most fit parent. In case of equal fitness between both parents, the disjoint and excess genes are inherited randomly like the matching genes.

2.6.4 Speciation

When networks change due to new mutations, the changed genomes might perform poorly at first, even though their unique features are crucial for solving the task. These genes might not survive beyond a few generations. To mitigate this, NEAT incorporates the concept of speciation, also known as *niching*. This mechanism which allows genomes to compete primarily within their own niches rather than against the entire population. This allows new innovations to live on and evolve without being killed off right away. If these innovations do not yield improvements, the species can become extinct if outperformed by another species [19].

Speciation also serves to prevent premature convergence on a local optimum.

If the entire population converges on a local optimum, other co-existing species can discover optima closer to the global optimum. The species approaching this global optimum will grow in size, exceeding the size of the stagnated population. If the stagnated population reaches zero in size due to no improvements over the better species, the stagnated population is considered extinct.

Genomes are separated into their own species determined by their genetic distance. The more disjoint two genomes are, the less compatible are they for crossover. The measurement of genome compatibility distance δ involves a straightforward linear combination of three factors: the count of excess genes E, the count of disjoint genes D, and the average weight difference $\bar{W}$ of matching genes (including those that are disabled):

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 * \bar{W} \tag{2.2}$$

The coefficients c_1, c_2 and c_3 allow for adjustment of each factor's importance. The factor N denotes the number of genomes in the larger genome. If both genomes consist of less than 20 genes, the N factor can be set to 1.

2.6.5 Fitness sharing

Once genomes has been evaluated and the fitness has been calculated, it is shared within each species' niche. This is called *explicit fitness sharing*. Explicit fitness sharing in genetic algorithms was introduced by Goldberg et al. [20], and applied to the NEAT algorithm by Stanley et al. [19]. Explicit fitness sharing is a key component in NEAT which preserves genetic diversity in smaller species with topological innovation. With this method, a species cannot afford to become too large regardless if it contains several well performing individuals. Every species must share a fitness value of one, where the species with lower population densities will receive a large share of this fitness pool giving them a higher chance of further selection. Larger species with a big quantity of high performing individuals is punished by having to distribute the same total fitness resource across more individuals.

The adjusted fitness f'_i of organism i is determined based on its distance δ from each of the other organisms in the population:

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(9, j))} \tag{2.3}$$

Where sh is the sharing function which is set to 0 if the genetic distance between organism i and j is above the threshold δ_t. If the genetic distance is below the threshold (meaning genome i it does not represent significant innovations over genome j, and thus is similar), the sharing function is set to 1 [21]. The genetic distance threshold δ_t hyperparameter is configured before training. All species are then potentially assigned a different amount of offspring proportional to the sum of adjusted fitness f'_i of its member organisms [19].

During reproduction the lowest performing members are eliminated from the population. Offspring of the remaining organisms in each species then replace the entire population (as long as the fitness improves within a configured number of generations, otherwise only the two top species are allowed to reproduce, focusing on the most promising species).

2.6.6 Mutation

The mutation mechanism in NEAT works by adding connections and nodes. A new connection, with a random weight, is added between two pre-existing nodes. While a new node is added within an existing connection, effectively splitting it. The old connection is disabled, a new connection is added leading into the new node which receives the weight of 1. The connection from the new node leading out receives the same weight as the old connection. This method ensures that the node receiving the input from the new node receives the same weight as the original connection.

In other neuroevolution methods with evolving topologies and weights, such as TWEANN, mutations can result in significantly decreased fitness for the subsequent offspring. This is due to the fact that randomly initialized weights often make little sense without some modification, requiring a few generations until the weights are adequately adjusted.

2.6.7 Why neuroevolution?

Neuroevolution has demonstrated exceptional performance in specific tasks, including the pole balance problem, a well-established benchmark challenge for reinforcement learning techniques. Neuroevolution methods show the ability to find solutions with fewer number of evaluations than methods based on reinforcement learning for non-Markovian environments [22]. The research carried out by Andersson [18] studied the distinguishing characteristics of NEAT and RL. For the same training time, NEAT produced slightly better scores in a Super Mario like game, meanwhile ending up with a significantly smaller ANNs compared to the fixed-topology neural network of the RL method. It can be argued that the ANNs NEAT produces are not as complicated as the ANNs RL produce, and it requires longer time than RL to reach the network size that would be suitable for the task.

On the other hand, NEAT manages to find the simplest networks that solve the task instead of starting out with a large overhead in terms of network complexity. Neuroevolution is also easily **scalable**. Being able to handle large action/state spaces, especially when used for direct action selection. One of the most important factors of what makes neuroevolution great for this thesis, is the way **diversity** is handled. Due to the speciation in NEAT, different species with diverging approaches to solving the problem are able to compete against each other in terms of fitness. In reinforcement learning, there is only one policy being trained, which does not allow testing of multiple strategies at once.

The diversity of species with their own emerging strategies is made possible by **open ended learning** in NE. In RL, a state gives a reward and the historical states leading up to the moment are rewarded. In contrast, NE rewards a solution given the outcome at the end of each episode. This leads to the evolution to find the means to be successful in the end without being explicitly told which actions were desirable. This can cause interesting strategies that diverge from how game-AI usually behave, which is desirable in this context.

Neuroevolution can, like other deep reinforcement learning methods, be **hard to debug**. If some unwanted behavior is observed, it can be hard to pinpoint why such an behavior has been learned due to the black box nature of ANNs [23].

2.6.8 NEAT neural network architectures

In the NEAT library for python there are two neural net architectures. The first architecture is the regular feedforward neural network, where nodes only send their activated output forward to the next layer in the network. The second architecture, is the continuous-time recurrent neural network or CTRNN as presented in Figure 2.6.5. As the name implies, this type of network allows for recurrent connections where output from a given neuron can travel to a neuron in previous layers of the network. Cyclic connections can also be present where activation can travel to the node that first sent the signal.

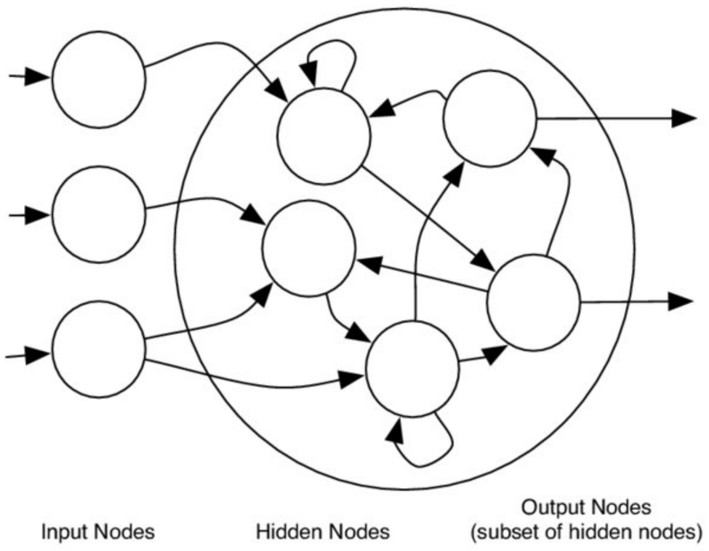


Figure 2.6.5: The CRTNN architecture

In CTRNNs, the neurons are usually of a type known as the *leaky integrator*. Each neuron has a continually updating internal state determined by a differential equation,

$$\tau_i\left(dy_i/dt\right) = -y_i + \sum W_{ij}\sigma\left(g_j\left(y_j - b_j\right)\right) + I_i \tag{2.4}$$

"the τ_i is the time constant, g_j is the gain and b_j is the bias of the neuron i, I_i, is any external input for neuron i and W_{ij} is the weight of the connection between neuron i and neuron j. σ is a non-linear transfer function which in our case is *tanh*." [24].

CTRNNs enable bidirectional network connections, including self-connections, by incorporating recurrency. This interplay between recurrency and internal state generates intricate patterns of activity within the system and produces a memory-like response to its environment [25].

The internal state of each neuron can essentially act as a state that keeps information from observation to observation. This leads to the output from each genome not being decided by only the input from the previous nodes, but also information from earlier inputs.

The internal state property of recurrent neural networks has been shown to consistently outperform feed forward NNs in certain partly observable domains. Where the agent does not observe the entire environment at once like in a car racing task [26], recurrent networks were able to learn much faster than regular feed forward NNs.

2.6.9 NEAT algorithm configuration

The customization of the NEAT algorithm is first and foremost done by implementing a configuration file passed to the methods of NEAT. This configuration file contains a substantial amount of hyper parameters, resulting in it often being used in its default state.

Faitas [27] explored the effect of different configuration parameters on training performance using NEAT. The task of which this evolution was applied, was to evolve the complex control of legged robots for the ability to walk and run among other abilities. The experiment was of a comparable complexity to ours, making the results useful. When the probability of adding nodes and connections were set to 0.8 (default 0.2) and probability of removal was set to 0.6 (default 0.2) in the experiment, it showed a 13.3% improvement in median fitness over the default parameters using the tanh activation function. Compared to the default parameters with the default sigmoid activation function, there was a 77.0% improvement in median fitness.

For the action space in the dodgeball environment presented later in Section 4.2, the output values from the network is intended to be normalized around zero with the tanh activation function. Every action ranges in value from -1 to 1.

2.7 fMRI

This section provides an overview of Magnetic Resonance Imaging (MRI) and its use in medical imaging, along with the more specific application of functional Magnetic Resonance Imaging (fMRI) in investigating brain activity. The workings of MRI, its components and the process of generating images using MRI will be explained in this section. The section will also discuss some of the difficulties involved in performing fMRI scans with substantial participant input, and the solutions that can be used to overcome these issues.

2.7.1 Magnetic resonance imaging

Magnetic resonance imaging (MRI) is a widely-used medical imaging technique that produces detailed images of the body's internal structures, including soft tissues such as the brain, muscles, and other organs. It is commonly used to diagnose and monitor conditions affecting these soft tissues.

MRI machines consist of a large cylindrical tube, known as the bore, within which a patient lies during the scan[28]. The tube houses the parts of the MRI machine responsible for generating and receiving the MR signals. The primary component is a large and powerful magnet that creates a strong magnetic field around the patient.

Gradient coils within the MRI machine are responsible for selecting the specific X, Y, and Z coordinates within the body to be imaged. These coils create smaller magnetic field gradients, which, when combined with the RF pulses, allow for precise localization signal from the hydrogen nuclei.

MRI takes advantage of the high water content in the human body, which contains hydrogen nuclei. Hydrogen is particularly suitable for MRI because it has a large magnetic moment and is abundant in the body due to the high water content, providing a strong signal. The spin property of hydrogen, a fundamental aspect of quantum mechanics, is a form of intrinsic angular momentum. This spin contributes to the hydrogen proton's overall magnetic moment, which is a measure of its tendency to align with a magnetic field. Thus, a proton's spin can be aligned in one of two ways in relation to an external magnetic field: either with it, often referred to as the 'spin-up' state, or against it, the 'spin-down' state, with the latter being of higher energy [29].

When a patient is placed inside the MRI machine, the external magnetic field aligns the spins of the hydrogen protons predominantly into the lower energy, 'spin-up' state, creating a net magnetization. This is the equilibrium state. The application of a specific radio frequency (RF) pulse can then flip these spins from the lower-energy 'spin-up' state to the higher-energy 'spin-down' state, disrupting this equilibrium [28, 30].

Once the RF pulse is turned off, the hydrogen nuclei begin to relax and return to their equilibrium state, the 'spin-up' state. As they realign with the external magnetic field, they emit energy that can be detected and translated into the detailed images for which MRI is known. This process is referred to as T1 or longitudinal relaxation. The rapid realignment of spins and the resulting emission of energy form the cornerstone of MR signal generation and, by extension, the utility of MRI in medical imaging. Simultaneously, another process called T2 or transverse relaxation occurs, where the spins of the protons in the transverse plane start to dephase or lose their alignment with each other due to interactions with their local environments [30]. This results in a decay of the transverse magnetization and the signal detected by the MRI scanner.

The T2 relaxation time is influenced by factors such as magnetic field inhomogeneities, interactions between neighboring spins, and tissue properties, making it highly sensitive to the local micro-environment [31]. Different tissues in the body exhibit varying T2 relaxation times, contributing to contrast in MR images.

In fMRI, a closely related concept is the T2* (T2 star) relaxation time. T2* is affected by both T2 relaxation and additional dephasing caused by microscopic magnetic field variations within the tissue, primarily due to changes in deoxyhemoglobin concentration in blood. fMRI exploits the changes in deoxyhemoglobin levels, and thus T2* signal, to detect brain activity [32].

When a brain region becomes active, it requires more oxygen, leading to increased blood flow and decreased deoxyhemoglobin concentration, which disrupts the local magnetic field. This reduced disruption results in an increased T2* signal, known as Blood Oxygen Level Dependent (BOLD) contrast. By mapping these changes in BOLD contrast over time, fMRI can infer neural activity in different brain regions, facilitating the understanding of how various cognitive tasks or processes involve different brain areas [32].

By applying the RF pulses in different ways, and also recording the signal emitted from the organ of interest in different ways, it is possible to generate many different types of scans with the same MRI machine. Scan types are often divided into two different types of scans. The scan types that depict anatomical features and pathology, are often referred to as structural scans. Physiological scans depict processes such as diffusion, blood flow and other physiological processes in organs are often referred to as functional scans. In this project we acquired both a structural scan, an anatomical scan of the brain, and a functional scan (fMRI scan). The fMRI scans of brain activity during video-game playing investigates the differences in brain activity related to playing against the different agents.

The goal of a functional magnetic resonance imaging (fMRI) scan is to visualize and study brain activity of the subject [32]. fMRI is often used in preparation for a brain surgery, for example planning the best approach for removing a brain tumor by analyzing which brain regions are active during certain stimuli. An additional use case for fMRI suitable for this project is observing and analyzing brain activity during certain stimuli during task performance. The tasks often involve the patient being presented with tasks on a screen and solving them using controller inputs.

fMRI is used to uncover the neuronal correlates of a particular task by comparing brain activity during that task to brain activity during another task or rest. By combining fMRI scans from several participant it is possible to generate statistical parametric maps displaying brain activity across the participants.

2.7.2 Magnetom terra

Magnetom Terra is a 7 Tesla MRI scanner manufactured by Siemens Healthineers. One such scanner is located at St. Olavs Hospital and is currently being used for research in medical advancements. This is one of the most advanced and powerful MRI systems available today. It is a large machine with a magnetic field of 7 T

(One Tesla, abbreviated as T, is equal to 10,000 Gauss and is approximately 20,000 times the magnetic field of Earth [29]).

The scanner at St. Olavs Hospital is situated under the ground, surrounded by a conductive enclosure known as a Faraday cage. This serves a dual function. First, it shields the scanner from external RF pulses that can interfere with the coil responsible for receiving signals in the scanner. Second, it safeguards the surrounding environment from RF emissions generated by the scanner. The facility comprises distinct areas: the previously mentioned scanner room and the operational room. The operational room is where the technical equipment and researchers are located. Additionally, there is a technical room connected to the scanner room, where all the computers controlling the scanner is located. All the equipment from the scanner room is connected to the operational room and technical room, which are outside of the Faraday cage where technical equipment is safe from the magnetic field.

As previously explained in section 2.7.1, the MRI scanner needs coils to target the imaging to a specific body region. The different coils have distinct designs and functionalities, which are tailored to specific applications, anatomical regions, or imaging needs. In this project, a specialized coil called a head coil was applied. The coil that was used for all scans during this project was a 8Tx/32Rx head coil specialized for the 7T scanner.

2.8 Eye tracking

The eye is a complex organ that plays a big role during game play, particularly reaction based games. Depending on which game being played, the eye and its ability to observe and react to certain elements in the game may be crucial to winning. Analyzing eye movements creates great value to developers and designers of both games and regular user interfaces. By analyzing what elements the subject observes initially can easily disclose confusing elements, elements that take too much attention relative to their importance, and the ease of navigating for the user.

There are two main types of eye movement, fixation and saccades. Saccades are sudden jumps of the eye to a new point in the environment and fixation is when the eye is fixed (resting) at a given point in a gaze. Fixation happens in between saccades. There is also one kind of eye movement called smooth pursuit which is "The eye movement that takes place when looking at an object in motion and following it." (EyeWare [33]). During such eye movement the eye can still take inn visual information in contrast to saccades where little to no information is gathered.

Eye tracking, it allows for the study of visual attention. The interest of the subject can be analyzed by observing both the conscious or unconscious actions of looking at specific objects. How objects in peripheral vision can also be detected.

Most eye trackers works by following the eye position and movements by using near-infrared or infrared light to illuminate the pupil. By analyzing the reflected infrared light, one can deduce which direction the pupil is pointed at, the pupil diameter, pupil center and the eye rotation.

2.8.1 Evaluating game experience using eye tracking

Examining the complex nature of game experience with eye tracking has been researched, with attempts to link game experiences with certain eye behavior.

Researchers have investigated the relationship between the player's eye movements and the immersive quality of the game. Jennett et al [34] observed that during immersive game sessions, eye movement tends to decrease over time, indicating a growing focus on the game task. Comparatively, during non-immersive conditions, the eye movement increased as the player's attention diverted from the task, leading to more visual exploration.

These notions of focus and exploration are echoed in the study by Mauri et al. [35]. They found that subjects set to do various tasks on Facebook were observed to have higher mean pupil dilation during states of overload than flow states. During relaxed states, pupil dilation has been found to be of a higher mean. Mauri et al. suggests that "participants were not necessarily under-aroused or bored, but may have been open to environmental input (i.e., in an exploration mode), hence their higher baseline pupil diameter".

For this project, we have access to an eye tracking accessory for the 7T fMRI scanner at St.Olavs. To utilize this, the WebLink [1] software can used. This software allows us to record the gaze and pupil size while people interact with the environment. The weblink software can also stream gaze data to any software capable of network communication. This way a possibility for game development is incorporating the gaze as a way of maneuvering the game itself.

2.9 Theory of flow

According to Csikszentmihalyi [36], being in a state of flow is characterized as being totally engulfed and focused on the task at hand, and can improve the quality of ones life. According to Mike Oppland it is when something is challenging but yet doable [37].

In 1990, Csikszentmihalyi conducted extensive research into what makes experiences enjoyable. He based his results on long interviews, questionnaires and other data collected over a dozen years from several thousand respondents [38]. Through his studies he found that the optimal experience, or as he calls it, flow, is the same for people across the globe. Flow is an experience 'so gratifying that people are willing to do it for its own sake, with little concern for what they will get out of it, even when it is difficult or dangerous' [36]. The experience of flow consists of eight elements that Csikszentmihalyi described as follows:

1. A task that can be completed and is balanced between challenge and skill.
2. The ability to concentrate on the task.
3. Clarity of goals and immediate feedback
4. The experience is intrinsically rewarding
5. The ability to exercise a sense of control over actions.

[1]https://www.sr-research.com/weblink/

6. A deep but effortless involvement that removes awareness of the frustrations of everyday life.
7. Concern for self disappears, but sense of self emerges stronger afterwards.
8. The sense of the duration of time is altered.

Combining these elements causes a sense of deep enjoyment that is so rewarding that people feel that expending a lot of energy is worthwhile simply because you can enjoy it [36]. According to Huskey et al. [39]. 'A balance between task difficulty and individual ability results in the highest levels of intrinsic reward'. Moreover, high levels of intrinsic reward corresponded to increased task-related attentional engagement. The flow model has also been adapted to a gameflow model, mapping the elements of flow to aspects of video games. This model is briefly presented in Appendix Section I.

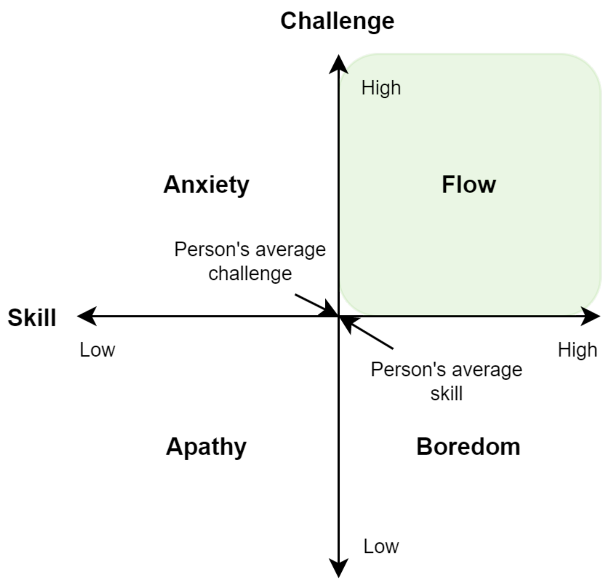


Figure 2.9.1: *The four-channel flow model. Adapted from Pace [40] .*

CHAPTER
THREE

RELATED WORK

There have been several studies on the use of genetic algorithms and neuroevolution to evolve AI applied to videogames, as well as studies that have used fMRI scanning to measure brain activity in relation to video games. In this chapter, we review these studies, highlighting key findings and limitations. This will provide a foundation for understanding the current state of the field and the specific contributions of this thesis.

3.1 Current research state of NEAT

NEAT or neuroevolution has been applied to deterministic environments like chess, checkers, go and pacman with varying success. For the popular chinese board game go, NEAT or HyperNEAT did not manage to beat the state of the art being MCTS, which is a popular search-tree based reinforcement learning algorithm. [15]. This was also the case for other board games such as chess[14]. However, for non-deterministic environments where state/action evaluation is not feasible, neuroevolution gains a clear advantage [23]. Neuroevolution has seen research in both single- and multiplayer games. Due to the nature of this project, the main focus will be on research applying neuroevolution to one versus one games.

Unfortunately, the application of neuroevolution to nondeterministic player versus agent games is confined to a niche domain, resulting in a limited body of available research. To address this research gap, less related research regarding singleplayer games, HyperNEAT and real-time NEAT (rt-NEAT) is included. In rt-NEAT, evolution happens in real-time by playing against human players. This optimizes the agents against what they would ultimately be playing against post-training; the human player. However, it requires a human player to train the agent by playing in real-time. rt-NEAT also benefits games with multiple opponents where individuals rather than whole populations can be swapped out, creating a smoother transition towards complexity.

3.1.1 NEAT vs HyperNEAT

A popular extension to NEAT is HyperNEAT, which is widely used for certain problem domains. HyperNEAT has been shown to provide improved performance over NEAT in less complex tasks where geometric symmetries are present. However, HyperNEAT performs worse in nondeterministic environments where the challenge and correct action may abruptly change from state to state. It was also observed that HyperNEAT took significantly longer than NEAT to reach the same number of generations, making it infeasible for tasks of greater complexity [41].

HyperNEAT utilizes a separate compositional pattern-producing network (CPPN) for evolving weights of a fixed topology network. The CPPN is able to capture geometric regularities in the task better than standard NEAT. A good solution may be reached faster than NEAT, while NEAT spends time continually developing the network topology. This feature empowers NEAT to discover optimal solutions while employing compact and appropriately sized network architectures. The most prevalent NEAT implementation [42], transcends any HyperNEAT implementation in popularity and developmental engagement, while supported by extensive documentation.

Bjerke [43] employed both NEAT and HyperNEAT in his work. The environment consisted of two fencers of with the end goal to defeat each other. However, the fencer agents had to primarily learn how to operate their limbs. He encountered some difficulties when the environment became more complex. Evolution with HyperNEAT performed quite well with a simpler environment, but more advanced environments resulted in poor performance compared to NEAT with the CTRNN network. As mentioned in his future work, the results could have been better by testing different configurations of HyperNEAT (and NEAT) to facilitate stable and efficient evolution training.

Hausknecht et al. [44] compared the performance of different algorithms including NEAT and HyperNEAT in the case of general Atari game playing. In their experiments, NEAT significantly outperformed the other algorithms explored while being statistically similar to HyperNEAT.

3.1.2 Application of rt-NEAT in games

rt-NEAT is essentially a extension to NEAT, but designed to evolve agents in real time. rt-NEAT tries to follow the same principles as NEAT. Instead of replacing the entire population at once, the agent with the lowest fitness is replaced with new descendants from selection and crossover every couple of game ticks. Miikkulainen applied rt-NEAT [45], allowing players to interact with evolving agents and tailor them to improve their combat skills. The project resulted feedback like the game feeling engrossing and entertaining, where battles were exciting and diverse.

Mandujano et al. [46] applied both NEAT and rt-NEAT to evolve a large number of agents in a top-down game environment. During their experiments with rt-NEAT they saw no significant increase in fitness, besides when evolving against a randomly moving hard-coded population. However, the experiments using NEAT outperformed rt-NEAT in terms of efficiency and fitness. The default NEAT configuration was used and the fitness function rewarded for hits, targeting enemies and escaping a enemy targeting, while punishing for being eliminated and being locked on by a enemy. The results could point towards a NEAT configuration not suitable to the challenge or a fitness function far too specific introducing unwanted behavior.

3.1.3 Application of NEAT in games

Aiming to maintain a general fitness function allows NEAT to obtain a multitude of different strategies after sufficient training times. Traish et al. [47] generated agents with complex adaptive behavior after forcing the algorithm away from stagnating at a single solution. This was done by selecting specialized genomes to beat certain opponents. 200 generations was sufficient to develop an agent that exhibited different strategies in response to different opponents.

NEAT has also been applied to Mega Man II, a 2D game akin to Super Mario, by Ishikawa et al. [48]. The objective of the game is to survive and vanquish enemies and bosses, some of which exhibit non-determinstic behavior regarding their performed actions. Their agents relied on a subset of the 20 sensors available in the game and managed to beat all the eight bosses in the game after 150 generations of evolution.

Mobile games have also seen the application of NEAT, with the Flappy Bird game as the most prominent example. Cordeiro et al. [49] applied NEAT to find the minimal neural net size to play Flappy Bird indefinitely. The algorithm converged after 20 generations to a minimal solution which resembled a perceptron with three inputs and one output node.

3.1.4 Evolving game AI for player vs AI games through evolutionary algorithms

Martinez-Arellano et al. [50] carried out a study involving the evaluation of AI characters. The agents were developed through genetic programming, by having them interact with human players in a street fighter-style fighting game called M.U.G.E.N. The human players rated the agents created by genetic programming higher in terms of rated difficulty and engagement. They found a significant correlation between difficulty and satisfaction, which suggests that humans generally prefer more challenging AI opponents. When human players encountered characters of roughly equal strength, their perception seemed to differ, suggesting the influence of factors beyond the mere strength of the opponent. Before the experiment, participants were familiarized with the game mechanics without exposure to any specific AI. The team also compared their AI agents with the standard AI that was issued with the fighting game. The result was that all AI agents developed via the genetic algorithm outperformed the hand-coded AI in terms of wins. During the training process, they observed a sharp rise in win rate within the first 8 generations, which later slowed down, indicating more gradual progress in win rate.

Further, Wittkamp et al. [51] demonstrated how genetic algorithms can foster complex and distinctive team behaviors with distinctive role development in Pacman. The behaviors were deemed to be superior compared to the original behavior of the ghosts in the game in terms of score.

Moving towards shooter games, Reeder et al. [52] applied a modular NEAT approach to evolving game agents in a 2D spaceship environment. The task goal was to kill opponent ships in the game. These opponents evolved using sensors as observation input, with two neural networks: one for shooting and one for movement. Their modular neural network NEAT approach was based on real-time NEAT, due to the jarring experience of all enemies changing behavior at once. Their agents were trained for 200 generations, where fitness saw a dramatic increase in the first 20 generations, then progression slowed significantly with fluctuations. .

Another shooter game game that has been used in AI development competitions is Unreal Tournament 4. Asensio et al. [53] set out to create ANNs within the game using genetic algorithms. Their ANN model demonstrated a considerable improvement in emulating human-like behavior compared to the previous winning model.

Priesterjahn et al. [54] developed human competitive agents in the arena shooter game Quake3 that could dominate the standard Quake3-bot in any difficulty setting. The agents were evolved using an evolutionary algorithm evolving input/output rules for the agents. They emphasized the necessity of subjectively evaluating the agent during evolution to assess the quality of evolved agents, where analyzing fitness was less important.

For research regarding the application of NEAT to fighting games, Kristo et al. [55] applied NEAT to FightingICE, a popular environment geared towards AI research. FightingICE is a one versus one game akin to the Street Fighter games. Their goal was to create an agent capable of challenging the current champion AI, which had been developed using K-means, during an IEEE conference. The agent is given control over seven discrete input keys, with the observations being the distance to the opponent in the x-axis and y-axis, hitpoints, and stamina of both players. The evolved agent did not beat the champion K-means AI, but still performed well against it and the other agents from the conference.

3.2 fMRI and computer gaming

Related research will be elaborated exploring brain activity with fMRI in different contexts. The research regarding brain activity in response to games will be the biggest focus, providing theory on what types of activation can be expected from the participants of this study. The research field of investigating the impact of AI in games on brain activity is seemingly unexplored, leading to the necessity of exploring less related research that can partly cover the methods used for this project. Research exploring neuronal responses in correlation to game events, self-reporting and the state of flow is investigated.

3.2.1 Neuronal correlates of flow: general findings

Research has attempted to link the experience of flow in gameplay to distinct patterns of brain activity. However, there are some variance and common theories and findings amongst research in this field. According to the study by Ferell [56], being in a state of "zone performance" or flow correlated with increases of activation in the cerebellum, the subcortical structures of the putamen and claustrum, and the cortical motor and sensory areas.

The study by Hirao [57] also supports the theory of decreased activation of the prefrontal brain area. He found a clear negative correlation between subjective reporting of high satisfaction and PFC activation.

According to Isak Andersson's literature review [58], "the dorsolateral prefrontal cortex and the putamen are the most frequently reported brain areas showing increased activity during flow". Meanwhile some studies report deactivation in the medial prefrontal cortex.

3.2.2 Dissecting the flow experience

In an article by Klasen et al. [59] they decomposed the different aspects of flow and measured them individually to find which neural correlates associated with the experience of flow. By analyzing the actions taken by the subject in game in addition to the situation, the first five aspects of flow could objectively be estimated.

The study consisted of 13 subjects playing a shooter game, where the level of balance versus skills was measured objectively by the ratio off kills (success) versus deaths (fails). They found a stronger activation in midbrain structures following a direct comparison between success and failure events. The brain regions activated at success were the head of the caudate nucleus, putamen, nucleus accumbens, superior parietal cortex, cerebellum, thalamus, motor and premotor areas. For failure situations, the cuneus showed stronger activation. The level of concentration and focus was based on the subjects response time before action upon seeing a enemy. The focus was measured during three game situations: 1) waiting for a new round to begin, 2) appearances of danger, 3) increased requirement of active engagement.

According to Klasen et al. [59]:

> *Increase of the player's focus was characterized by an increase of activation in the cerebellum and the visual system, in the precuneus and premotor areas as well as by a decrease of activation in bilateral intraparietal sulcus and the orbitofrontal cortex and the rostral part of the ACC.*

The direct feedback states were observed when the participants received immediate feedback if the the action was successful (defeating the opponent). No significant effects were found for this factor. The effect of clear goals during gameplay was performed by comparing game phases without any visible or audible enemy contact lasting longer than 10 seconds, with enemy contact phases. Clear goals were characterized by increased activation in the bilateral intraparietal sulcus and fusiform face area and decreased activity in the dorsal anterior cingulate cortex and precuneus. Participants sense of control was characterized by the influence over game content, for example changing weapons. The easier participants could transform their coices into successful actions, the more sense of control they experienced. It was found that levels of control related to increased activity in the networks of the visual, cerebellar, thalamic, and motor-cortical regions and decreased activity in the bilateral temporal poles, and bilateral angular gyrus.

Katsyri et al. [60] also investigated the effects on the brain in terms of control. Active and vicarious gameplay was compared by analyzing the brain activity. Participants playing a game actively reported to have a elevated flow experience, lower negative affect, higher immersion, and greater spatial presence. They also rated loss events as more unpleasant when playing actively rather than passively watching. Each participant was rewarded or punished monetarily for both active and vicarious play.

During active gameplay, several activation clusters were revealed in the bilateral striatum, midbrain (including ventral tegmenmtal area (VTA) / substantia nigra (SN)), sensimotor cortices (pre- and post-central gyri), and ventral visual stream. The orbitomedial prefrontal cortex exhibited stronger activations when winning compared to losing in both active and vicarious play. There was observed a suppression in midbrain and striatum activation for both win and loss events during active play. However, the striatal suppression particularly in the putamen was greater for loss events.

Additionally, the study revealed that the striatum exhibits activity not only during sensorimotor control for corrective hand movements but also in response to winning or losing. However, the activity related to winning or losing was found not just to be due to hand movement.

3.2.3 Linking subjective game experiences with brain activity

In an article by M. Klasen et al. from 2008 [61], comparisons were made between the actual brain response, the game events derived from analysis, and lastly, the subjective interpretation of the events. They mapped specific game situations to specific activation patterns in the brain. During more demanding game situations and situations that require great concentration and focus, the visual system and the cerebellum had increasing activation clusters.

Subjects demonstrating greater displeasure exhibited stronger activations in the right precentral gyrus (PCG). Additionally, they found a correlation between rated displeasure and focus during game play. Which can be explained by participants that react to focus periods by exhibiting readiness for action, dig not enjoy the game possibly due to experiencing tense- and tightness. Responses like displeasure can be assessed through parameters like heart rate, activation in precentral cortex areas, and eye movement behavior.

In a study by Ulrich et al. [62] from 2014, they observed distinct functional association when subjects reported experiencing a state of flow:

> *"increase of neural activity in the putamen possibly reflecting increased outcome probability, and in the left inferior frontal gyrus which might reflect a deeper sense of cognitive control. Reductions in neural activity were observed in the medial prefrontal cortex, suggesting a decrease in self-referential processing that has previously been shown to associate with negative affectivity. The decrease in rCBF was also evident in the amygdala, which could mirror a decrease in arousal that contributes to or refutes positive emotional experiences during flow. "*

They also found a correlation between participants subjective ratings of flow and changes in neural activity in the inferior frontal gyrus and amygdala. The medial prefrontal cortex had a slight correlation. It was concluded that the neural activity for these brain regions embody mental processes that align with the characteristic features of flow:

- Putamen: Coding of increased outcome probability
- Inferior frontal gyrus: deeper sense of cognitive control
- Medial prefrontal gyrus: decreased self-referential processing
- Amygdala: decreased negative arousal

Saito et al. [63] explored brain activations following gameplay of games requiring logical thinking versus real-time reaction. For the real-time reaction games Tetris and Space Invader, prefrontal cortex, premotor cortex, parietal cortex, and visual association cortex were found to have increased activation during play in comparison to the games requiring logical thinking. These brain regions are known to be involved in planning and preparing for actions. Furthermore, the task difficulty was found to influence the degree of activation in the dorsal prefrontal and premotor cortices, but not in the parietal cortex. This confirms the established theory that the prefrontal and premotor cortices form a circuit involved in task planning.

Ju et al. [64] uncovered brain networks linked to different experiences. They investigated certain subjective gaming experiences like immersion, flow, and challenge. Immersion and flow were found to be positively correlated with parts of the dorsal and ventral visual streams involved in processing visual information. The dorsal stream is often associated with spatial awareness and guidance of actions, and the ventral stream is associated with object recognition and form representation. The insula also exhibited increased activation, which is a region of the cerebral cortex that plays a role in diverse functions usually linked to emotion or the regulation of the body's homeostasis such as perception, motor control, self-awareness, cognitive functioning, and interpersonal experience.

An article by Katsyri et al. [65] investigated the differences in brain activity when winning against a human opponent versus winning against a computer opponent in a game called EZFlag. The participants were only told they were playing against a computer, where in reality all of the 14 subjects played against humans. Through a Region of Interest (ROI) analysis, they found a general stronger activation when winning in the ventral and dorsal striatum, as well as in the prefrontal cortex (vmPFC). There was found a significant association between participant's perceived pleasure and elevated activity in the nucleus accumbens. Additionally, they found stronger responses in the vmPFC and dorsal striatum when winning against humans rather than a computer opponent. When participants played against a human, the signal changes were greater in these regions compared to when playing against a computer.

A Wilcoxon signed-rank test was performed and revealed no significant differences in terms of performance or learning within the games between games played against humans and computer. The performance during games was determined by the participant scores and learning was measured by comparing performance difference between early and late game phases.

3.2.4 Synchronization theory

A theory on the neuronal factors underlying the experience of flow is the synchronization theory by Weber et al. [66]. They proposed a theory where synchronized cerebral neuronal networks are responsible for the flow experience and the synchronized oscillation of neurons and resonance with other groups of neurons does not necessarily have one specific function but serves many purposes. Flow is described as "a state of holistic consciousness that is more than its parts". This model takes inspiration from Posner's attentional model [67], which was introduced in 1987. This model suggest that the flow state arises from the synchronization of focused attention networks (alertness and visual orienting networks and the striatal reward networks, whose activation would allow the pleasureable component of flow state to rise. The Synchronization Theory of Flow predicts strong frontal cerebral brain activity.

3.2.5 Large scale network approach

Van der Linden et al. [68] suggest flow is a constant interaction between a few cerebral networks:

1. Default mode network
2. Central executive network
3. Salience network

The Large Scale Network model proposes that flow arises from the complex interplay within key brain networks. These networks include the Default Mode

Network, associated with self-referential thoughts and mind-wandering; the central executive network, implicated in focused attention and engagement; and the salience network, responsible for managing neural resources and maintaining balance between networks [69, 70, 71].

According to this theory the the harmonious interaction between these key brain networks is necessary to achieve the flow state. It was suggested that activity in the Default Mode Network would be low and the central network would show high activation during flow. This was supported by measurements using fMRI and EEG.

3.2.6 Neural research conclusion

Common findings amongst the studies is the involvement of the striatum, a region of the brain that is involved in reward processing, cognition and motor control. Katsyri et al [60] found that during gameplay, the striatum exhibited activity during both sensorimotor control over corrective hand movements and winning or losing. Similarly, Ferell [56] found that being in a state of "zone performance" or flow correlated with increases of activation in the cerebellum, the subcortical structures of the striatum, and the cortical motor and sensory areas. Zone-state performance and normal performance were hypnotically recalled,

However, there are also some differences in the findings. For example, Ulrich et al. [72] found an increase in putamen activity during flow, which they interpreted as reflecting an increase in successful outcome probability. In contrast, Saito et al. [63] found increased activity in the prefrontal cortex, premotor cortex, parietal cortex, and visual association cortex during real-time reaction games, suggesting that these regions may be important for processing the fast-paced, demanding nature of those games.

Another interesting difference is in the role of the prefrontal cortex. While some studies report increased activity in the dorsolateral prefrontal cortex during flow, other studies report a decrease in activity in the medial prefrontal cortex. Hirao [57] found a clear negative correlation between subjective reporting of high satisfaction and PFC activation, while Ulrich et al [72] found a decrease in medial prefrontal cortex activity during flow. This suggests that different types of flow experiences may be associated with different patterns of prefrontal cortex activity.

The theories regarding holistic activations and interplay between connected brain networks highlight the complex interplay between neural activity and subjective experiences during game play. While certain regions of the brain, such as the striatum and putamen, appear to be consistently involved in processing rewards and motor control, the specific patterns of neural activity may vary depending on the type of game, the level of difficulty, and the individual's subjective experience.

3.2.7 Game relevant brain regions to be investigated with region of interest analysis

Previous studies [65, 59, 62] and general findings regarding important game relevant brain activation revealed that the amygala and nucleus accumbens were interesting candidates for a region of interest (ROI) analysis. The amygdala is associated with emotion processing, consolidation of emotional memories and decision making, particularly the emotions and motivations that are related to survival [73]. Emotion processing includes shaping the emotional experience during play, whether it is the excitement of victory, tension of a close match or the frustration of a loss. A stronger activation in the amygdala could suggest an increased in emotional responses, which could make a game feel more thrilling, tense or exciting. However, negative emotions like frustration or anxiety, can also be strengthened if the if the game is too challenging for the player. An increase of activity in the amygdala can also point to an increased probability that emotionally charged events and other experiences associated with the game are remembered.

Both the amygdala and nucleus accumbens are associated with decision-making. However, the amygdala is most active in decision-making when emotional factors are involved. The nucleus accumbens is particularly involved when the decision making includes evaluating potential rewards [74]. Together, strong activation in these regions can point towards players being more inclined to take risks in the game. This could manifest as bold or aggressive strategies in the game. Additionally, activity in the nucleus accumbens is tied to motivation, increasing the desire to work for rewards. The brain area has been suggested as the "pleasure center", involved in reinforcement learning. Release of dopamine in the nucleus accumbens have been tied to receiving rewards, and is central to goal-oriented behavior. Activity in both the amygdala and nucleus accumbens, can suggest a enhanced game experience where players are more engaged and motivated by playing, while experiencing more intense emotions which are remembered more strongly afterwards.

CHAPTER
FOUR

METHODS

In this chapter, we describe the methodology used in the study in detail. This includes the design of the videogame, the development of the AI using neuroevolution with the NEAT algorithm, and the procedures for conducting the fMRI measurements. We also discuss the data collection and the data analysis methods. By providing a detailed description of the methods used in the study, we ensure that the results can be replicated and the validity of the study can be evaluated.

4.1 Unity and the selection of environment

To achieve the aims of this project, it was important to select an environment able to tie all the technologies together. As such we knew that a well used technology would help, as there would be extensive documentation, libraries and packages that could save many hours in a study that already was quite short on time. This is why Unity was selected to be used as the engine for our game. Unity is one of the most popular game engines for videogame development. Much if this popularity is due to Unity being easy to use, but a highly contributing factor is an extensive asset store.

Game development is notoriously time consuming. which deemed developing the game environment from scratch unrealistic within the time frame of this project. To remedy this, game templates were explored to serve as a base outline of the game. Many templates were considered, but they all had difficulties as the requirements for our study were specific. We needed a game that was easy to interpret as we wanted players to spend minimal time getting accustomed to the game, and avoid creating unnecessary complications. We also wanted an environment where players would be facing a single opposing agent. If several agents were playing in the same environment as the player, it would be harder for the player to interpret the opposing agents moves. This kind of chaotic environment could have been more interesting to play in, but would be inexpedient for the study purpose. Finally the game needed custom controls that would suit an fMRI study. Given the limited equipment able to be used in an fMRI machine, and limits to the equipment available at St. Olavs Hospital, the game could not exceed a certain level of complexity. This selection of controls will be explained later in Section 4.3.1.

The selected environment is a version of the machine learning agents(ml-agents) toolkit made for Unity. The ml-agents toolkit is a plugin made for researchers and developers, allowing training of agents using different techniques of artificial intelligence within the Unity game engine. It is an extensive toolkit that contains several different game modes and environments for the agents to train and act upon. Many of these environments put agents up against an environment, where the task at hand is to learn to walk or solve a given task to proceed, while other environments place agents against each other in simulated sports. Some of these environments simulate tennis or football, which could be great environments for our study. However, in July 2021, the ml-agents team presented a new environment for agents to compete in a dodgeball environment[75]. This environment is fairly developed compared to the previously mentioned environments.

4.2 Dodgeball videogame environment

The dodgeball environment is an open-source collection of premade assets that are put together to create two main game modes, elimination and capture the flag. The elimination game mode, which we use, puts players against each other in a 4 versus 4 match, where the goal of the game is to run around in a small arena and pick up balls and throw them at the enemy team. If you hit an enemy enough times, they are eliminated, and if the whole team is eliminated, the game ends with the remaining team as the winner.

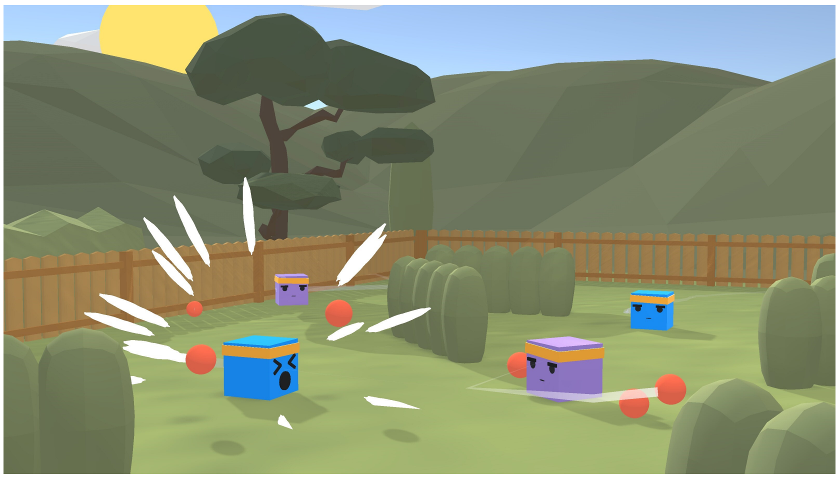


Figure 4.2.1: ML-Agents Dodgeball environment[75]

The agents in this game have some individual properties, such as how many hits they can receive before being eliminated, also known as hitpoints (HP), and how many balls they can hold. This data is stored individually on the agent, while data such as players remaining on the team is stored in a centralized game controller. The agents also hold data about their teammates, such as their position and their HP [75]. The agents are taught how to play the game through reinforcement learning, which suggest several properties. In Section 2.3, we discussed the functioning of reinforcement learning agents, including their ability to receive rewards and punishments based on their actions in the environment. Similarly, these agents require the capability to observe their surrounding environment in order to make informed decisions.

4.2.1 Observations

In the dodgeball environment, the agents mainly observe the environment through raycasts. Raycasts are projections of rays into the scene. Each of the raycast sensors has a set of tags that the agents are continuously looking for, shown in the Table 4.2.1. Should the ray collide with the game object it is tagged to detect, it returns a boolean value and a float value between 0 and 1. The boolean value returns true if the raycast hits a tag, while the latter value is an indication of the distance between the agent and the point where the individual raycast line was obstructed by an object.

BallRaycast	*AgentRaycast*	*WallRaycast*	*BackRaycast*
dodgeBallActive	purpleAgent	wall	wall
dodgeBallPickup	blueAgent	bush	bush
	purpleAgentFront		
	blueAgentFront		

Table 4.2.1: The sensors that agents perceive the environment through, and the tags they can perceive.

The BallRaycast tags look for balls that are available for pickup on the map, and dodgeballs that are being thrown, in case it can dodge the ball. The AgentRaycast now only looks for the enemy player cube, and the WallRaycast detects walls and bushes in front of the agent. The number of ray casts varies, but most of the sensors cast rays in an arc in front and to the sides of the agent. The BackRaycast differs from the rest, only casting rays behind the agent instead of in the front.

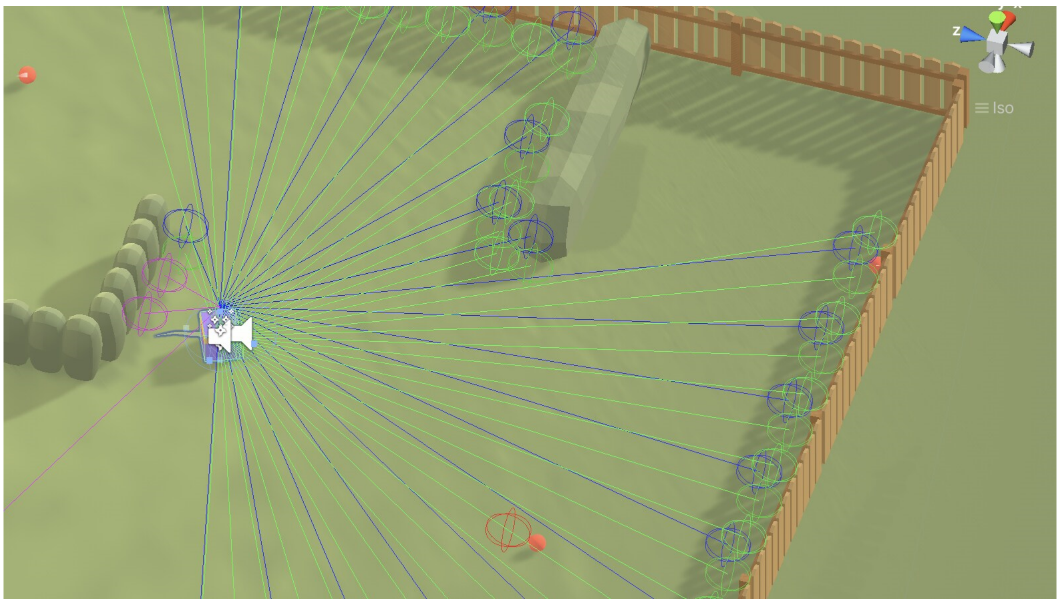


Figure 4.2.2: The raycasts used to discern tags. [75] To see the individual raycasts clearer see Figure C.0.1 in the Appendix

4.3 Player versus agent dodgeball environment

Observing individual NPC behavior is difficult in a chaotic environment. Since this project aims to evaluate the emotional responses provided by play against a singular NPC, and not an AI director or a swarm, we decided to cut down the amount of agents. Agents on each team were reduced from 4 to 1, converting to a player versus agent environment. This has both advantages and disadvantages discussed further in this chapter. The main advantage is that players are allowed to direct their attention to a single agent and observe its behavior more thoroughly. Events and observations are also easier to correlate to brain behavior, since the environment is much less noisy with the reduction of agents.

Win conditions were slightly altered to account for this change. Instead, the players receive *hitpoints*, or HP, which are a common way to describe health in video games. Each player has 3 HP, which means they can get hit 3 times before they lose the game. This is indicated to the player via the headband on their cube. If the headband is green, the player has 3 lives left, yellow for 2 lives left and finally red for 1 life left as seen in Figure 4.3.1. This way the player has the same access to this information as the agent as, without having to keep count. Since the agent has a state that keeps track of hitpoints, this is a natural way for a human to receive the same information. Additionally, this removes the need for a heads-up display (HUD), showing the current health of the player. The same technique is used for the amount of balls the player is carrying being displayed by holding the balls on the agents back, providing both player and agent with the same information.

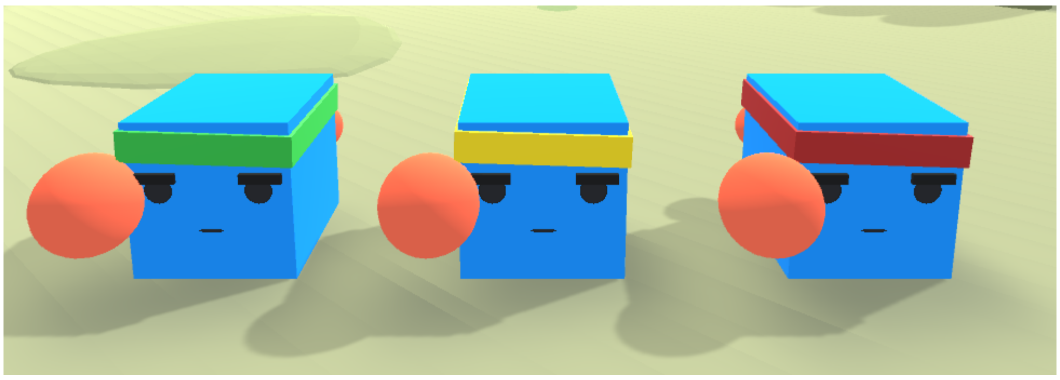


Figure 4.3.1: The headband colors displayed on the player controlled agent.

We wanted the player and the agent to be on equal terms when playing the game, to alleviate the feeling of unfairness from the player when measuring their brain activity. As the agent was unable to see beyond bushes, it was deemed appropriate to prevent the player from spotting the agent over bushes, thus eliminating any potential for gaining an unfair advantage. To achieve this balance, the player camera was lowered, and the height of the bushes was increased slightly, effectively limiting the player's visibility. This makes the player camera angle and the raycast sensors provide the same information for player and agent.

4.3.1 Actions and controls

The means of interactivity in the dodgeball environment is through a set of actions consisting of five inputs, where the first three are continuous and last two are discrete.

1. Vertical movement (forward and backwards movement, continuous value from -1 to 1)
2. Horizontal movement (left and right, continuous value from -1 to 1)
3. Rotation (continuous value from -1 to 1)
4. Shoot (discrete boolean)
5. Dodge (discrete boolean)

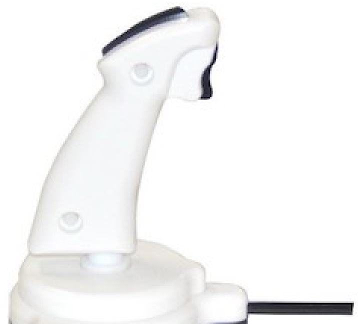
One of the core problems of playing a game inside an fMRI machine is figuring out a way for the player to input these actions into the game. Today there are many different controllers that allow this type of input in different ways. This section will explain the choices made in regards to the control scheme used in the game.

The dodgeball environment supports most of these controllers natively, but was made with the intention of controlling the game with a mouse and keyboard. Bringing a mouse and keyboard into the fMRI machine is a daunting task, both due to ferromagnetic components and physical space. The bore size (size of the cylinder hole where subjects are placed inside the fMRI machine) of the magnetom terra is 60cm in diameter [76, 28], giving little freedom for movements such as moving a mouse. Ideally subjects could control the game with something that keeps their hands stationary, such as a console gaming controller. This is a very standardized control scheme, being used for most of the popular gaming consoles. Sadly, one cannot bring a normal gaming controller into the fMRI machine due to the intense magnetic field emitted by the machine.

There are fMRI compatible console gaming controllers that contain no ferromagnetic components, and are considered safe for use in an fMRI machine. However, no such controller is located at the 7T office at St. Olavs Hospital. Being a very niche type of equipment, these controllers are also highly expensive[77]. Because of this, we decided to look into the controllers available to see if they could contribute to a satisfactory control scheme.

In the 7T office, we had access to a modest range of controllers that are considered MR safe. One of them was a four-button controller, which was connected to the operational room via a fiber-optic cable. Fiberoptic cables are commonly used in fMRI experiments as they transmit data through reflections of light rather than electrical currents. There was also a pair of grip controllers available, also known as response grips. These are a paired set of controllers that allow two inputs on each hand, one thumb and one index finger input. Lastly, we had the Tethyx joystick, a two-button controller that utilizes a dual-axis shaft encoder to manage two-dimensional grid positioning. Notably, like the four-button controller, this Tethyx joystick also uses a fiber-optic cable for data transmission.

Given our controller options, we concluded that the most effective control setup would involve using the joystick for player navigation, complemented by the four-button controller for actions like strafing or turning left and right. The joystick's two buttons could be designated for actions such as "shoot" or "dodge". Using the joystick for directional inputs is crucial, as none of the other controllers at our disposal offer the same level of navigation capability However, a single port for fiber-optic cables prevents these two controllers from being used simultaneously. Therefore, our final control scheme pairs the Tethyx joystick with one of the grip controllers.


(a) The Tethyx joystick[78]

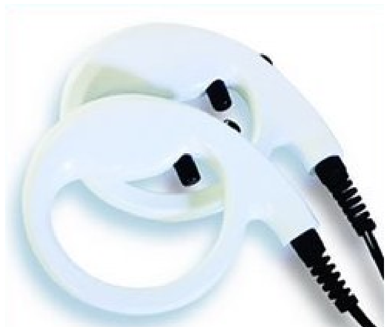

(b) The grip controllers[79]

Figure 4.3.2: The controls used in this study

A final decision needed to be made regarding whether strafing(moving left or right) or turning(turning left or right) should be bound to the grip controllers. Through play testing on voluntary participants, it was found that eight out of ten participants preferred strafing with the grip controller over turning. The final control scheme selected for the game is presented in Table 4.3.1.

Controller	Input	Action
Tethyx	Left/Right	Turn
Tethyx	Forward/Back	Move
Tethyx	Index Trigger	Shoot
Tethyx	Thumb Trigger	Dodge
Grip	Index Trigger	Strafe Right
Grip	Thumb Trigger	Strafe Left

Table 4.3.1: Control Scheme

4.3.2 Scenes

To achieve the different purposes of the project, several scenes were created. A scene is a self-contained environment that contains all the necessary elements required for a part of the game to be played. The elimination game mode initially existed as a preexisting scene, which was subsequently modified to serve distinct purposes within the project. Specifically, individual scenes were created for each of the agents that were developed for gameplay, allowing for the loading of different agents for each play session. Additionally, a baseline scene was created with a stationary agent. Finally, many scenes were also created to train the NEAT agent.

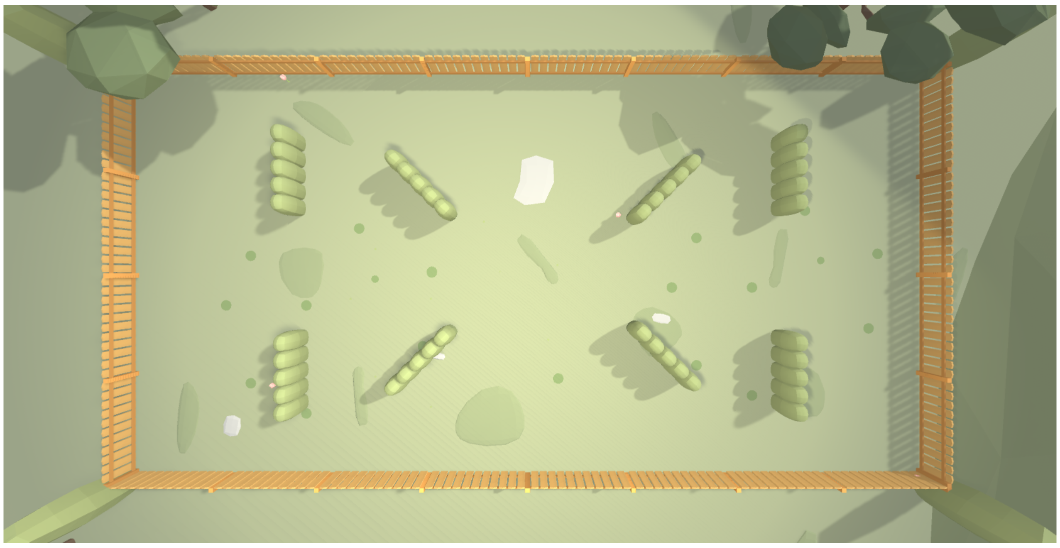


Figure 4.3.3: The stage used in the elimination game mode. Player and agent spawn on opposite sides of the stage.

The main scene used for elimination mode was unaltered from the original map created in the ml-agents toolkit. It features a rectangular shape with eight hedges placed in a symmetrical pattern to create cover. The hedges are made up of five individual bushes, which each have jiggle physics that make them interactable. This feature allows players to pass through the hedges and even send balls through them if they are precise. The stage was kept like this to retain the interesting and complex scenarios that arise when hiding for cover.

4.4 Game Agents

Several agents were developed for specific purposes in this project. This section will elaborate the different agents' purpose and explain choices made when developing them.

The player controlled agent kept the original blue color from the original dodgeball environment. It is a suitable color that differs from all headband colors and the color of the dodgeballs, making it simple for players to observe these points of information.

To differentiate the different agents, it can be easier to visualize them with different colors. This also helped participants distinguish between the agents which again helped participants connect their opinions of agents to something other than the order of play. The colors of the agents were selected based on their their visibility and compatibility with the rest of the environment. It was important that each agent had a unique color, but didn't have an advantage over the others, such as being camouflaged by the green bushes. Because of this, each of the agents received colors that made them "pop" in the environment and stand out.

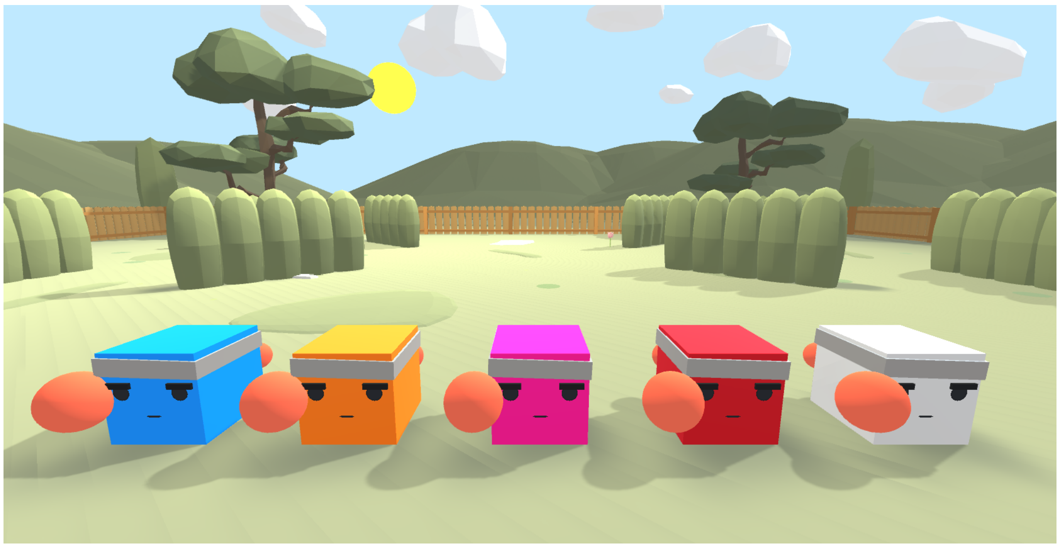


Figure 4.4.1: The different colors of the dodgeball agents. From the left: Player controlled agent (Blue), FSM agent (Orange), NEAT agent (Pink), MA-POCA agent (Red), Baseline agent (White)

The baseline agents purpose was to be a test dummy that participants would play with when practicing controls and in the baseline part explained in section 4.6. A neutral white color was selected for this so players would not have any bias towards one of the agent colors. Its worth noting that the original dodgeball environment consisted of a blue team and a purple team. The purple color used in the original environment was also removed in case participants should have prior experience with the environment, but was used during the training of the NEAT agent.

4.4.1 MA-POCA attention network agents

As described in the blog-post by Berges et al. [75], the dodgeball environment was primarily developed to showcase the capabilities of the Unity ML-Agents toolkit. One of the highlights of this environment is the MA-POCA (Multi-Agent POsthumous Credit Assignment) trainer, an advanced algorithm designed to train groups of agents to behave cooperatively. MA-POCA is a reinforcement learning algorithm designed specifically for cooperative multi-agent environments, like the Dodgeball environment utilized in this project. The algorithm operates on the principle of centralized learning with decentralized execution. A neural network, the centralized critic, processes the states of all agents to gauge their performance, while the decentralized actors, one per agent, control the agents' actions. Each agent makes decisions based on its localized perception and simultaneously assesses how well its behavior contributes to the group's overall performance.

The MA-POCA algorithm uses a special type of neural network architecture, called attention networks, which can process varying numbers of inputs. This allows the centralized critic to process any number of agents, making MA-POCA particularly effective for promoting cooperative behaviors in game environments. The algorithm can handle the dynamic addition or removal of agents from the group, which is useful in game environments where characters may be eliminated or spawn mid-battle. Furthermore, MA-POCA encourages agents to make decisions that benefit the group, even at their own expense. This selfless strategy, typically challenging to code manually, is learned based on the value of an agent's most recent action to the group's success.

This particular strength of MA-POCA is not fully utilized in this projects version of the dodgeball environment, where the numbers of players on each team has been reduced from 4 to 1. It is important to note this, since the performance of the MA-POCA agent in a 1v1 match may not reach the same level of efficiency as it would in a multi-agent setting. The algorithm's emphasis on actions that benefit the group, even when they are detrimental to the individual agent, might result in a compromise on individual performance.

Despite MA-POCAs primary design for cooperative multi-agent scenarios, the pre-trained MA-POCA agent also demonstrates adaptability in 1v1 settings. The decision-making process, which takes into account localized perception and overall performance learned during training, remains effective even when facing a single opponent. This agent is not further trained in this project, so its behavior and strategies are set and do not adjust to the specific game environment or the actions of the other agents it interacts with.

In the dodgeball environment, the agent's localized perception plays a significant role. This refers to the agent's ability to make decisions based on its immediate environment, such as its current position in the game field, the position of the opponent, or the location of the ball. This localized information is crucial for the agent to react effectively to changes in the game environment, allowing it to dodge incoming balls or aim accurately when throwing the ball.

4.4.2 The finite state machine agent

The MA-POCA agent is an excellent agent to compare against the NEAT agent, though neither of these two are anything alike the AI NPCs that players are familiar with in today's games. To observe the differences in emotional responses provided by classic NPC behavior and modern advanced AI NPC behavior, another agent was developed. This agent functions as a finite state machine, transitioning between states based on its observations in the game environment. By having an agent with behavior more alike that of classic NPC behavior, we can measure the performance the other agents by using the FSM as a benchmark for what normally is considered fun and engaging for a player.

As shown in Figure 4.4.2, the agent begins in the Patrol state. This is its default state, denoted by a circular shape. If the agent doesn't perceive any information around it, it will patrol on a set of waypoints. The agent continues this behavior until an observation triggers a transition between states. The agent passes through each of the eight waypoints in the waypoint list. After passing through all waypoints, the list is shuffled to randomize the agent's behavior. This strategy prevents participants from easily guessing that the FSM is a simple agent and disguises its behavior.

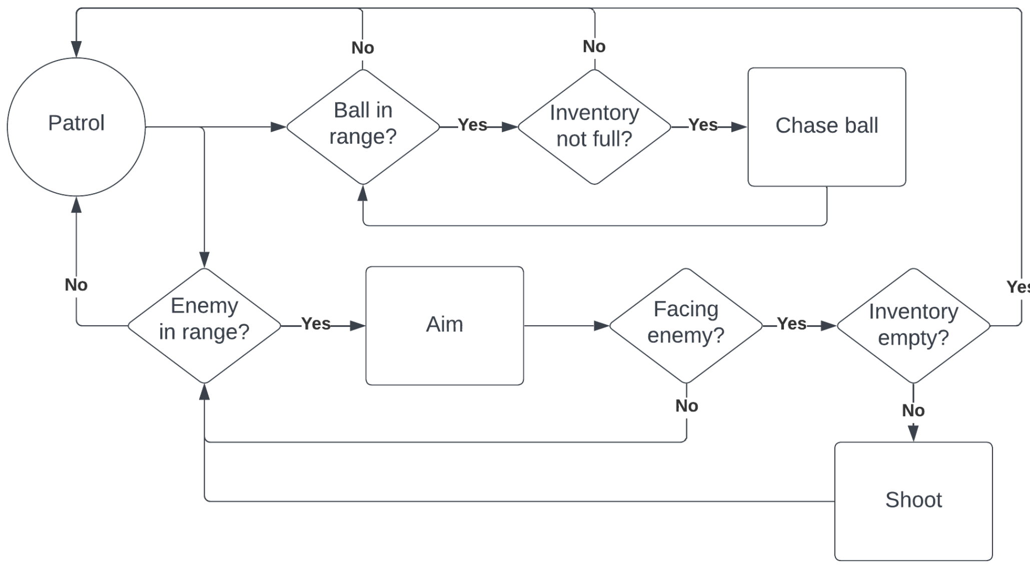


Figure 4.4.2: A diagram of the finite state machine agents states and transitions

The agent has two spherical sensory fields surrounding it. As the game takes place in a two-dimensional plane when viewed from above, these fields can be simplified to circular shapes. The sensory field for detecting an enemy has a radius of 10.0f, or 10 units, and the field for detecting balls is 5.0f, or 5 units. Every game object has an identifier called a 'tag'. The agent uses this tag to check whether an enemy or a ball is within its range.

When the agent perceives a game object with the correct tag within its sensory field, it initiates a transition towards a new state. This transition is based on the other conditionals represented as diamonds in the diagram. In the case of detecting an enemy, the agent will immediately transition into the Aim state.

In the Aim state, the agent slowly turns towards the detected enemy. However, its movement is still dictated by another state. This means that the Aim state is a concurrent state, which can occur simultaneously with other states such as Patrol and Chase Ball.

When the agent detects an enemy in front of it, it quickly checks its inventory for balls and immediately enters the Shoot state. This check is performed using a singular raycast in front of the agent, scanning the same game object tag as the sensory field detecting enemies. The agent will remain in this loop until it is out of balls. If the agent runs out of balls, it will keep the player in sight if close but will return to the Patrol state to find more balls.

A similar strategy is used to find balls scattered across the game map. When a game object with the ball tag is detected within the smaller sensory field, the agent will update its position to move towards that game object, as long as it can pick up more balls. If the agent is starved of observations or the conditions for transitioning into a new state are not met, it returns to the Patrol state as its default behavior.

The FSM agent is explainable and simple, while still providing enough complexity to create a somewhat engaging gameplay experience. Most importantly though, it is a decent benchmark to compare to the other agents and to evaluate their performance. While not as complicated as a fully developed NPC by today's standards, it is still built upon the same foundation of those NPCs [2, 3].

4.4.3 Adapting the dodgeball game for NEAT

The dodgeball game and its components were built around facilitating efficient learning of the MA-POCA agent issued with the dodgeball environment. Introducing our own NEAT agents to learn in the same environment posed some problems. First, as the focus was on the elimination game mode where the goal of the agent is to eliminate the opposing player, the observations regarding flags which were specific to the "Capture The Flag" gamemode were removed from the input to NEAT. Additionally, the number of observations were quite substantial. MA-POCA ran with over 1000 observations, where stacked observations made up the majority. After analyzing the observations in the original game, the stacked observations were redundant as the environment returned nothing but zeros for the extra observation stacks. The stacked observations were reduced to only one, which reduced the observation size greatly.

The game was already built on the MLAgents framework for Unity. The MLAgents library supports python by a package that delivers a low level API for python. This API was used to control the environment in Unity, similarly to the popular Gym [1] library for Python. The code for simulating a match of DodgeBall was implemented as a run_agent() function which is used by the python-NEAT package for fitness evaluation. The DodgeBall environment shipped with a training environment containing twelve platforms.

In Python, the agents on each team were assigned their own policy or genome from the Python-NEAT library, which in turn were used to take actions based on the observed state from Unity. The resulting actions which were applied to the Unity environment contained both discrete and continuous values. For the discrete values, the outputs from the neural nets were set to 0 (no action) if the output was less than 0 or set to 1 (action) if more than zero.

During a simulation, the rewards are collected for each genome of the population. At simulation end, the fitness is calculated using the accumulated sum of reward

[1]https://github.com/openai/gym

of which a particular agent received. Python-NEAT then performs the selection, crossover and mutation based on this accumulated fitness.

The population size was set to 72 or 144 depending on training against MA-POCA or self training. For MA-POCA training, the existing trained MA-POCA NN model was applied to the blue agent. Evolution was then performed with NEAT for the 72 other agents. For self-play, a fixed NEAT model controlled 72 agents and the other 72 agents were subject to evolution.

Originally the environment was built for attention networks, where each of the 12 levels in the environment for training ran independently performing actions and receiving rewards. This is not desirable in a neuroevolution context. It resulted in generations ending every time an agent won in a given level. This made training highly unstable and resulted in sub-optimal training results due to some generations lasting only a couple of seconds. For such short generations the rewards gathered were minuscule and sometimes even zero.

The code ended the episode for all agents when a any particular agent won. We adapted this by simply rewarding the agents and then removing them from the agent pool. Letting the rest play out their games or hit the 5000 steps limit before resetting all the environments. When a agent is eliminated, it and its opponent is removed from the game which registers them as "terminal_steps". The implemented python code checks for this and adds the agents to removed agents. When the last pair of agents finish or the max time-steps is reached, the generation is ended and a new one commences.

4.4.4 Solving continuous and discrete hybrid actions

The environment has three continuous actions for movement and two discrete actions for movement and dash. The simplest way of solving this is to simply convert two of the output values into a binary discrete output. Values lower than 0 are set to 0 and values higher than 0 are set to 1 to ensure a even distribution from -1 to 1.

For the output layer of the neural network, a hyperbolic tangent function which outputs in the range of -1 to 1. This is the same possible range of the continuous actions which makes it suitable for this task.

$$output_{discrete} = \begin{cases} 0 & \text{if } output_{continuous} < 0 \\ 1 & \text{if } output_{continuous} > 0 \end{cases} \tag{4.1}$$

4.4.5 Counteracting erratic AI behavior

In early phases of training, the movement performed by the AI may be erratic. This can be due to no obvious advantage of not being erratic, using erratic behavior to observe more of the input space in shorter time or similar. For a human playing against AI exhibiting consecutive actions of which delta is quite large, it can be judged as incorrect, confusing or even annoying.

Zuniga et al. [80] discovered that smooth physical movement, logical reasoning, and goal-oriented behavior were traits commonly associated with human players. Interestingly, participants who mistook a human player for an AI also reported these same traits as being indicative of a human player. However, the number of times participants misidentified an AI as human was significantly lower than correctly identifying a human player.

Most subjects that misjudged a player for AI used jerky movement, i.e. fast and sporadic movement with no discernible purpose as reasoning. These actions were interestingly overlapping with what factors were used to identify AI. AIs were also identified by reasoning with factors like goal indirectness and illogical reasoning.

Jerky movement is usually prominent during early training stages, requiring substantial training time to reach a smooth movement pattern. Some methods to aid in removing such jerky undesirable behavior, is to punish those during training. Large deltas, or differences between two subsequent actions an agent makes, can be observed as jerky and random. By punishing such large changes, the agent will learn to not avoid such behavior.

The agents might also learn to spin around to effectively observer 360 degrees around them in the dodgeball environment. This can be jarring to play against and could potentially be quite distracting. In this case the input would not change much, but by punishing full subsequent rotations, such behavior can be avoided. Jerky movement can also be avoided by applying action output smoothing to the output from the neural network. The output can be limited to stay within a range of the previous output.

4.5 NEAT algorithm implementation

In this chapter the implemented framework with NEAT will be discussed. Experiments on the most optimal configurations to facilitate efficient training and experiments on the optimal training approach is some of the key subjects with substantial impact on the outcome of training sessions. Challenges like erratic behavior and parallelization of the training process is also elaborated upon.

4.5.1 Configuration

The DodgeBall environment required relatively complex neural net topologies, due to the complexity of the environment. Hidden nodes were set to 0 due to a bug discussed in Appendix Section D. To avoid overly simplistic network topologies and long training times, the probabilities of adding nodes and connections were set to 0.8. The probability of removing nodes and connections were also increased but kept at 0.5 to facilitate bigger network topologies.

The NEAT configuration file consists of 50 parameters that are tuneable for best results in a specific task, and previous research (mentioned in Section 2.6.9) has shown success in a similarly complex environment by tweaking the probabilities of adding and removing nodes and connections.

The network is configured with 364 inputs for the adapted state space from the DodgeBall environment and 5 outputs for the possible actions. In case of population stagnation, the two species with the highest fitness are kept for the next generation to avoid losing all genetic evolution following a population refresh.

The compatibility threshold is kept at the default 3 to maintain a sufficient level of speciation in the population. The weight mutation power is kept at the default 0.5, which is on the higher end. This allows the weights to be mutated by bigger factors for more exploration. The fitness criterion is set to 2.1 calculated as a mean, meaning the mean fitness of the entire population would be at a satisfactory level (2.2 is the best fitness score obtainable). Two separate configurations were implemented, one for regular feed forward ANNs and one for CTRNN. The difference between them was the feed forward was set to false to enable the CTRNN network to develop recurrent connections.

The initial connection type was set to partial_direct connections. This is essentially the same as a full_direct connections where each input node is connected with all hidden and output nodes. The difference is that each connection has a probability of being present with the defined chance of 0.5 e.g. 50%. The full configuration can be seen in Appendix Section C

4.5.2 Evolution framework with NEAT

Before evolution with NEAT can commence, the Unity environment needs to be connected to the NEAT implementation in python. This is done by connecting the python trainer to the MLAgents instance within the Unity environment as illustrated in Figure 4.5.1.

Once a connection is established, all the agent indexes of the initial agents requesting actions are stored. Due to MLAgents assigning new agent indexes for every generation, a mapping from local indexes to agent indexes was created with a dictionary. This mapping lessens the dependency on agent indexes in the python script.

Initially before evolution, NEAT generates a set of genomes in a population with mutations. Together with the config, these genomes are used to specify the creation of ANNs or CTRNNs. Additionally, new ANNs or CTRNNs are created between generations through the specification of the descendant genomes and the config before they can be simulated in the environment.

The Dodgeball Unity environment is then used to simulate these genomes in the environment for evaluation. The observations as elaborated in Section 4.2.1, are combined into a single observation array. The observations is then fed into the ANN or CTRNN to obtain a action from each genome by inference as shown in Figure 4.5.1. Once five continuous actions are obtained per agent, the last two actions are discretized. The discrete boolean for both actions are set to 1 if the neural net output is more than 0, while outputs less than 0 is set to 0. This ensures that the action output from the neural network is normalized from -1 to 1. Actions are then set for each agent in the environment, and the environment is signaled to continue the simulation until the agents requests a new decision.

If any agent has finished the game with their opponent, they appear in the terminal_step object. This object is used in the algorithm to keep track of all the eliminated agents. If the simulation after calling env.step() results in a reward, it is directly summed up the reward of the genomes of NEAT. Once all the agents has finished their game or the time-step limit of 5000 has run out, the generation ends and NEAT performs crossover and mutation on the generation.

4.5.2.1 Saving progress

During evolution, it is important to keep the progress saved in multiple forms. The most important way to store progress in the NEAT library for python, is

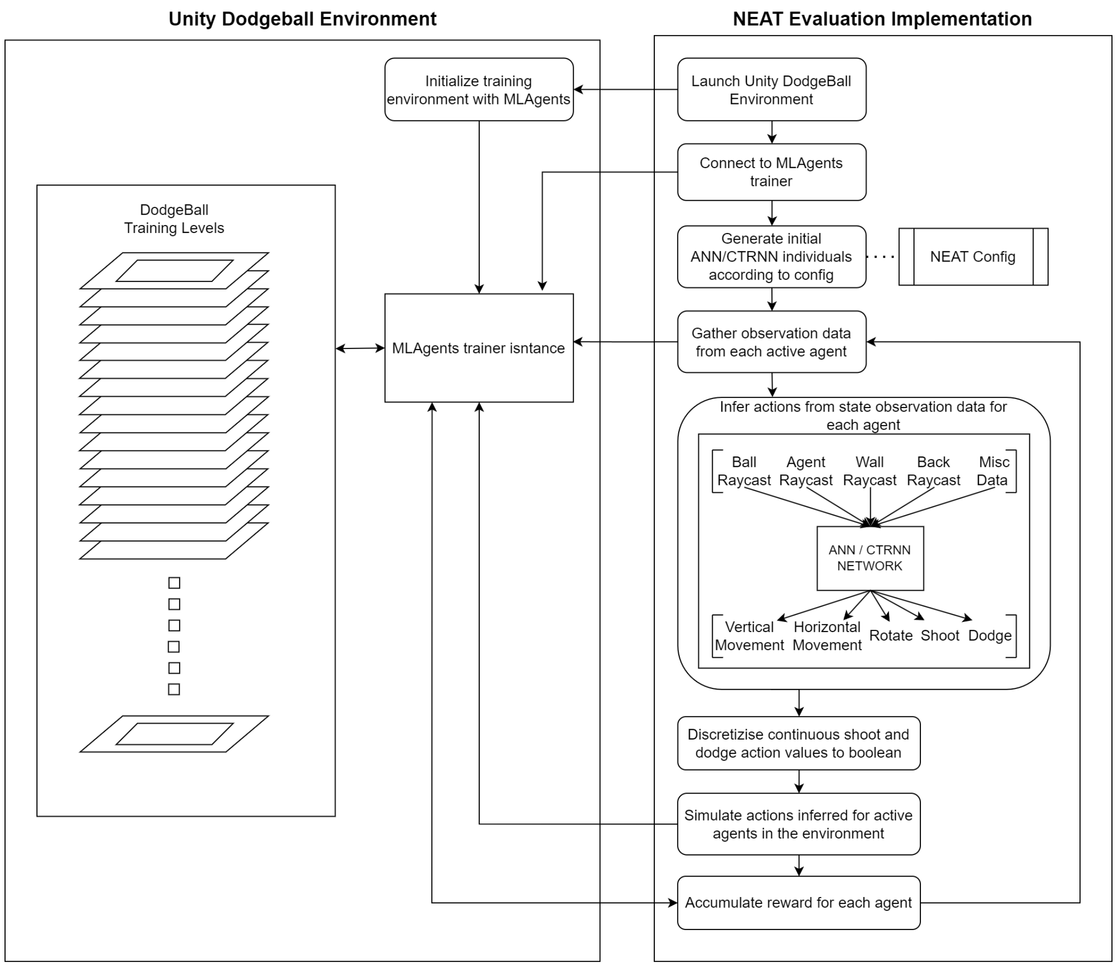

Unity Dodgeball Environment
NEAT Evaluation Implementation
Initialize training environment with MLAgents
Launch Unity DodgeBall Environment
Connect to MLAgents trainer
DodgeBall Training Levels
Generate initial ANN/CTRNN individuals according to config
NEAT Config
MLAgents trainer isntance
Gather observation data from each active agent
Infer actions from state observation data for each agent
Ball Raycast
Agent Raycast
Wall Raycast
Back Raycast
Misc Data
ANN / CTRNN NETWORK
Vertical Movement
Horizontal Movement
Rotate
Shoot
Dodge
Discretizise continuous shoot and dodge action values to boolean
Simulate actions inferred for active agents in the environment
Accumulate reward for each agent

Figure 4.5.1: Figure illustrating the evaluation process as part of the NEAT evolution process. This evaluation happens during each generation and produces a fitness of which the genomes are selected by during the selection process. The dodgeball environment is connected using the MLAgents library, where the actions are inferred from the evolved genomes by inputting the observations gathered from the environment into the model and obtaining values for the 5 types of continuous and discrete actions.

checkpoints. Checkpoints contains that entire training progress, information about fitness, genome history, population data, stagnation and more evolution relevant parameters like the general configuration. For the checkpoint storage, the built in checkpoint methods within NEAT were utilized.

The framework built for evolving agents with NEAT was set up to save the checkpoints every 20 minutes or 25 generations, whatever comes first. Additionally, the best genomes from every 50th generation were saved to file including which generation it came from in the filename.

To keep training progress visualization data, the average fitness, best fitness and species progress were saved to file. During early training, there was issues with NEAT training crashing and the need to start up the training from a checkpoint. For a training standpoint it had little impact, but for visualizing training progress it resulted graphs with varying generation length.

4.5.3 Experiments on optimizing the training strategy

The optimal training strategy is hard to find, however through experiments, the training strategy can be improved to allow efficient evolution and good results. From the beginning, the environment was built to fit the RL algorithm as described in Section 4.4.3. Where all agents were continually playing against each other to learn state-action relationships. However, this did not fit well with the evolution evalutation process. All the training levels in the Unity dodgeball environment had to be synced, instead of restarting independently, in order to assess each genome in the population until a collective game ending.

The environment was changed into removing agents from a level once one of them was eliminated. This benefited by lessening the computational load once the simulation progressed and more agents got removed thus removing the need to render them. Once all agents finish, as in one of the opposing agents were eliminated, or the time limit runs out, the environment is reset and ready for simulating the next descendant population.

Multiple training runs were performed to evaluate the training efficiency of incremental challenge, fixed challenge and self-play. NEAT did not spawn any other species than the one it started with, which is due to low genetic distance within the population. After extracting the best genome self-play, it was used as the fixed challenge for the next phase of training. Even though the resulting agent was not great, it served as a basic challenge for training. The number of levels, or playing fields consisting of two opposing agents, in the environment was doubled from 36 to 72 with 144 agents. This was done to keep the original population size of 72 since the purple agents (half of the agents) was controlled by a single fixed genome policy. The training started from scratch with the best genome.

Employing a fixed opponent as the challenge used for evaluation, lead to every genome being evaluated against the same task. Self-play were deemed too unstable due to the vast variations in challenge. Training against fixed opponents led to better training stability and clearer fitness progress.

Playing against a fixed agent from scratch, individual fitness scores began stagnating at around 150 generations, were the best agents consistently got a reward of 2.1 or higher. The adjusted species fitness per generation began stagnating at around the same time, where the adjusted fitness hovered around 0.500-0.700. After around 350 generations, 5-10 agents began eliminating their opponent quick enough to get a fitness above 2.1 every round. This progress was a great indication that the agents began learning.

After training against the agent with basic skill for 400 generations, the best genome from the last generation of this training run was saved. This more skilled opponent was used as the next challenge for further training. The next 140 generations did not show a clear improvement just by analyzing the best and average fitness progress as seen in Figure 4.5.2.

However by visually inspecting the playing style, movement was less random and shooting was more accurate. After the run of 140 generations, the best genome was saved from the 140th generation. This genome was then used as the new challenge for the next 200 generations of which fitness progression can be seen in Figure 4.5.3.

After finishing 740 generations of training with incrementally increasing chal-

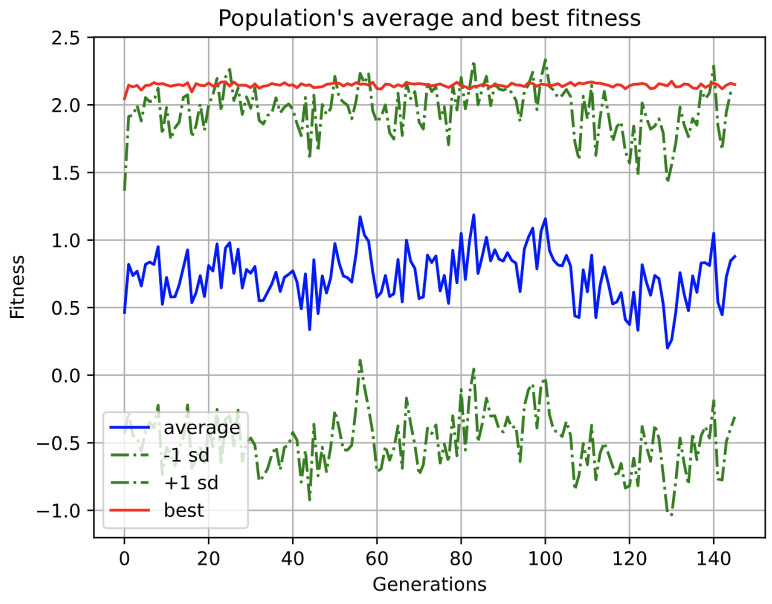


Figure 4.5.2: Training progress for the second phase for 140 generations with a incrementally skilled opponent. This training continued from the last checkpoint of the initial training of 400 generations

lenge, the agents learned to defeat their opponent challenge within reasonable time. It only took around 10-20 generations before the newly evolved genomes managed to beat the challenge consistently. With the new skill level of the agents, it was time to introduce more complexity to the game environment. First, the number of balls spawned was reduced from 15 to 6. Having 15 balls on the field reduced the need to actively seek out balls to fight with. The agents also tended to hit pools of balls which resulted in flying balls over the whole playing field. Reduction would force the agents to be more clever about how they throw and obtain balls. Secondly, bushes were added in, with two bushes in the middle at each their half of the field. The new training setup can be seen in Figure 4.5.4. The fitness progression of 300 generations with this setup can be seen in Figure 4.5.5.

Training with this setup was continued until 3000 generations had passed. The best genomes stored periodically from generations were evaluated to not be on par. Mostly due to spinning and failing to obtain dodgeballs. It seemed like there was some kind of overfitting that happened. Overfitting on a strategy managing to beat the challenge but not function well in a environment against a human.

When this learning strategy did not give desired results, we looked into training against the MA-POCA agent that shipped with the environment. Our theory on why the agents did not behave as expected was that they overfitted too much to the suboptimal playstyle of primitive agents. The MA-POCA agent had seen 160 million timesteps of training, leading to a relatively smooth in-game behavior, which resembled human gameplay more than the early NEAT agents could. In addition we changed the configuration of NEAT in order to promote the growth of network topology. Facilitated by changing the connection and node add probability rate to 0.8 while keeping the remove probability at 0.5.

A new custom environment was created to allow the genomes to be evaluated against MA-POCA, and the original level layout was kept as seen in Figure 4.3.3.

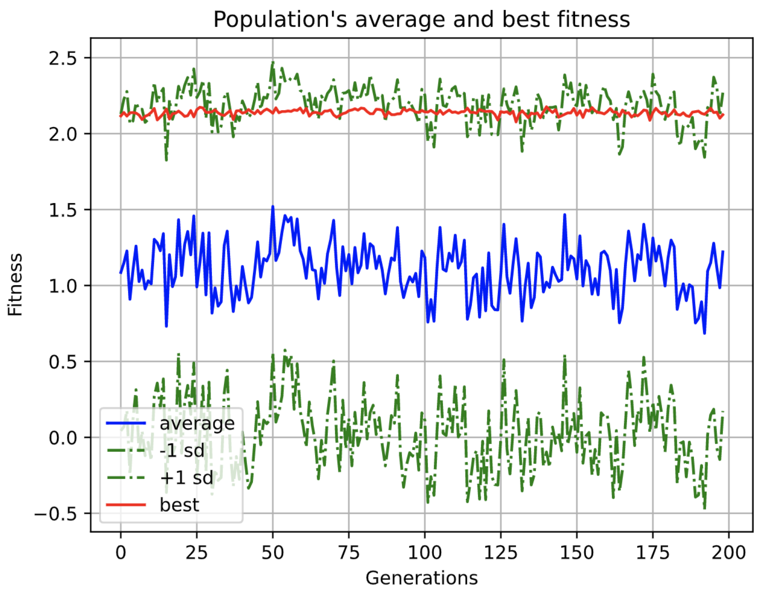


Figure 4.5.3: Training progress for 200 generations with genomes evaluated against the best genome from the end of the previous training phase as seen in Figure 4.5.2. This training continued from the last checkpoint of the last training phase.

The original level was originally designed for 4v4 agents, but its bigger size and abundance of bushes gives more breathing room for the player. This was important due to the control scheme at hand. Training was run for 2000 generations with the CTRNN setup, we noticed that genomes based on CTRNN that did not stray far from each other in fitness could behave noticeably different, some could be unplayable (as in not functioning properly) meanwhile others were skillful enough to defeat human players.

As can be seen in Figure 4.5.6, seemingly optimized agents in terms of fitness were found within the first 100 generations. The population average fitness began to stagnate at around 500 generations. Subjective evaluation of the saved genomes was performed by playing against them.

The ANN genomes, on the other hand, analyzing behavior in game told a different story than the fitness progress would indicate. The best genomes had failed to develop any significant behavior, with strange behavior where the genomes preferred to stay at their base while only moving left and right while shooting the balls it coincidentally picked up. A fresh run of 500 generations from scratch with the ANN confirmed this behavior, which could not be explained by investigating the code. This was the reason that ANN was scrapped for the empirical superior CTRNN network architecture. The CTRNN on the other hand exhibited skillful behavior, though with some spinning and jerky characteristics.

4.5.3.1 NEAT configuration experiments

Experiments were performed to assess the training progress impact of the default config parameters (besides obvious configurations to make the algorithm work with the environment) suggested by the libraries creators and a modified configuration

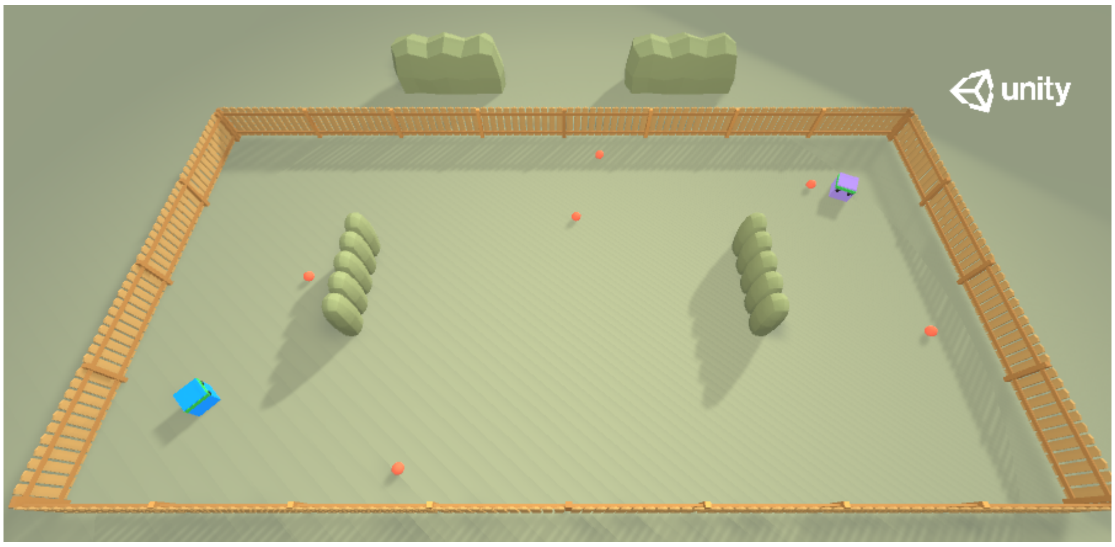


Figure 4.5.4: Challenge including bushes, less dodgeballs spawned and the best genome from previous training phase.

to suit the task. The custom configuration, as seen in Appendix Section C, had a higher connection and node add and removal probability. The add probability was set to 0.8 and removal to 0.5. The initial connection was also set to partial_direct with a probability of 0.5, in comparison to the unconnected default.

As hypothesized, greater complexity and more connections allowed the networks to be evolved towards defeating the opponent faster than with the default configuration as seen in Figure 4.5.7. Because of this, the modified configuration was used for further training. Interestingly, dips in the best genome fitness can be observed, accompanied by the +1 std decreasing and -1 std increasing towards the average. This signalizes a low variance in the population where innovations are either lost or does not represent a consistently skillful solution. The complexity and stochasticity of the environment (ball trajectory when interacting with the environment) might have been the reason that genomes receiving a high fitness were lucky and cannot reproduce the win in the next generation. The ANN seemed to perform better with a consistent high fitness, but this may have been attributed to luck during initial search, while evolution with CTRNN found a local optimum. As such dips in best genome fitness is evident in both configuration types, suggests that too aggressive mutation setting of connections and nodes amongst other hyperparameters cannot be blamed entirely.

The performance effect of the fitness sharing was also tested out as seen in Figure 4.5.8, in order to assess the impact on learning speed of the algorithm. Although 100 generations is arguably not enough to draw conclusions and assess the long term effects of disabling the speciation and fitness sharing components of the algorithm.

As expected there was not a big difference in performance. The ANN showed faster fitness development towards 0.0 in fitness, but the CTRNN did not show a clear difference. The faster fitness development could also be due to randomness in search. When disabling the fitness sharing, the individual genomes are not punished for being parts of a big species which would lead to more exploitation and less exploration. Even though this may speed up the training process initially,

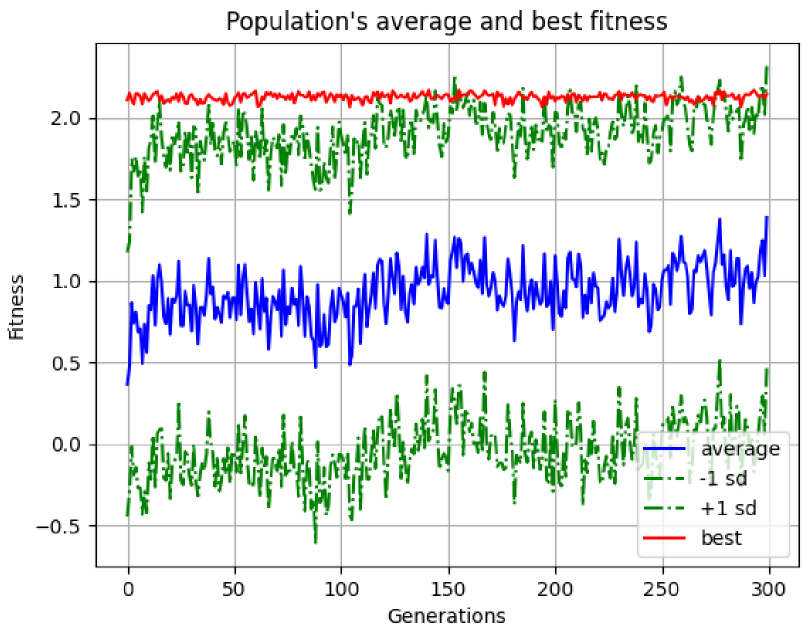


Figure 4.5.5: The fourth training phase showing a example of the fitness progress with bushes and less dodgeballs for pickup.

it would most likely stagnate once multiple species spawn and lead to less diversity amongst genomes.

4.5.4 Erratic behavior

The genomes had a tendency to exhibit erratic behavior as explained in Section 4.5.3. We discussed approaches to solve this in Section 4.4.5. We started with the simplest method for limiting the unwanted behavior. The values passed as input to the agent were limited to a interval as seen in Equation 4.2, where f is the limiting function and y is the raw action output inferred from the model. This change had a big impact on the behavior of the agents, the resulting movement and behavior were smoother and visibly more concise. Observing the agents, the 0.3 interval limit did not seem to hinder the agents ability to turn around quickly enough.

$$f(y) = \begin{cases} -0.3 & \text{if } y < -0.3 \\ 0.3 & \text{if } y > 0.3 \end{cases} \tag{4.2}$$

4.5.5 Saving high fitness individuals and selection of the best genome

After the completion of the evaluation of each generations, the best genome in terms of fitness were saved. During evolution those best genomes were saved to file every 50th generation, keeping history and the possibility of comparing the agents at a later stage. The training performance was periodically inspected by extracting a sample of high fitness individuals, playing against them and judging the playing style. The selection of the best genome for the experiment, were done by extracting

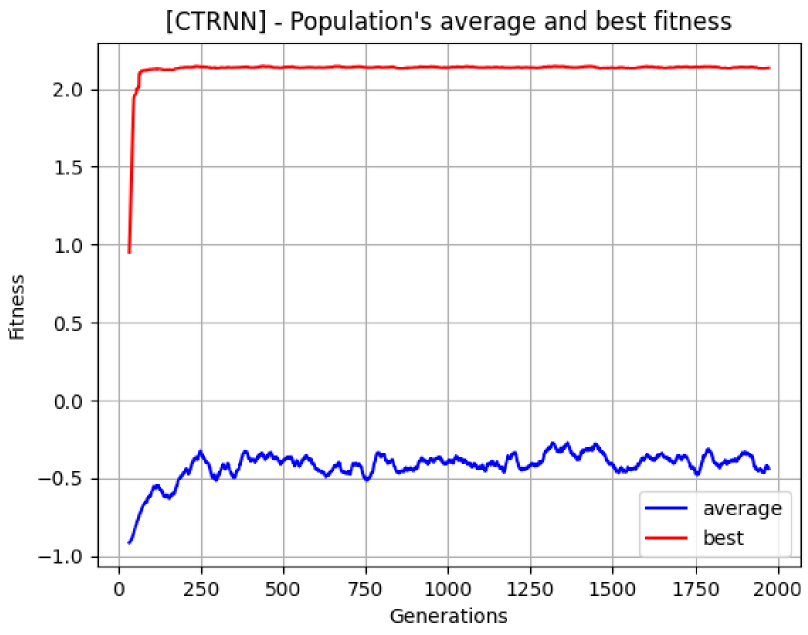


Figure 4.5.6: Fitness development training using the CTRNN network architecture against MA-POCA agent for 2000 generations.

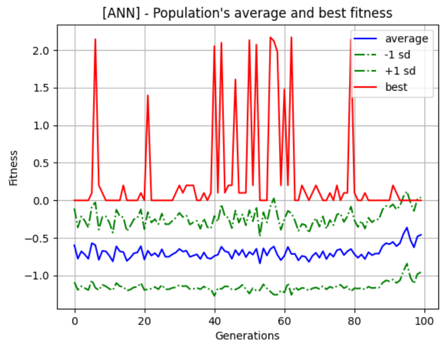
the genomes saved during evolution and playing against them to evaluate the best genome. The selection were performed as a tournament were pairs of genomes were evaluated against each other. A player and two others observing judged the winner of each pair, to avoid selecting based on one preference.

4.6 The fMRI experiment

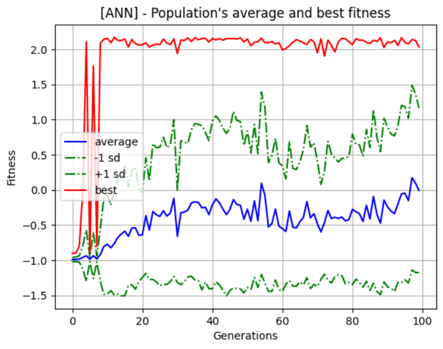
This section will delineate the methodology employed in the fMRI study, following the completion of development for both agents and the game environment. We will explain the preparatory measures taken for the study and provide a comprehensive, step-by-step account of the procedures carried out during the studies. The procedures applied for each of the 13 participants, were mostly the same. Some variations were implemented to reduce order bias towards the agents. There were also unexpected challenges that happened during the studies, that had to be accounted for. These are also elaborated upon in this section, and discussed further in Section 6.4.2.

4.6.1 Preparations

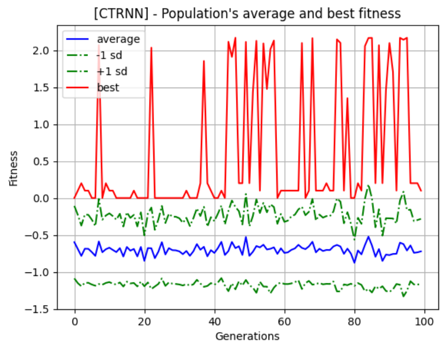
Conducting an MR study needs thorough preparation to not be wasteful of the resources required for the project. The Magnetom Terra is a highly expensive piece of equipment and is desired for several clinical researches. Due to the limited booking time available for the project, the fMRI study had to be well prepared to be as efficient as possible. A total of 20 hours was available for booking. With an estimate of 1 hour and 30 minutes needed for each participant, 13 slots were allocated for participants over the span of 8 days.



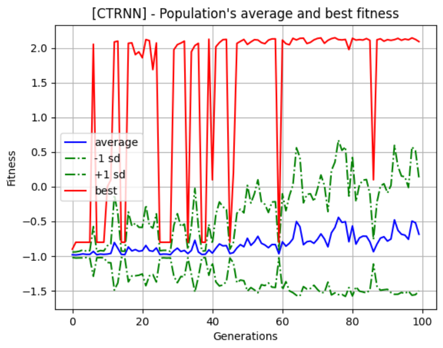
(a) ANN Default Config



(b) ANN Custom Config



(c) CTRNN Default Config



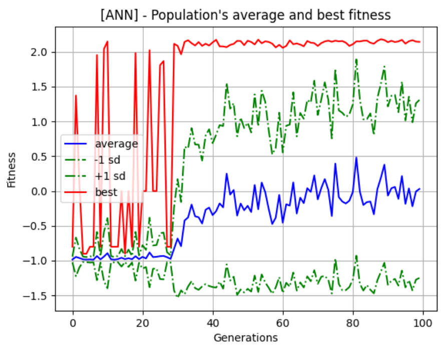
(d) CTRNN Custom Config

Figure 4.5.7: Training progress comparing the default configuration and the customized configuration. Evolution was run once per configuration for 100 generations. Comparing ANN by (a) versus (b) and CTRNN by (c) and (d) indicates a more effective fitness progression with the new parameters. With the new configuration, the best genome can be seen to more consistently hit higher fitness.

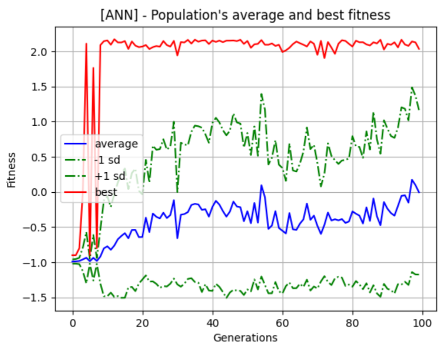
Participants were recruited for the study through several methods. There were no prerequisites for the candidates, though a preference was made towards younger audiences to account for easy adaptation to the difficult control scheme described in 4.3.1. All participants in the project were aged between 20-29 and had a certain degree of experience in playing video games. When asked about their experience in playing first person or third person shooter games, the lowest score was 2 on a scale from 1 to 5, where 1 is no experience and 5 is great experience. The mean related game experience was 3.7 out of 5, meaning players perceived themselves as skillful in game experience. The rest of the participants answered 3 or higher. Having previously trained motoric skills and muscle memory in shooting games helped accelerate the learning curve of the game to not waste time when conducting the study routine.

4.6.2 Arrival at 7T

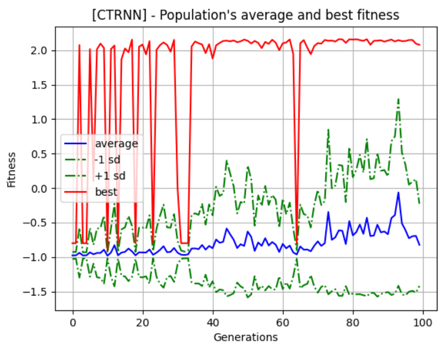
Upon arrival, participants were greeted and escorted to a designated room where they could securely store their belongings. This room, part of the facility, is located across the hallway from the operational room. It offers a space for participants



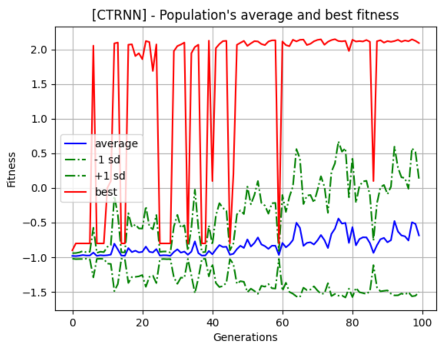
(a) (ANN) No fitness sharing



(b) (ANN) Fitness sharing



(c) (CTRNN) No fitness sharing



(d) (CTRNN) Fitness sharing

Figure 4.5.8: Training progress comparing the performance of the custom configuration with and without fitness sharing. For the comparison, the custom config (Appendix Section C) was applied. Evolution was run once per fitness and NN setup for 100 generations.

to sign agreements, complete questionnaires, and change into MRI-compatible clothing. Furthermore, it keeps participants away from the operational room to conceal information regarding the task potentially introducing bias.

Initially, participants were asked to read and sign a non-disclosure agreement. They were then permitted to practice against a stationary agent in a baseline scene, enabling them to familiarize themselves with the game's rules and controls. The baseline scene features the same game map as the one used during actual play, but with a white, inactive enemy agent. A researcher monitored the practice session to ensure all players comprehended every game mechanic. This step guaranteed a consistent base understanding of the game among participants, reducing bias in the results. All other information about the agents was concealed, except for the fact that they would compete against three differently colored agents.

Once the participant was prepared, the radiographer conducted an fMRI safety routine with them. This procedure is a vital safety measure to prevent any harm to the participant or the equipment. Subsequently, participants were guided into the scanner room and positioned on the fMRI table. Given the limited space and awkward posture for operating the controls, additional time was devoted to securely fastening the Tethyx joystick controller in place. This ensured the necessary comfort for approximately an hour of scanning exercises.

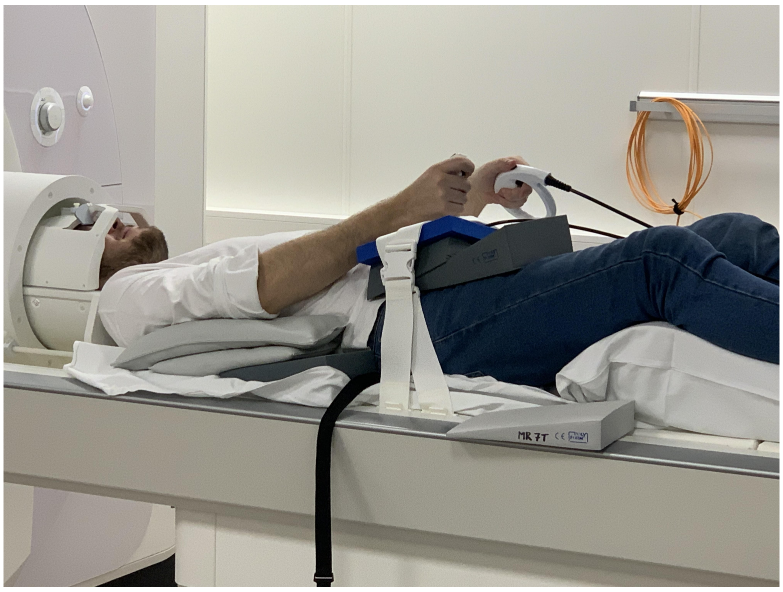


Figure 4.6.1: An photo of the equipment setup used as a reference by the radiographers

4.6.3 In the fMRI machine

As the participants were positioned inside the fMRI machine, researchers in the control room carried out several procedures. First, all the systems responsible for the transistor-transistor logic (TTL) sync signal, which is the signal sent from the fMRI scanner declaring the commencing of the scan, the controllers and video signal to the screen where turned on and configured correctly. The module called BOLDscreen 32 AVI box, responsible for transmitting the HDMI output to the experiment screen, was configured to use the HDMI with mirroring activated.

Next, the NNL-sync box was activated, a module responsible for the TTL synchronization signal. The box was configured by entering synchronization mode and configuring the volume setting to 9999. Then, the Current Design box was configured by selecting "configure new joystick", then selecting "autoconfigure", whereafter the correct joystick name should appear on the display on the device.

Once all systems were ready and the display PC was connected with the WebLink interface, the eye-tracking system needed to be calibrated to match the participant's vision. This could be executed while the initial brain mapping was being scanned. The preliminary scan conducted by the fMRI machine generates a "map" of the participant's brain. Subsequent scans, which display neural activity, are then superimposed on this map to identify the active brain regions. Since the initial scan does not involve mapping neural activity, this time was efficiently utilized by having participants perform the eye calibration concurrently. The calibration process involved participants focusing on dots that appeared at specific locations on the screen.

After completing the preliminary scan, participants proceeded to engage in the game. Counterbalancing was implemented to account for order effects and potential biases when assessing the appeal and interest of the three game agents. This was accomplished by varying the sequence in which the agents were introduced to the 13 participants, ensuring equal opportunities for each agent to be encountered in different positions (first, second, or third) across the participant group. Initially, we overestimated the time slot of one and a half hour, and assumed that participants would have time to play against each agent for 15 minutes. As shown in Section F in the Appendix, specifically the notes from the first three days, this overestimation led to down-scaling of playtime. Each agent was therefore allotted 8 minutes of playtime against the participant after planning more carefully and timing the events of the fMRI study. This means that the three first participants each played against two agents instead of three.

Following the 8-minute period, it was necessary to provide participants with a brief break to allow their brains to "rest" before proceeding to the subsequent agent. This brief pause of 5-minutes enabled the brain to settle, preventing residue of neural activity associated with one agent from influencing the experience with the next agent. During this interval, participants were asked questions about the agent they had just interacted with.

This set of questions was designed to assess the engagement and enjoyment experienced by participants during their interaction with the agent. A total of eight questions were posed, with the first six requiring them to rate their experience on a 1 to 5 scale. This activity was conducted during the fMRI scan to ensure that the agent's behavior remained vivid in the players' minds. The scale consistently used 1 as the lowest score and 5 as the highest score.

Following the rating process, participants were asked to provide more in-depth responses to two additional questions. The first question inquired whether they had to develop a strategy to defeat the agent, while the second focused on determining if they observed any particular behavior exhibited by the agent.

To eliminate the possibility of bias where participants could observe which agent was launched during the study, preventive measures were taken. First, the references to the agents were changed to the name of their respective color. Secondly, it was important to avoid participants observing the running of the Python script upon launching the NEAT agent. This was avoided by disconnecting the screen from the laptop before running each agent. Curious participants could possibly deduce when the NEAT agent was run if the screen was unplugged only before one of the agents.

4.7 Eye tracking

For eye tracking data, a solution called WebLink was made available for this project. As mentioned in 2.8, an eye tracking system is an accessory module available to the 7T fMRI scanner at St.Olavs. There are quite a few necessary factors to control for the setup to work. First off, the WebLink software was a key component requiring some configuring.

The WebLink software enables the display PC (the computer running the experiments) to have full control over the EyeLink software on the Host PC through an Ethernet connection between the two. Before WebLink, experiments communicated directly with the EyeLink software on the Host PC for recording control. For the integration to work, IP 100.1.1.2 and 100.1.1.2 must be allowed through firewall and open to sending and receiving requests on the display PC. Without it, the display PC will fail to connect to the Host PC.

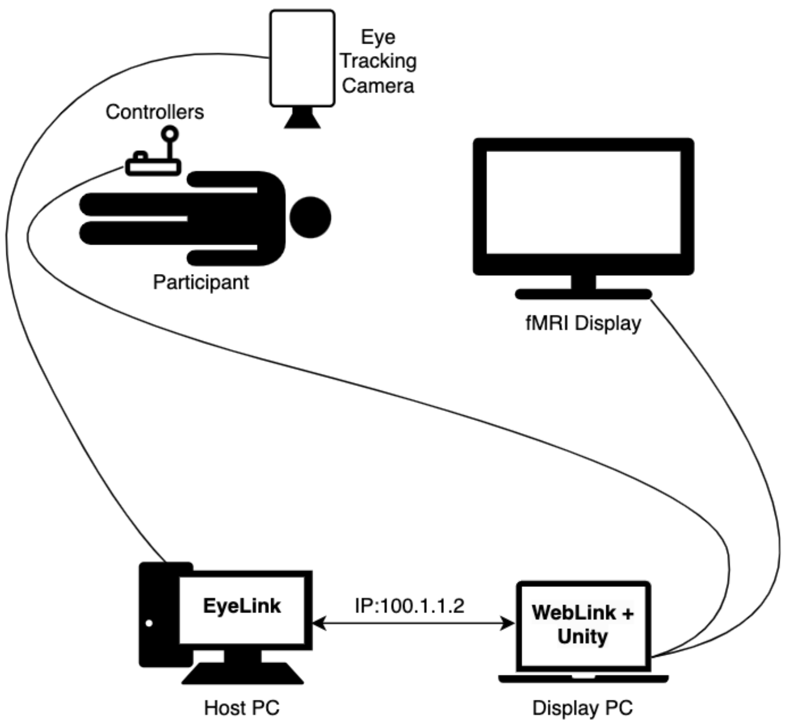


Figure 4.7.1: Overview of the complete eye tracking setup with the Host PC running EyeLink and Display PC running both the WebLink software controlling the Host PC in addition to the Unity game.

The task running on the Display PC will then be recorded in addition to the eye data once a eye tracking recording session is started. A data file containing the screen recording, eye data, log (including key presses by the participant during the session) and more is saved to the display PC.

The tracker relies on successful calibration for the eye to be tracked correctly. Due to visibility constraints on the screen, the five point calibration scheme was utilized. Which consisted of a dot in the center, in addition to dots directly left, right, above and below the center. The calibration was essential to compensate for the different head-position of each participants within the scanner. In addition correctly focusing the lens on the tracker is vital for a sufficient tracker accuracy. The focusing required might change from participant to participant.

The EyeLink software provided functionality called a timeline, where the experiment could be planned out. Unfortunately, it was hard to utilize this with the setup that had to be run for the experiment. For the FSM and the MA-POCA agent, only the game itself had to be run. This could have been automatically launched in the WebLink software. However, the NEAT agent required more attention. The implemented framework for the NEAT agent included automatically launching the game directly from python using the MLAgents python library. However, as the game was set up to wait for the TTL signal, it would cause the MLAgents connection to time out if it was left in this state for too long. Timing the start of the NEAT agent and the corresponding environment with the start of the fMRI scan was hence crucial, leading to not utilizing the timeline to plan out the experiment.

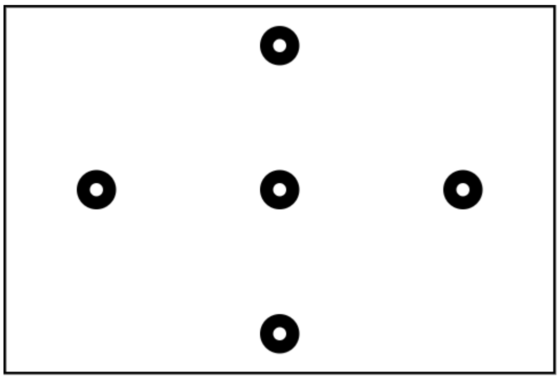


Figure 4.7.2: The five-point calibration screen as presented to the participants.

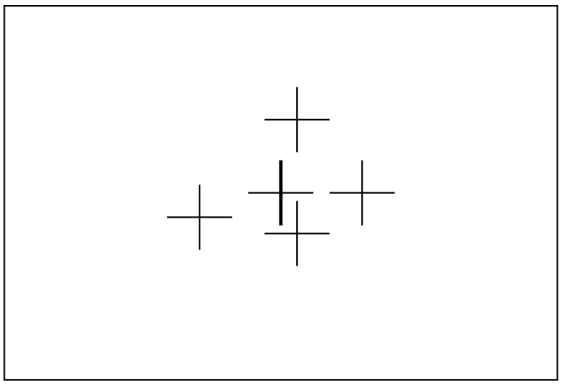


Figure 4.7.3: How a poor five-point calibration result can look like like.

4.7.1 Logging attention areas in-game

The WebLink software has support for interpreting interest areas which can be logged for game objects in Unity. These interest area files can be imported into the DataViewer tool, which is the WebLink tool for analyzing all the data collected after the experiment. Logging of interest areas were implemented in Unity for the game. The interest area consisted of the opponent enemy. Every game tick where the opponent was visible for the player, it would be logged. When imported into DataViewer, it shows up as a cube around the enemy. Together with the gaze data recorded with EyeLink, the DataViewer software analyzes amount of times and the duration of times where the eye focused on the interest area.

4.8 Questionnaires

Gathering data subjectively experienced by the participant, was done by employing questionnaires. Two questionnaires were developed for the case of this study. One questionnaire was designed to be asked throughout the scan session, with participants providing scores and comments immediately after experiencing each agent, thus capturing their impressions while the experience was still fresh in their memory. Additionally, the participants were asked to fill out a post-scan questionnaire after the complete fMRI scan session had been completed. The post-scan questionnaire gave an opportunity to rank the agents based on the full context of all agents. In conjunction, both questionnaires can determine the

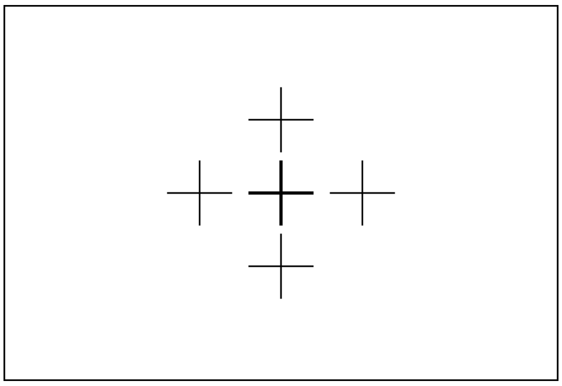


Figure 4.7.4: How a good five-point calibration result looks like

consistency of the answers given by the participants.

4.8.1 Questionnaire after each agent

The questionnaire asked throughout the scan session was designed to assess the emotions experienced by each participant. The construction of the questionnaires were inspired by the concept of flow [36] and the Self-Determination Theory (SDT), a motivational framework developed by psychologists Edward L. Deci and Richard M. Ryan [81]. Similarly to the theory of flow, STD also addresses the balance between skill and challenge for optimal engagement. It emphasizes the importance of intrinsic motivation, which is driven by an individual's interest and enjoyment in an activity. The theory proposes the idea that optimal motivation and engagement occurs when three basic psychological needs are met: autonomy, competence, and relatedness. The finished questionnaire surveyed both the qualities that the agent brought to the game and the aspects contributing to intrinsic reward and the feeling of flow.

1. Participant name
2. Rate to what extent you liked to play against the agent (1-5)
3. Rate to what extent the behavior and actions of the agent led to a more interesting game (1-5)
4. Rate to what extent you felt mastery when succeeding against the agent (1-5)
5. Rate to what extent you felt the agent offered a balance between challenge and joy (1-5)
6. Rate the entertainment value of the agent (1-5)
7. Rate to what extent the agent had a fitting challenge to your skill level (1-5)
8. Elaborate on any unique strategy utilized to beat the agent if it was necessary.
9. Comments regarding the agent

Each question aims to measure an aspect of the participant's experience playing against the particular game agent, including their level of enjoyment (Question 2), perceived mastery or competence (Question 4), balance between challenge and joy (Question 5), and the adequacy of the challenge level to their skills (Question 7). The overall entertainment value (Question 6) and open-ended responses (Question 8 and 9) allow for further exploration of the participant's experience and their interaction with the game agent.

4.8.2 Post-scan questionnaire

The final questionnaire was based on ranking of the different agents in relation to each other. After playing against each agents, the participants had a holistic experience of playing against every agent. The final questionnaire was used to assess the participant's preferences between the three agents. It consisted of several rankings on different metrics, with following options to elaborate on the reasoning behind the ranking. Additionally data like concrete age, level of first or third person shooter game- and joystick experience was surveyed for a reference during analysis.

Questionnaire

The participants were asked to answer the following questions:

1. Participant name
2. Rate your total liking of playing against the agents (1-5)
3. Rate how noticeable the difference was between agents (1-5)
4. Rate how motivated you were to keep playing and improve (1-5)
5. Rank agents from 1-3 on which gave the best general impression
6. Elaborate your previous ranking
7. Rank agents from 1-3 on which were most challenging
8. Elaborate your previous ranking
9. Rank agents from 1-3 on which employed the most unique strategy
10. Elaborate your previous ranking
11. Rank agents from 1-3 on which were the most frustrating to play against
12. Elaborate your previous ranking
13. Further comments
14. Rate your previous experience with first or third person shooter games (1-5)
15. Rate your previous experience with joystick (1-5)
16. Participant age

The first three questions of the post scan questionnaire was aimed towards gauging the general experience of all the agents. First a question surveying the general liking of playing against the agents where asked, this provided a measure of how generally satisfied the participant was. The general liking or satisfaction can give an indication of how immersed they are in the game. Noticeable difference between the agents served as a metric for the distinctiveness of the AI methods applied to each agent. Their general sense of motivation to keep playing and improve was also surveyed, indicating their proneness to intrinsic reward during play.

Their ranking of general liking towards each agent, can indicate their general satisfaction towards a particular AI. A higher score may also indicate how well balanced that particular agent was in respect to the players skill level. Additionally, AI challenge balance was surveyed by ranking each agent's difficulty, considering the participant's previous experience with first or third person shooter games and joystick controllers.

Novelty was measured by surveying the strategy uniqueness ranks per agent. The novelty that an agent can provide towards the gameplay experience is an important aspect that the NEAT algorithm can facilitate, making this metric one of the foundations to measure the success of research question 3.

Surveying the frustration each participant felt towards the particular agents helps identify AI behaviors that negatively impact the player experience.

4.9 Pre-Pilot technical testing

To conduct the pilot study, all the technical aspects of the project had to be quite final. This is why a pre-pilot technical testing phase needed to be conducted. During this testing phase, the developed controller scheme with the Tehtyx joystick controller and the two buttons on the grip controller was tested out in practice. This way we could assess if they were intuitive enough given the restrictions of the 7T scanner, and possibly further modify how the player controls the game. The joystick in particular needed some fine adjustments in terms of deadzone calibration, as such MR equipment rarely are accurate. The deadzone and differences in movement ranges left to right were adjuster for, so it felt symmetrical during play.

Secondly, it was a chance to test out the eye tracking solution to make sure it was working as intended. In particular testing out the integration between the WebLink software and EyeLink running on the host PC. Unfortunately, not enough time was available to land on a final working setup for the EyeLink eye tracker in time for the experiments. Resulting in using the first week of the experiments to reach a working setup in parallel.

4.10 Analysis

Before analysis on the game data logs could be performed, the data had to be cleaned properly. First the data logged before the logged TTL signal (which is in the form of a "S" keypress sent from the TTL system) were removed. Then, since the produced data from fMRI scans operate with elapsed time from start and not timestamps, the Unix timestamps were converted to elapsed time since the TTL signal. All the logs were limited to a elapsed time of eight minutes or 480 seconds, which was the actual scan time per agent. This removed unnecessary logs produced by the game running longer than the fMRI scan actually lasted. To keep track of the participants data, every participant was assigned a unique ID for their respective data points. The IDs ranged from 0-12 for the 13 participants.

4.10.1 Extraction of relevant game log features

The features logged in the game logs, laid the foundation of extracting new features from aggregating and analyzing the original features in the logs. First the ratio between hitting the opponent and getting hit by the opponent was extracted per participant. This could give an indication of the challenge level per agent each participant played against. Win rate was extracted by calculating the percentage of games won to the games played, this was also a great indicator on game challenge level. Shot accuracy was calculated for both participants and the agents, which were the ratio of shots that dealt damage to shots thrown. Additionally, the average number of balls the participant had during play against a certain agent was extracted.

4.10.2 Aggregating and analyzing results from questionnaires and game logs

In order to analyze the game log data with the questionnaire data, all the game log extracted features were collected into a big dataset together with the data from the questionnaire. The questionnaire data consisted of the participants rating of the agents on given metrics, which was performed during the scan, and the final ranking of each agent performed after the scan. Participant anonymity was kept by assigning a participant ID to the data belonging to each. In the dataset, each agent ID was one hot encoded for feasibility of further analysis.

The first three participants were removed from the correlation analysis due to incomplete data. Each of them did not play against one of the agents, where each agent was left out once from the first three experiments. The correlation analysis was then performed to analyze the linear relationships between variables. Additionally the chi squared test was used to assess the significance of the most important finds.

4.10.3 fMRI image acquisition

A 7T Siemens scanner and a 32 channel head coil (Siemens AG, Erlangen, Germany) was used to aquire the Functional and anatomical MR images. The T2*-weighted fMRI images were acquired using a 2D echo-planar imaging pulse sequence that covered the whole brain (1.5x1.5x1.5 mm3 voxels, TR = 1450 ms, TE = 17 ms, flip angle = 64°). More accurate anatomical information was obtained by acquiring a T1 weighted image with 0.75x0.75x0.75 mm3 voxels.

4.10.4 fMRI preprocessing

The preprocessing of the fMRI data was conducted using fMRIprep [82] [2], and a Nipype based tool [3]. This preprocessing included motion correction, a susceptibility distortion correction (by comparing the fMRI data, acquired in the anterior-posterior phase encoding direction, with that of an fMRI image acquired in posterior-anterior phase-encoding direction) [83], linear registration of the fMRI data to the anatomical T1 image, and non-linear registration of the T1 image to MNI standard space [4].

4.10.5 Statistical fMRI analysis

To extract valuable data from the fMRI scan, different analysis methods were used. One of the methods was voxel based whole brain analysis. Such an analysis could reveal areas of activation troughout the brain without being restricted to

[2]http://fmriprep.readthedocs.io/en/stable/index.html
[3]http://nipype.readthedocs.io/en/latest/
[4]http://www.bic.mni.mcgill.ca/ServicesAtlases/ICBM152NLin2009

a particular area. This method captures peak activation throughout the brain, while giving data on the extent of the activated brain regions. Coordinates of voxels which is activated are captured, with their magnitude. However with this method, smaller but important brain regions can have significant activation while not exceeding the voxel treshold, and thus not ending up in the results.

In addition to the voxel based analysis, region of interest (ROI) analysis was performed. ROI analysis enables capturing activations in smaller important brain regions related to flow and games which could otherwise drown in voxel based whole brain analysis. The regions of interest for this project were chosen to be the cerebral cortex, amygdala and nucleus accumbens.

The statistical analysis of the fMRI data was conducted using FSL 6.0 and a general linear model framework. First, the individual data were spatially smoothed using a Gaussian kernel of 6 mm. Then, within each participant and for each of the five runs including the three agents and two baselines which involved playing against the stationary agent, a model was defined with one explanatory variable that reflected active play against an agent (Active) and the other explanatory variable the short break between game end and the start of the next game (Rest). The contrast created for this model was Active > Rest for comparing the neuronal activity.

Each agent were investigated individually by comparing the agents pairwise. The relationship between behavioral scores and brain activation across agents were investigated using the program Randomise to test for significance non-parametrically [84]. Individual agent analysis were performed by investigating the Active > Rest contrast, while the pairwise agent comparisons were done by comparing neuronal activity for NEAT vs. FSM, NEAT vs. MA-POCA, and FSM vs. MA-POCA. The agents were compared by creating a repeated-measures model with one explanatory variable of interest that compared the two agents across subject, and one explanatory variable for each of the subjects included in the analysis to remove subject mean effects.

Finally, to test for the the relationship between the behavioral data and fMRI data across agents, a repeated-measures model for each behavioural measure (questionnaire rankings and gradings) was defined. These models included the variable of interest that was the average agent-specific behavioural score for each subject, a second explanatory variable that removed the mean effect across all the agents, and then one explanatory variable for each of the subjects included in the analysis to remove subject mean effects.

Cluster mass statistic was used to evaluate significance [85], with $p = 0.005$ as the cluster forming threshold and clusters considered significant at $p = 0.05$, corrected for multiple testing. Uncorrected results were investigated when comparing agents pairwise, due to the exploratory nature of this study with a small number of participants.

For all the models, analysis was conducted for every gray matter voxel in the brain, with gray matter being defined by using the Harvard Oxford Structural Atlases and the MNI-fnirt cerebellar atlas (probability threshold set to 50 percent) [5]. A region of interest (ROI) analysis were also conducted of the amygdala and accumbens, as these regions were of particular interest based on the findings in previous studies. The ROIs were defined by using the Harvard Oxford atlas as

[5]part of FSL; http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Atlases

described, and the average activation was extracted for all the voxels showing a positive effect in each of the ROIs per individual agent and the contrast Active > Rest.

4.10.6 Analysis of effect size for grouped averages using Cohen's D

Due to the grouping of participants favouring particular agents cannot result in large enough groups as the sample size is 13, Cohens D effect analysis was performed.

Cohens's D is a standardized measure of effect size that quantifies the difference between the means of two groups in units of standard deviation. Its calculated as the difference between the means of two groups, normalized by the pooled standard deviation of these groups. Let $\bar{x}_1$ and $\bar{x}_2$ be the means of the two groups, and s_p the pooled standard deviation of the two groups. The formula for calculation Cohen's D is given as:

$$d = \frac{\bar{x}_1 - \bar{x}_2}{s_p}$$

Let s_1 and s_2 be the standard deviations of the two groups, and n_1 and n_2 the sample sizes of the two groups. The pooled standard deviation of two groups is calculated as:

$$s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

The resulting Cohen's d value can be interpreted as follows:

- Small effect size: 0.2
- Medium effect size: 0.5
- Large effect size: 0.8

CHAPTER
FIVE

RESULTS

In this chapter, we will present the study findings, including measures of brain activity and self-reported data obtained from the fMRI study. Additionally, results from the NEAT evolution process is presented.

5.1 Results of NEAT evolution

The results from the NEAT evolution shows how the agent progressed in terms of fitness, but also presents how the agents speciated during evolution. We also present a snippet of the topology of the NEAT agents neural network to explain its complexity.

5.1.1 Evolving CTRNN genomes

The final most successful training setup with the custom configuration as discussed in section 4.5.3.1 and the final training set up with the full size map and six balls resulted in some interesting finds.

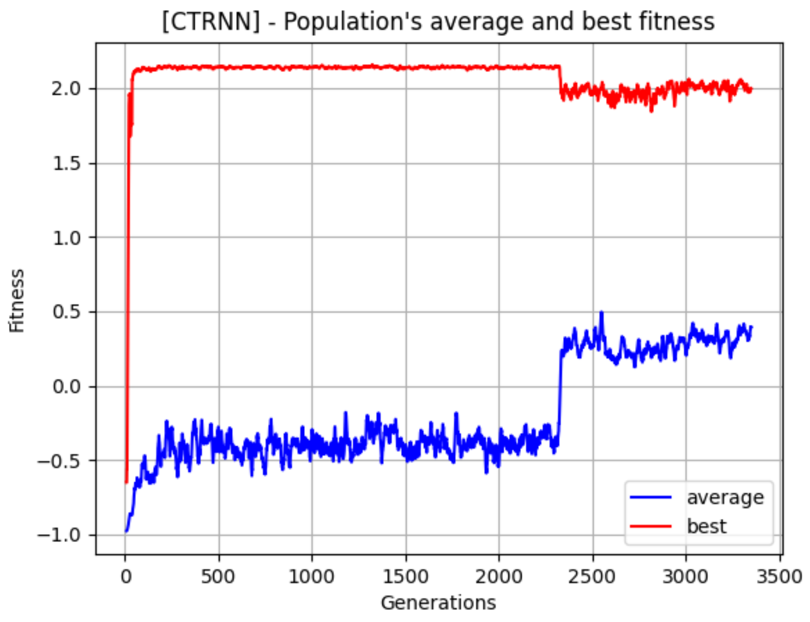


Figure 5.1.1: CTRNN Fitness progression during evolution after 3350 generations. The best genome from the first training phase of 2324 generations were subjectively analyzed and picked to be set as the new challenge for the next training phase until 3350 generations.

As seen in Figure 5.1.1, progression in fitness was significant until 150-200 generations. Following this, the fitness progression seemed to stagnate, which was in line with the previous experiments that were run. Further training had no significant advances in fitness. However, by subjectively evaluating the stored best genomes, advancements in agent behavior could be observed.

During the training, only the initial two species were spawned as seen in Figure 5.1.2, with no new species emerging. During other testing, this seemed to be somewhat random. The network size and complexity were sufficiently evolved after 3500 generations, which gives room to enough genetic diversity allowing multiple species to spawn based on the genetic distance metric. Speciation is a key mechanism in NEAT that can help the search move out of local optimums, which makes the low number of species unusual.

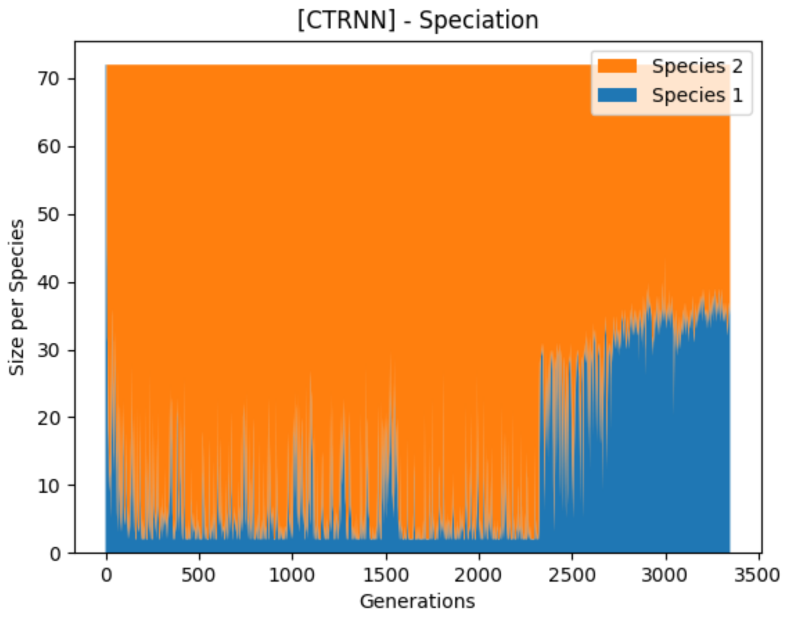


Figure 5.1.2: CTRNN Species count during evolution after 3350 generations. Only two species competed against each other during evolution. The spike at 2324 generations is due to replacing the MA-POCA as challenge with the best genome up until that point. Species 1 represented solutions which fared better against the new challenge.

However, possibly due to recent mutation or major exploration during the search, some "best" genomes at later training stages were not optimal. As seen in the research by Reeder et al. [52], this phenomenon is not uncommon for such evolution of game agents. Even though some agents had trained significantly longer than others, it did not necessarily imply improved behavior.

Initially the ANN-based agents were supposed to be included in the experiments, but due the issue discussed in section 4.5.3, there were no adequate results. The ANN genomes did not evolve to exhibit remotely intelligent behavior, leading to its exclusion from the experiment. Due to time constraints at the 7T fMRI machine, this decision were beneficial. The CTRNN genomes were superior in their performance in comparison, where the agents had learned to achieve a great shot accuracy in addition to exhibiting interesting movement and behavior patterns.

5.1.2 Selection of the most interesting genome

For the fMRI experiment, only one genome from the CTRNN training could be chosen as the means for comparison. Since the quality of the behaviour traits for the agents controlled by each genome could not be directly deduced from the obtained fitness, the selection was performed subjectively.

After the training session until 3350 generations had passed, there were a total of 67 genomes of which behavior and performance could be evaluated. It quickly became apparent which periods of training did not result in adequate behavior. Genomes resulting from the training until 2000 generations and beyond 3000 generations exhibited poor behavior. However, during the generational interval between 2000-3000, quite interesting genomes were observed. By investigating Figure 5.1.1, its not clearly evident that certain intervals should perform worse than others. The spike in fitness was due to swapping the challenge, apparently to a less challenging opponent.

The mentioned promising genome ranges were played against, in a tournament style fashion where the worst genomes were eliminated one by one. The most promising genomes were from generation 2350, 2500, 2800 and 2950. Finally, the best genome from generation 2500 were selected as the winner amongst the promising genomes.

5.1.3 Analyzing the chosen genome

The genome as a result of 2500 generations of evolution, had a network size of 680 nodes. A example of the resulting CTRNN can be seen in Figure 5.1.3, see appendix Section A for the whole CTRNN. Out of the 364 observational inputs to the model, 131 of the inputs were pruned from the network due to no significance. Due to the nature of sensor observations within unity, every RayCast sensor had an additional boolean for each ray flagging whether or not the ray reached its full length without any hit. This summed up to 83 inputs, leaving 48 inputs to be without significance towards achieving a better fitness. Subtracting the 131 non-significant inputs left 233 significant inputs from RayCasts and environment data. The renderings of the CTRNN illustrates the weight of a particular connection and the strength of that weight between two nodes. A positive weight is illustrated as a green line, negative as red and the strength as the thickness of that line. The input nodes, being the entry point of observations from the dodgeball environment, is illustrated as grey rectangles, while hidden nodes are white circles, and output nodes are denoted by blue circles. The number on each hidden node denote the innovation number, which is unique, while the output nodes is numbered from 1-4 with the 0th output node denoted as 'control'.

Visually inspecting the resulting recurrent neural network, the recurrent connections were not apparent. Analyzing the network by checking for cycles, there was only found one enabled recurrent connection, where the output went into the same node. This node were pruned from the main network resulting in no influence on the output. For the disabled connections two additional recurrent connections were found. This could have been due to recurrent connections not having a significant positive contribution towards achieving a high fitness, or a issue with the specific implementation of NEAT. However, since recurrence existed, the mechanism for mutating and adding recurrent connections were working. A interesting find was a particularly strong positive direct connection between a input node and the action node controlling the 'shoot' action (Figure 5.1.4). This suggests a triggering of the 'shoot' action upon a AgentRaycast hitting the opponent and returning a boolean flag of 1. The complete ordered list of actions can be seen in Section 4.3.1.

5.2 Analysis of logs and questionnaires

After the experiments were successfully performed, an abundance of data were available for analysis. Firstly the logs contained all the game events that occurred in the game, with multiple metrics giving an idea on the state of gameplay.

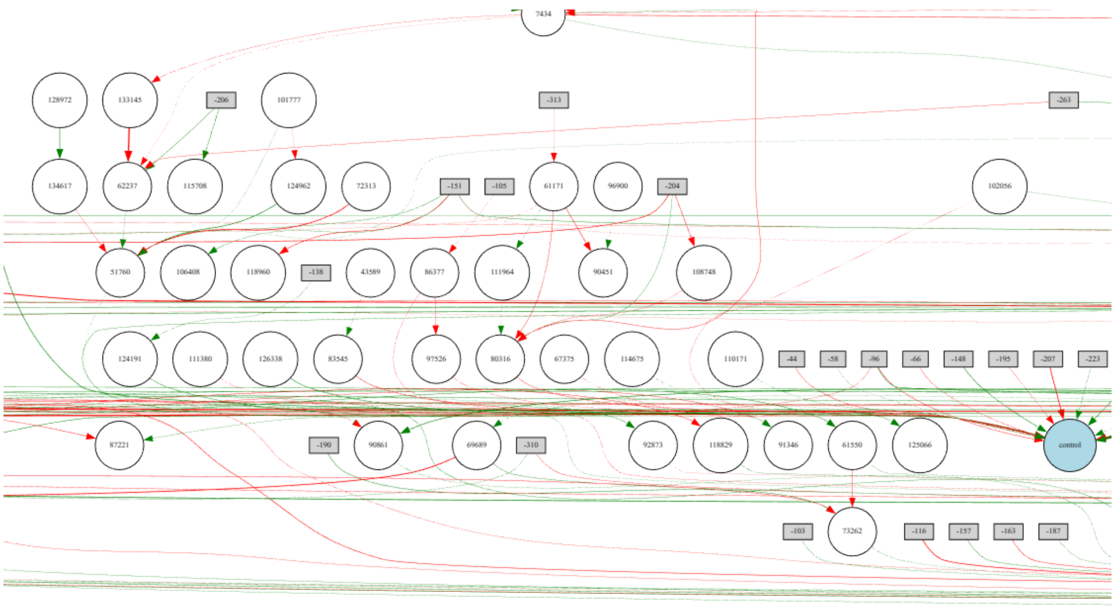


Figure 5.1.3: Smaller extract of the large CTRNN neural network showing enabled connections. The grey rectangles denote the input nodes, the white circles denote hidden nodes and the blue circle denote output nodes. In this example, the 'control' node is the first action output being vertical movement.

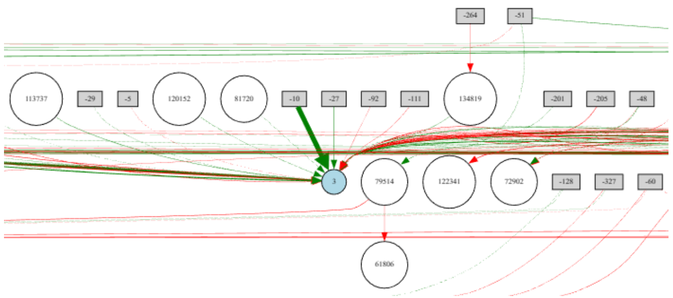


Figure 5.1.4: A particularly strong positive connection in the network directly between a input node and the third action being 'shoot'.

Secondly the questionnaires gives valuable insights in the subjective experience of the participants, rating and describing the agents on many different metrics.

5.2.1 Logs

The data registered during playing the game was stored in the form of logs, one for each agent that a participant played against. Analyzing the data by aggregating it into a global average (as can be seen in full detail in Table 5.2.1) per agent, revealed poor performance for the MA-POCA agent. The MA-POCA agent had the lowest accuracy at 20.0%, while having a player win percent of 80.0% indicating that the agent did not give sufficient challenge to the participants. The NEAT and FSM agents both had similar accuracy, where NEAT had 40.8% while FSM had 42.8%. Interestingly the player accuracy had a inverse correlation with the enemy accuracy, indicating that players feeling more threatened missed more shots due to the risk of getting hit. The players against MA-POCA had the highest average accuracy, while players against FSM had the lowest accuracy. NEAT and FSM were more balanced than MA-POCA, in terms of being closer to a 50% win-rate. Players against NEAT won 64.2% of time time while players against FSM won 35.3% of the time, indicating that FSM provided more of a challenge than NEAT, with both being almost equally far off the 50% balance target. Analyzing data from the participants which played against all three agents revealed that MA-POCA always had the lowest shot accuracy against players. Both FSM and NEAT had the best accuracy 5/10 matches.

5.2.2 Questionnaires

Analysing both the questionnaires answered both during and post fMRI scan, gave an impression of the participants experience.

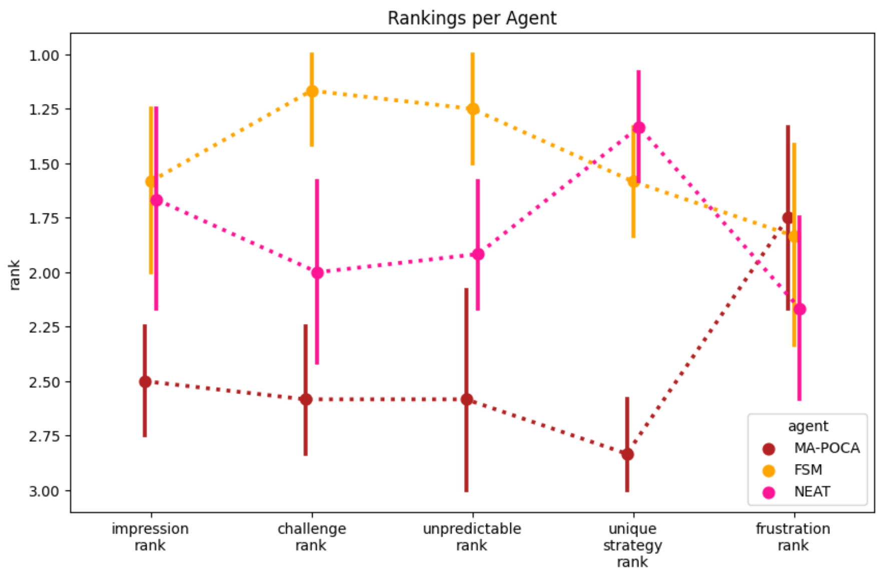


Figure 5.2.1: The rankings the participants gave to each agent post fMRI scan. Represented as a point plot, where the point is the mean, while lines denote the uncertainty. Due to the low participant number, this uncertainty is relatively high

The analysis plot of the ranks given (see Figure 5.2.1) to each of the agent visualizes how the FSM agent significantly beats NEAT and MA-POCA in challenge and unpredictable rank. The challenge and unpredictability rank is very consistent, where MA-POCA is the least challenging and unpredictable, NEAT is in the middle and FSM is the most. For the unique strategy rank, participants tended to rank NEAT higher than FSM, while MA-POCA were quite consistently ranked last. Interestingly, the ratings of frustration are quite close, where FSM and MA-POCA were rated more frustrating than NEAT.

The grades given to each agent (see Figure 5.2.2) after each gameplay revealed interesting trends. NEAT was rated the highest on liking of the agent, with a average grade of 3.91/5 compared to 3.75/5 for FSM. Furthermore, it received the highest mean grade of 3.91/5 of mastering when winning compared to 3.41/5 for FSM. Participants also preferred the challenge/joy balance of the NEAT agent, giving it a mean grade of 4.16/5 compared to the 3.66/5 for the FSM agent. Lastly, the NEAT agent received the highest mean score for entertainment value with a

Game performance metric			
	MA-POCA	NEAT	FSM
Seconds with EnemyLives = 3	2504	2332	3114
Seconds with PlayerLives = 3	3475	2772	2797
Seconds with EnemyLives = 2	1193	1352	1025
Seconds with PlayerLives = 2	812	1595	1006
Seconds with EnemyLives = 1	1158	1198	696
Seconds with PlayerLives = 1	638	767	1009
Time at 3 health player / enemy ratio	1,39	1,19	0,90
Time at 1 health player / enemy ratio	0,55	0,64	1,45
Average enemy health	2,28	2,23	2,50
Average player health	2,58	2,39	2,37
Seconds with BallsLeft = 4	372	199	303
Seconds with BallsLeft = 3	675	496	618
Seconds with BallsLeft = 2	987	1113	942
Seconds with BallsLeft = 1	1216	1493	1312
Seconds with BallsLeft = 0	2057	1956	2045
Average Ball Count	1,26	1,14	1,20
Seconds in corner	584	58	459
Player took damage	164	169	324
Player hit enemy	376	237	217
Ratio took damage / hit enemy ratio	0,44	0,71	1,49
Player Throws	957	814	806
Enemy Throws	811	414	757
Enemy Shot Accuracy	0,20	0,41	0,43
Player Shot Accuracy	0,39	0,29	0,27
Player Ball Pickups	1084	911	931
Environment Resets	141	96	133
Player Wins	112	61	47
Player Loss	28	34	86
Win to loss Ratio	4,00	1,79	0,55
Player Win % of matches played	80,0	64,2	35,3

Table 5.2.1: Game performance data averages for all participants per agent.

mean grade of 3.75/5 compared to 3.66/5 for FSM as well as fitting challenge with a mean grade of 3.83/5 compared to 3.75/5 for FSM. However, the participants rated FSM as the most interesting agent, with a mean score of 3.83/5 compared to the score of 3.5/5 for NEAT. MA-POCA consistently scored the lowest both during scanning and when comparing all agents.

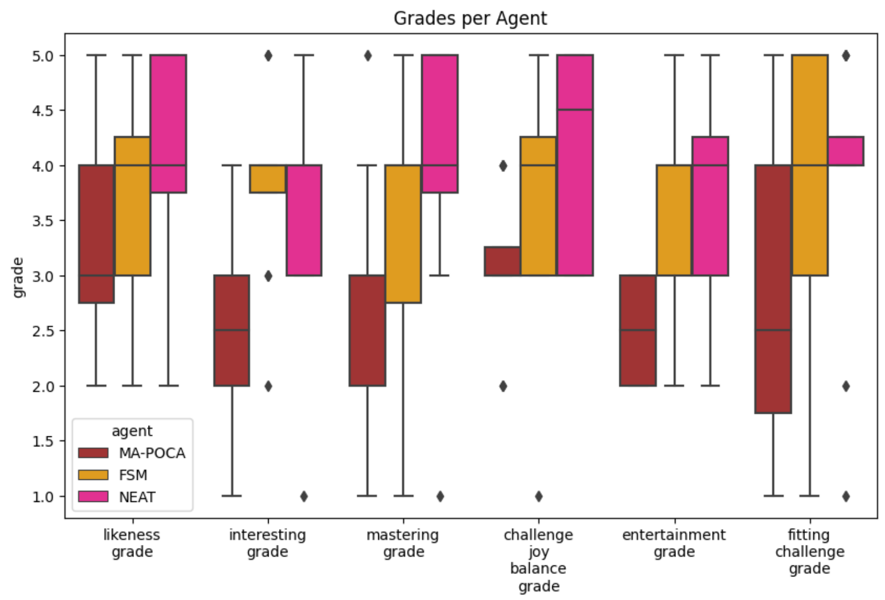


Figure 5.2.2: The grades given to each agent on the basis of different metrics after playing against each. Diamonds indicate outliers outside 1.5 times the interquartile range of the data

The ratings from the questionnaires both asked during the fMRI session and after where quite consistent. The average rating the participants gave for the different aspects of the agent during gameplay, matched with their highest ranked agent for having the best general impression. Participants only favoured FSM and NEAT agents. Seven participants ranked NEAT the highest generally, while six ranked FSM as the highest. These two participant groups were grouped together using pandas, and all the columns were averaged to get a better look at data trends among the groups.

Participants ranking NEAT as their favored agent reported on average a 11.0% higher score on motivation to play further against the agents in general. Additionally reporting a 6.0% higher average score on a clear difference between agents and general liking of all agents.

The participants ranking FSM as their favored agent had generally higher scores on game familiarity and joystick familiarity, indicating that players with more experience seemed to enjoy the more challenging FSM agent as implied from the all time average 64.2% wins by players against NEAT, while players against FSM had a win percent of 35.3%. Participants favoring FSM had a higher average ballcount by 24.0% and 4.3% lower average health.

5.2.3 Correlation analysis

R values above 0.70 are generally considered to represent a strong linear relationship between two variables. With this in mind, the relationships between all the variables gathered from both the logs and the questionnaires were gathered in a single dataset.

Some interesting correlations were found by analyzing the dataset, which can be seen in Figure 5.2.3. For the full correlation matrix with Pearson significance, see Appendix Section E.0.1. First it was found that when the participants played

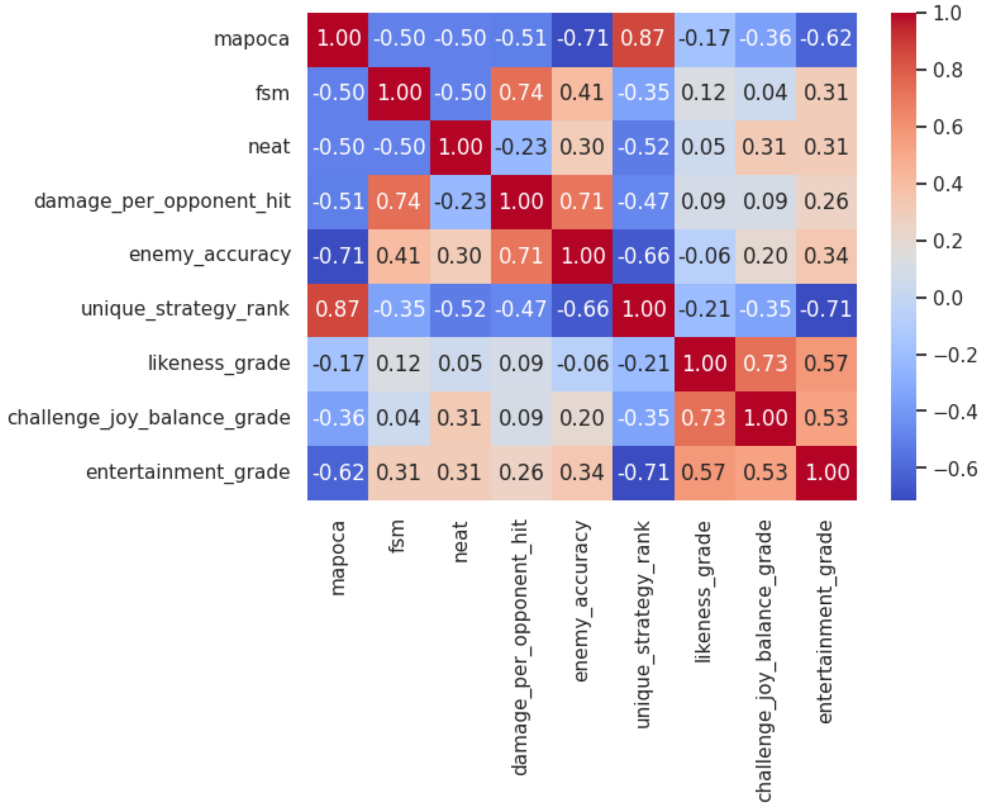
Parameter	NEAT avg	FSM avg
enemy_health	2.210526	2.266471
player_health	2.413158	2.315294
ball_count	1.075263	1.334118
damage_per_opponent_hit	0.991579	0.951176
enemy_accuracy	0.353684	0.344118
player_accuracy	0.320526	0.330588
win_pct	61.444211	59.087059
took_damage	16.736842	20.294118
player_throws	73.578947	74.176471
impression_rank	1.894737	1.941176
challenge_rank	1.894737	1.941176
unpredictable_rank	1.894737	1.941176
unique_strategy_rank	1.894737	1.941176
frustration_rank	1.894737	1.941176
likeness_grade	3.526316	3.764706
interesting_grade	3.157895	3.352941
mastering_grade	3.473684	3.294118
challenge_joy_balance_grade	3.526316	3.764706
entertainment_grade	3.157895	3.470588
fitting_challenge_grade	3.473684	3.352941
game_familiarity	3.526316	3.941176
joystick_familiarity	1.105263	1.647059
age	23.052632	24.470588
experience	3.684211	4.294118
general_liking	4.421053	4.176471
clear_general_difference	4.789474	4.470588
motivated_to_play	4.263158	3.823529

Table 5.2.2: Grouped averages for the participants favouring FSM or NEAT. None of the participants favoured MA-POCA, hence its absence.

against the MA-POCA agent, they rated it to have a low unique strategy with a correlation of 0.87. In addition playing against the MA-POCA agent correlated strongly with a worse opponent accuracy with a correlation of -0.71. In games where the participant faced the FSM, the number of times the opponent hit the player in relation to times the player hit the opponent was greater with a correlation of 0.74.

An interesting positive linear relationship in the questionnaire data was found between agents rated with a higher entertainment score and having a unique strategy. This relationship had a correlation of -0.71 due to comparing ranking with rating, where the ranking goes from one to three where one is the best. The chi-squared score between these two variables results in a value of 0.01 which is statistically significant.

Participants giving a higher rating to an agent for the level of balance between joy and challenge, simultaneously gave a high rating on how much they liked that particular agent. This with a correlation of 0.73. Calculating the chi-squared score results in a p-value of 0.002 meaning this correlation is statistically significant.



	mapoca	fsm	neat	damage_per_opponent_hit	enemy_accuracy	unique_strategy_rank	likeness_grade	challenge_joy_balance_grade	entertainment_grade
mapoca	1.00	-0.50	-0.50	-0.51	-0.71	0.87	-0.17	-0.36	-0.62
fsm	-0.50	1.00	-0.50	0.74	0.41	-0.35	0.12	0.04	0.31
neat	-0.50	-0.50	1.00	-0.23	0.30	-0.52	0.05	0.31	0.31
damage_per_opponent_hit	-0.51	0.74	-0.23	1.00	0.71	-0.47	0.09	0.09	0.26
enemy_accuracy	-0.71	0.41	0.30	0.71	1.00	-0.66	-0.06	0.20	0.34
unique_strategy_rank	0.87	-0.35	-0.52	-0.47	-0.66	1.00	-0.21	-0.35	-0.71
likeness_grade	-0.17	0.12	0.05	0.09	-0.06	-0.21	1.00	0.73	0.57
challenge_joy_balance_grade	-0.36	0.04	0.31	0.09	0.20	-0.35	0.73	1.00	0.53
entertainment_grade	-0.62	0.31	0.31	0.26	0.34	-0.71	0.57	0.53	1.00

Figure 5.2.3: Most significant and interesting correlation finds. The heatmap scale is the R value of each correlation.

Surprisingly, the self-proclaimed skill level, age, joystick familiarity, and game type familiarity of the participants had negligible correlations with most of the other variables. Some weaker correlations were an increase in the average ball count with greater familiarity with the joystick (R = 0.68).

The experience variables were as expected correlated with each other, where general gaming experience had a high correlation with first and third person game experience with a correlation coefficient of 0.85. The familiarity of the joystick was weakly correlated with the experience of first- and third-person games with R = 0.65 and with general game experience R = 0.68.

Gameplay experience also tended to weakly influence the rated liking and interest level of agents with a correlation of R = 0.55 and R = 0.50 respectively.

As participants only preferred the FSM and NEAT agent, the participants were grouped into two groups and the averages were analyzed. Cohens D effect analyzis were used to assess the significance and effect size of the relationship between a participant favouring a given agent and other metrics. The recorded game metric of ball count per participant, had a Cohens D value of -0.913 with a pooled standard deviation of 0.28332. This effect size is quite large, which makes the fact that participants preferring NEAT on average had significantly lower ball count than participants preferring FSM. A sample size of 64 participants would be required to achieve 80% power at an alpha level of 0.05.

5.3 Eye tracking results

Unfortunately, there were some issues with the eye tracking system that manifested during the experiments. Some were implementation specific problems which were not discovered during the pilot study. This led to no results from the eye tracking during the first week of scanning, but continually iterating on the configuration for a working setup. The entire setup was working for the second week of experiments. At the final days of the first week, the setup was working but the focus of the eye tracker did not work, resulting in no useful data. Some results and key takeaways from the process of performing eye tracking is elaborated in Appendix Section H.

5.4 Results of fMRI evaluation

By investigating the results achieved through the fMRI study, we can compare the different agent's performance by looking at brain activity during play. To properly and objectively look at the full picture of brain activity during play, the data will be presented mainly by looking at the entire brain. We also present data mainly by looking at mean activity and voxels in relation to baseline activity during the idle period between game end and start. A voxel refers to a volume pixel, which is the smallest distinguishable box-shaped part of a three-dimensional image. The choice of how to present the data is discussed further in chapter 6.3.

5.4.1 Neuronal activation across agents

All the activated brain regions were compared between all the agents. The common activated networks can be seen in Figure 5.4.1. The figure shows brain activation across the grey matter of the entire brain. Some regions of the brain are activated during play with all of the agents, while some regions have almost exclusive activation from particular agents.

For a detailed view, all brain regions activated per agent, the hemisphere, cluster number, t-value and coordinates are listed in Appendix section D. The voxel based whole brain analysis revealed that across all three agents, both the lateral occipital cortex (LOC) and superior parietal cortex (SPC) showed positive activations. This suggests that these brain regions are actively engaged during gameplay regardless of the opposing agent. The LOC and SPC brain regions are associated with visual processing and spatial attention/coordination respectively.

The FSM was revealed to induce the most brain activity out of all the agents, implied by the greater t-values and voxel cluster size. Greater t-value indicates a greater significance in the neuronal activation. Voxel cluster size is no indication to the significance of the activation, but rather how big part of the brain is activated, as seen in the figure 5.4.1. The FSM agent induced the most activity in the LOC and superior parietal cortex, suggesting that the FSM agent was the most challenging agent, and also possibly more engaging to play against.

The box plots (presented in Figures 5.4.2, 5.4.3 and 5.4.4) show mean activation in whole brain or ROI. The baseline plot shows that play was significantly more interesting and provided more stimuli when faced with an opponent. The fact that baseline2 always displays lower activations than baseline1 suggests that players have mastered the game more when playing the last baseline (baseline2), which

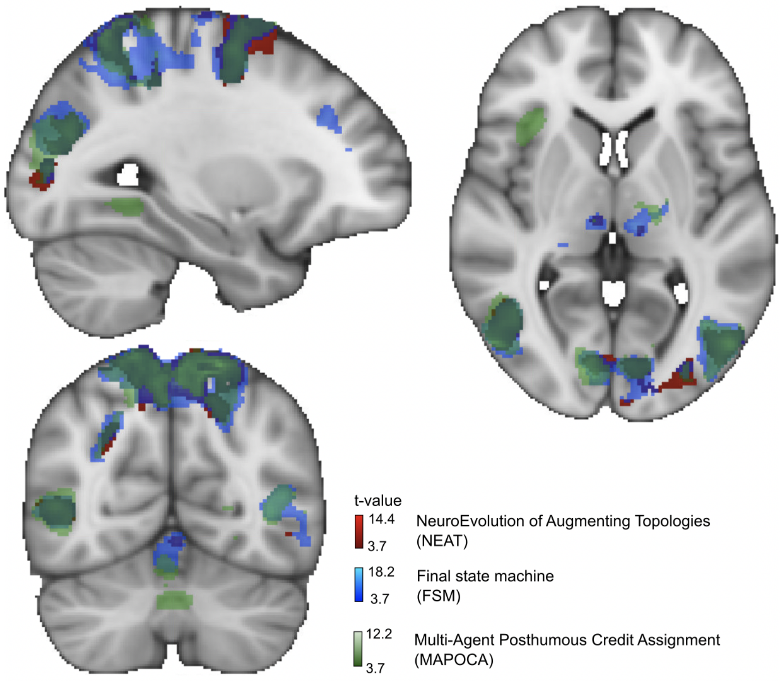

t-value
14.4
3.7
NeuroEvolution of Augmenting Topologies (NEAT)
18.2
3.7
Final state machine (FSM)
12.2
3.7
Multi-Agent Posthumous Credit Assignment (MAPOCA)

Figure 5.4.1: Common brain network activations across all agents. Corrected for multiple comparisons across brain voxels, as there is done one permutation based t-test for each voxel in the brain. The images display different viewpoints of the brain: the top left shows a right side view, the top right shows a top view, and the bottom image shows a rear view.

was always played at the end of a study, while the first baseline (baseline1) was played against at the beginning. It also suggests that the novelty of the game has worn of slightly, and generates less interest in the player.

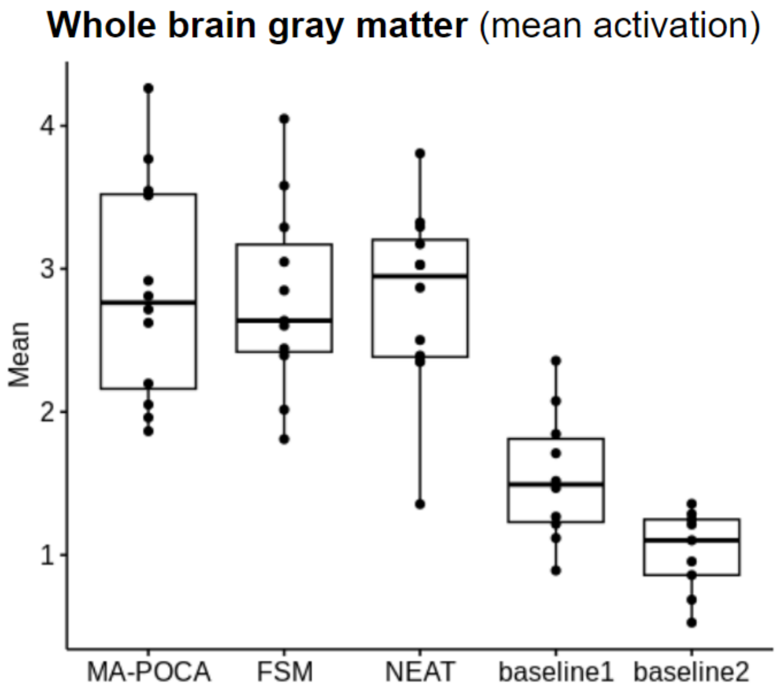
The ROI analysis revealed that participants playing against the NEAT agent exhibited greater mean activation in the amygdala and nucleus accumbens than the other agents as illustrated in Figure 5.4.3 and Figure 5.4.4 respectively. The box plot presented in Figure 5.4.3 displays a higher mean activation during play against the NEAT agent than the other agents. The result highlighted in red displays a significant difference in activation between NEAT and MA-POCA.

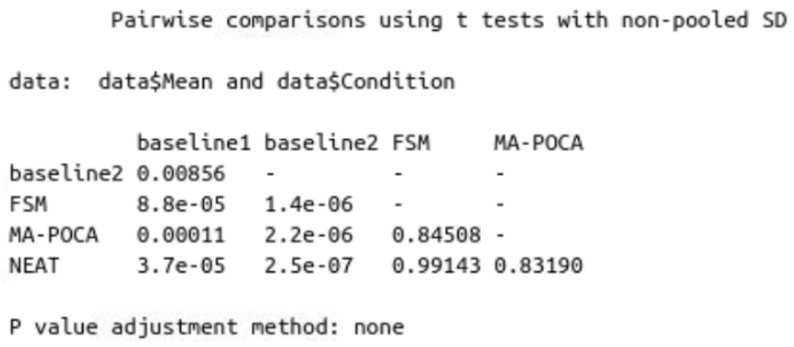
5.4.2 Thirdlevel correlation analysis

Correlation analysis between questionnaire and game data with brain activation revealed some findings as seen in Table 5.4.1. The self-reported grade of experienced mastering during play against the agents had multiple negative correlations. The correlations were the primary somatosensory cortex, primary motor cortex and premotor cortex. This suppressed activation can be indicative of less brain resources needing to be allocated as player skill increases.

5.4.3 Comparing brain activity between the agents

Analyzing brain activity with voxel based analysis on the entire brain reveals some neuronal activity differences between the agents for participants playing





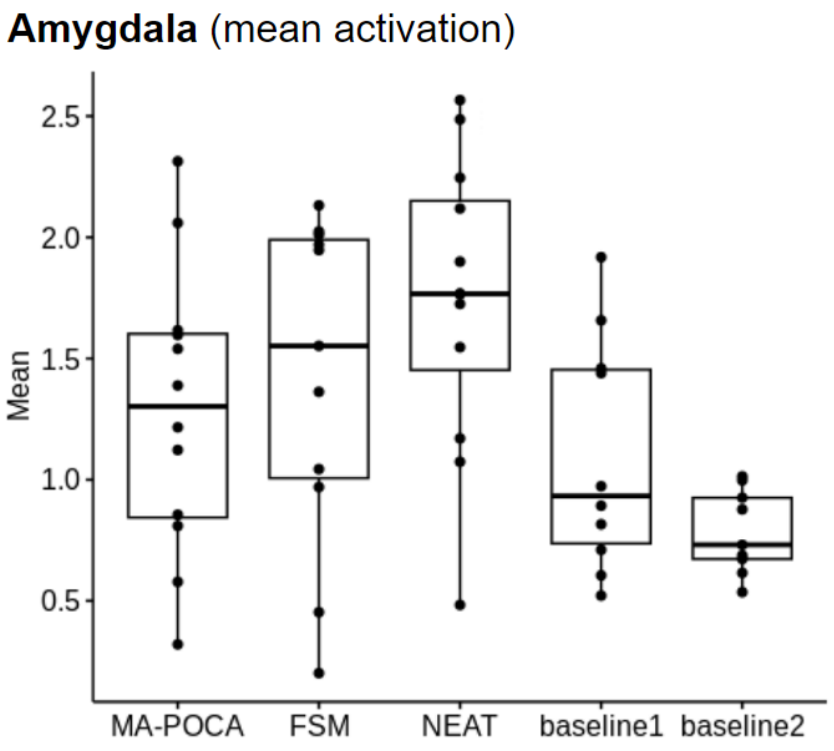
```
        Pairwise comparisons using t tests with non-pooled SD

data:  data$Mean and data$Condition

          baseline1 baseline2 FSM     MA-POCA
baseline2 0.00856   -         -       -
FSM       8.8e-05   1.4e-06   -       -
MA-POCA   0.00011   2.2e-06   0.84508 -
NEAT      3.7e-05   2.5e-07   0.99143 0.83190

P value adjustment method: none
```

Figure 5.4.2: Box plot showing the mean activation of the whole brain gray matter, the matrix in the bottom shows the difference in significance between agents and baseline

against them. Figure 5.4.5 compares the differences in neuronal activity between different agents across all participants. The figure shows increased activity for an agent supporting the hypothesis of increased activation over another agent. Clear differences in the activated areas can be observed for FSM and NEAT over MA-POCA. MA-POCA had no increased activation surpassing the other agents. When observing the increased activation of the NEAT agent compared to the MA-POCA agent, there is a trending increase in activation in the orbital and pre-frontal cortex. The FSM exhibited trending increased activation surpassing both NEAT and MA-POCA. The middle and superior frontal gyruses exhibited particularly strong activation in comparison to the other agents.

Brain region	**Hemisphere**	**C #**	**C size**	**t-value**	**X**	**Y**	**Z**
			————–Mastering Grade————–				
			–Negative effects–				
Primary somatosensory cortex	R	1	624	5.75	55.5	-18	51
Primary somatosensory cortex	R	1	624	5.31	49.5	-18	55.5
Primary somatosensory cortex	R	1	624	4.84	40.5	-16.5	43.5
Primary motor cortex	R	1	624	4.84	39	-22.5	48
Primary somatosensory cortex	R	1	624	4.75	40.5	-19.5	46.5
Premotor cortex	R	1	624	4.46	39	-16.5	60
			————–Took Damage————–				
			–Positive effects–				
Superior parietal cortex	L	1	805	6.97	-21	-61.5	67.5
Superior parietal cortex	L	1	805	6.65	-19.5	-58.5	64.5
Lateral occipital cortex	L	1	805	5.84	-12	-67.5	60
Lateral occipital cortex	R	1	805	5.18	13.5	-64.5	64.5
Superior parietal cortex	R	1	805	5.14	7.5	-63	66
Precuneous cortex	R	1	805	5.09	1.5	-57	63

Table 5.4.1: Correlation analysis between game and questionnaire data with neuronal activity. The mastering grade is a grade from 1 to 5 of how much the participants felt mastering when playing against a particular agent. The took damage game result, is how many times the player took damage during the game from the opponent. There are multiple coordinates listed for the same brain regions due to multiple activations peaks. Brain regions denote which part of the brain the activity was measured in, with hemisphere denoting whether the left (L) or right (R) hemisphere of the brain was activated. C stands for Cluster and is abbreviated to make space for the table. The t-value is the maximal value measured. X, Y, and Z pinpoint the brain coordinates for the peak signal.



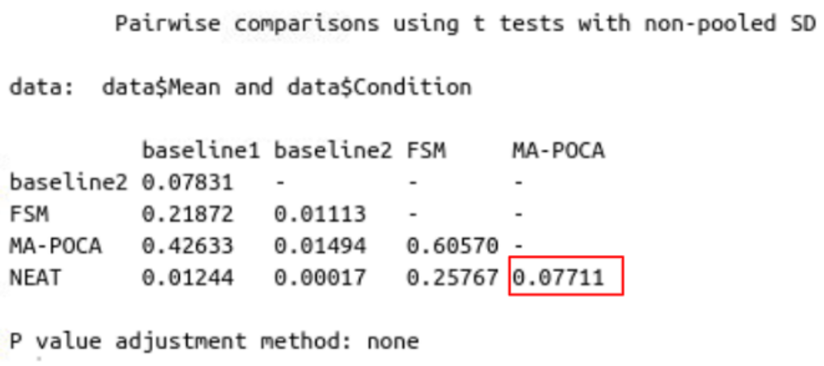
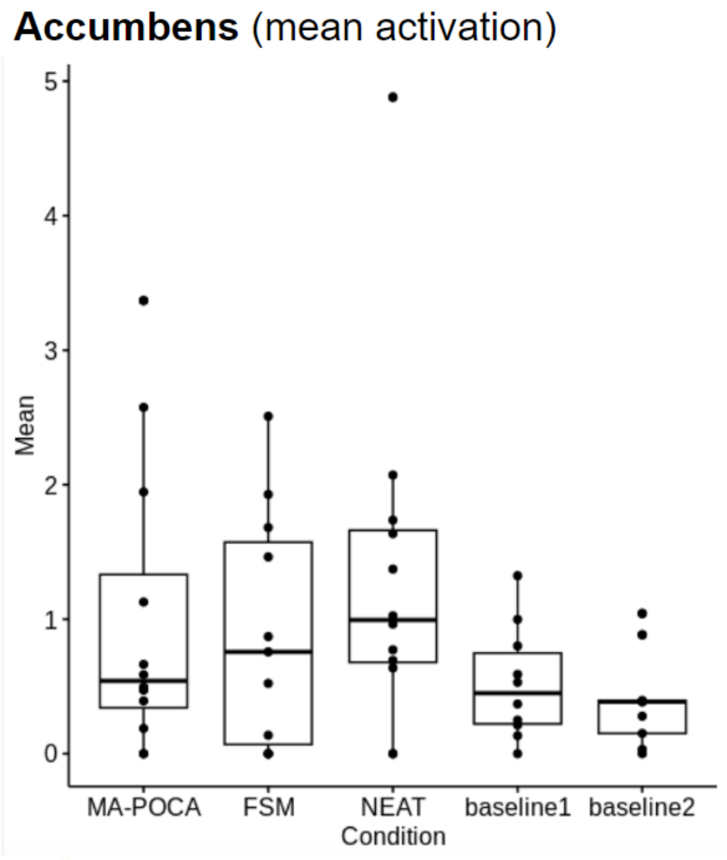
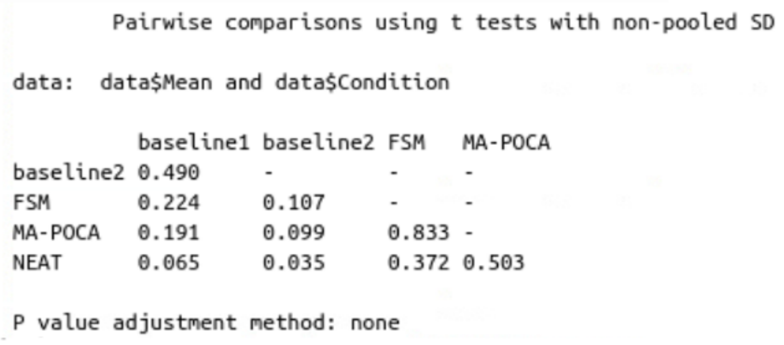

```
        Pairwise comparisons using t tests with non-pooled SD

data:  data$Mean and data$Condition

          baseline1 baseline2 FSM     MA-POCA
baseline2 0.07831   -         -       -
FSM       0.21872   0.01113   -       -
MA-POCA   0.42633   0.01494   0.60570 -
NEAT      0.01244   0.00017   0.25767 0.07711

P value adjustment method: none
```

Figure 5.4.3: Box plot showing the mean activation in the amygdala as a result of ROI analysis, the matrix in the bottom shows the difference in significance between agents and baseline





```
        Pairwise comparisons using t tests with non-pooled SD

data:  data$Mean and data$Condition

          baseline1 baseline2 FSM   MA-POCA
baseline2 0.490     -         -     -
FSM       0.224     0.107     -     -
MA-POCA   0.191     0.099     0.833 -
NEAT      0.065     0.035     0.372 0.503

P value adjustment method: none
```

Figure 5.4.4: Box plot showing the mean activation in the accumbens as a result of ROI analysis, the matrix in the bottom shows the difference in significance between agents and baseline

Brain region	Hemisphere	C #	C size	t-value	X	Y	Z
			—FSM > MA-POCA—				
			–Positive effects–				
Superior_parietal_cortex	L	5	67	6	-19.5	-52.5	57
Superior_parietal_cortex	L	5	67	5.63	-19.5	-54	61.5
Superior_parietal_cortex	L	4	62	6.39	-28.5	-70.5	61.5
Lateral_occipital_cortex	L	4	62	5.11	-33	-73.5	57
Visualcortex_v1	L	3	45	5.5	-12	-87	3
Intracalcarine cortex	L	3	45	5.47	-16.5	-88.5	4.5
Occipital pole	L	3	45	5.05	-15	-91.5	4.5
Lateral_occipital_cortex	R	2	45	4.51	16.5	-75	58.5
Superior_parietal_cortex	R	2	45	4.48	13.5	-75	60
Lateral_occipital_cortex	R	1	42	5.02	24	-76.5	43.5
Lateral_occipital_cortex	R	1	42	4.25	18	-79.5	45
Lateral_occipital_cortex	R	1	42	4	15	-81	43.5
			—FSM > NEAT—				
			–Positive effects–				
Middle_frontal_gyrus	R	7	230	7.04	30	9	60
Superior_frontal_gyrus	R	7	230	5.29	22.5	10.5	52.5
Superior_parietal_cortex	R	6	103	8.46	4.5	-45	66
Precuneous_cortex	R	6	103	5.42	1.5	-48	63
Superior_parietal_cortex	R	6	103	4.43	1.5	-42	61.5
Superior_parietal_cortex	R	6	103	4.15	7.5	-37.5	58.5
Primary_somatosensory_cortex	R	5	98	5.81	30	-39	43.5
Superior_parietal_cortex	R	5	98	4.76	34.5	-36	39
Superior_parietal_cortex	R	5	98	4.29	30	-45	48
Superior_parietal_cortex	R	5	98	4.24	30	-45	52.5
Primary_motor_cortex	L	4	98	8.85	-9	-37.5	60
Superior_parietal_cortex	L	3	97	7.03	-30	-64.5	58.5
Superior_parietal_cortex	L	3	97	5.66	-27	-64.5	55.5
Superior_parietal_cortex	L	3	97	5.21	-24	-69	61.5
Lateral_occipital_cortex	L	3	97	4.89	-31.5	-66	49.5
Superior_parietal_cortex	L	3	97	4.8	-30	-63	64.5
Inferior_parietal_cortex	L	2	86	6.23	-54	-34.5	46.5
Supramarginal_gyrus	L	2	86	5.96	-51	-34.5	42
Premotor_cortex	L	1	56	6.5	-9	-13.5	69
Premotor_cortex	L	1	56	6.3	-7.5	-19.5	69
			—NEAT > MA-POCA—				
			–Positive effects–				
Premotor_cortex	R	4	104	7.41	40.5	-4.5	60
Premotor_cortex	R	4	104	6.3	42	-3	57
Premotor_cortex	R	4	104	5.6	40.5	-3	51
Paracingulate_gyrus	L	3	90	6.54	-15	54	-1.5
Frontal_pole	L	3	90	5.74	-12	58.5	-4.5
Frontal_pole	L	3	90	5.61	-3	57	-4.5
Frontal_pole	L	3	90	5.47	-1.5	57	-1.5
Frontal_pole	L	3	90	4.67	-13.5	61.5	-3
Frontal_pole	L	2	79	6.24	-25.5	49.5	-9
Frontal_pole	L	2	79	5.83	-33	51	-10.5
Frontal_pole	L	2	79	4.32	-34.5	43.5	-7.5
Precuneous_cortex	L	1	59	4.78	-6	-60	37.5

Table 5.4.2: Comparisons of brain activation between agents. The inequality signs $<$ and $>$ denote the hypothesis tested for. There was no negative effects for any of the comparisons. Brain regions denote which part of the brain the activity was measured in, with hemisphere denoting whether the left (L) or right (R) hemisphere of the brain was activated. C stands for Cluster and is abbreviated to make space for the table. The t-value is the maximal value measured. X, Y, and Z pinpoint the brain coordinates for the peak signal.

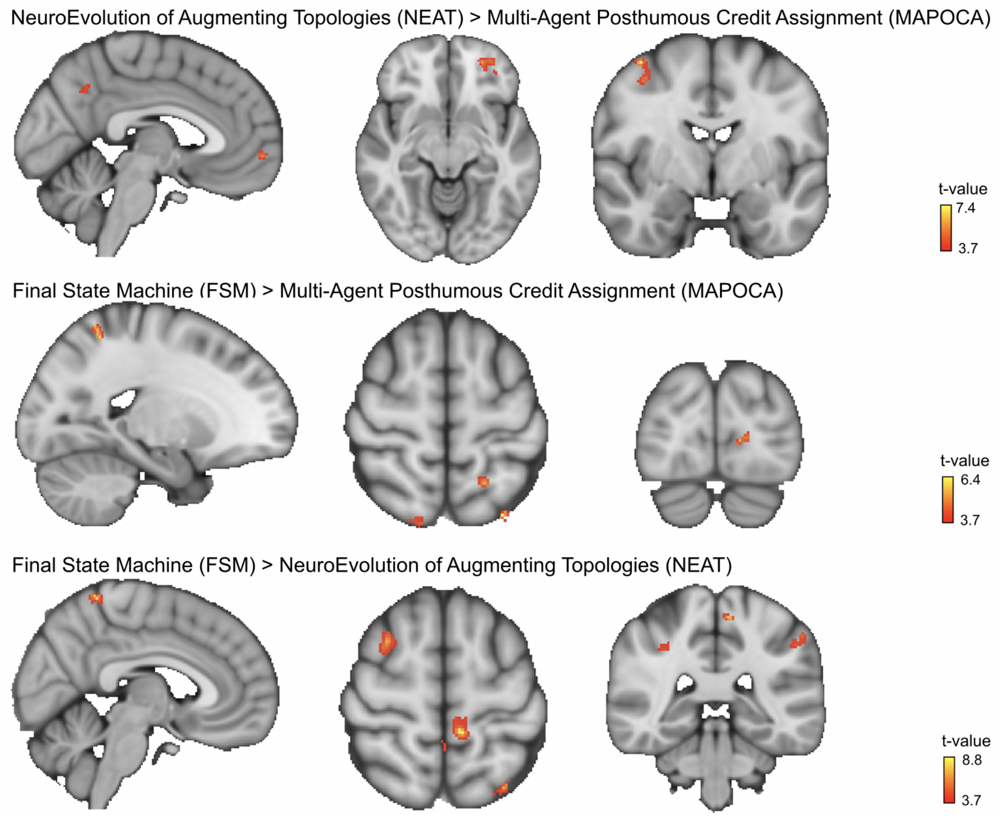

NeuroEvolution of Augmenting Topologies (NEAT) > Multi-Agent Posthumous Credit Assignment (MAPOCA)
t-value
7.4
3.7
Final State Machine (FSM) > Multi-Agent Posthumous Credit Assignment (MAPOCA)
t-value
6.4
3.7
Final State Machine (FSM) > NeuroEvolution of Augmenting Topologies (NEAT)
t-value
8.8
3.7

Figure 5.4.5: Differences in activation between the agents. Not corrected for multiple comparisons (higher uncertainty)

CHAPTER
SIX

DISCUSSION

In this chapter, we will discuss the implications of the results and the limitations of the study. The results of the fMRI measurements will be compared to the answers from the questionnaires, to see whether there is a correlation in brain activity and self reported data. The discussion will also include a comparison of our results with previous studies, and an evaluation of the contribution of this thesis to the field of AI in videogames and the use of fMRI machines in measuring brain activity. Finally, we also provide suggestions for future research, including potential improvements to the methodology and potential applications of the findings.

6.1 Highlights

In the previous chapter, we presented the results from the long evolution of the NEAT agent and the fMRI study conducted with 13 participants. All three agents exhibited diverse behavior, supported by the trending and significant differences in brain activation, self-report data and game data. There were observed trending differences in the whole brain voxel based analysis, but the most interesting findings were observed following the ROI analysis, where the amygdala and nucleus accumbens exhibited clear differences in activation levels.

While common neuronal activation patterns were observed, a more widespread and higher peak neuronal activation were observed for participants playing against the FSM agent, concluded to be due to its more demanding nature supported by the game data and questionnaires. Participants playing against the NEAT agent exhibited a trending increase in mean neuronal activation in the amygdala and nucelus accumbens, two important brain regions in the context of games.

The self-reporting data clearly indicated a preference for playing against the NEAT agent, while the FSM agent also received high scores due to its challenging nature. Together, the higher mean amygdala and nucleus accumbens activity, self-report data and game data, suggests that playing against an agent evolved with NEAT is more engaging and interesting compared to FSM and MA-POCA. The discussion sections below will provide further elaborate upon these findings and their relevance.

6.1.1 Evaluating the effectiveness of agents inducing flow

The aspects of flow discussed in Section 2.9 inspired the developed questionnaire. In order to assess if any of the agents induced a sense of flow, we can look at the self-reporting, performance and fMRI data. Previous studies investigating the relationship between the flow experience and neuronal activity have yielded mixed findings. However, specific brain regions have consistently shown activation during the flow state. Participants playing against NEAT and MA-POCA exhibited substantial activation in the prefrontal cortex, specifically in the superior and middle frontal gyrus in addition to the brocas area, as seen in Section D in the appendix. Some studies found that the prefrontal cortex [72, 63] exhibited increased activation during flow experiences.

The higher activation of the amygdala for the NEAT agent is inconsistent with the findings of Ulrich et al. [62], which argued that the heightened amygdala activation was due to participants experiencing stronger negative emotions. It is possible that the strong feelings experienced by the participants in their study, were mainly negative due to the simple nature of the task. The comparison was done between boredom, overload, and a balanced challenge. The dynamic nature of the dodgeball environment with the diverse agents within could have contributed to stronger positive feelings due to experiencing novelty and learning, which is hard to experience in mental arithmetic tasks. An additional argument can be made that the higher amygdala activation for the NEAT agent is due to the NEAT agent exhibiting a unique strategy, which challenges the participant to learn and adapt in order to counteract it. The novel behavior potentially activating the amygdala could have kept participants immersed and engaged. The involvement of the nucleus accumbems indicates heightened levels of intrinsic reward during play, which can motivate a player and engage them during play. This supports the flow aspect of clarity of goals and immediate feedback. Similar findings were found by Klasen et al. [59], where success states were associated by increased activity in the nucleus accumbens amongst others. Katsyri et al. [65] additionally associated the nucleus accumbens with participants perceived pleasure.

Examining the self-report data from the questionnaires, NEAT emerges as the agent most likely to induce a flow state in participants. It scored notably high in terms of unique strategy, aligning with Csikszentmihalyi's assertion that enjoyment stems from a sense of novelty and accomplishment [36]. Additionally, NEAT received the lowest frustration ratings, suggesting that participants may have felt a sense of control and ease during gameplay. NEAT outperformed other agents in categories directly linked to flow: it was rated highest for both balance between challenge and skill, one of the key criteria for flow, and for a sense of mastery. These findings indicate that NEAT was particularly effective in providing intrinsic rewards and immediate feedback, which are essential components of flow.

In conclusion, the findings suggest that the NEAT agent is more likely to induce a state of flow in participants compared to the FSM agent. This is supported by the lower activity observed in the prefrontal cortex, amygdala, and nucleus accumbens during interactions with the FSM agent. The MA-POCA agent exhibited the most activity in the prefrontal cortex out of all the agents, as can be seen in Section D in the appendix. However, the MA-POCA agent failed to pose participants a real challenge, which is reflected in the participant win rate of 80%, lowest challenge rank, challenge and joy balance, and lastly the participant comments in Appendix Section G. This fails to meet the criteria of achieving a state of flow, particularly the balance between challenge and skill.

6.2 Evolving dodgeball agents using NEAT versus other agents

Training agents with NEAT was not a straight forward process. Success can depend heavily on the configuration and implementation of fitness functions, hyperparameters, reward function and environment design. It was a process to end up with a framework enabling the evolution to happen in a satisfactory manner. After performing experiments on the optimal training strategy, we realized that judging a individuals performance solely on fitness was not going to work. Bjerke [43] also showed a similar progression in fitness like we had. His CTRNN NEAT evolution fitness progression were substantial the first 100-200 generations and then it stagnated by going up and down in waves if training continued for thousands of generations. Reeder et al. [52] also found that longer training times not necessarily implied improved behavior.

The fluctuating and stagnated progression can be explained by the high complexity of the environment. It can additionally be explained by some stochasticity as in dodgeballs which hit other dormant dodgeballs in the line of fire results in a miss by the trajectory completely diverging. Many individuals looking promising in terms of their fitness could have undesirable and unplayable behavior. This is of course some of the drawbacks of the NEAT method, where the high level of exploration can lead to genomes exploiting the weaknesses of the given challenge to obtain a good fitness.

As elaborated in Section 5.1.2, the genome selected for use in the experiment after elimination trials, was the best genome from generation 2500. This genome was the most promising genome exhibiting great shooting accuracy, substantial amounts of movement and exploited game mechanics like picking up balls and dodging incoming shots.

To answer research question 3, through evolution with NEAT, we were able to generate an AI opponent with unique and diverse behaviors, which contributed positively to the gaming experience. This is substantiated by both the questionnaire and the fMRI findings. Participants rated the agent evolved with NEAT highest in terms of liking, sense of mastery, balance between challenge and joy, fitting challenge, and entertainment value. It is noteworthy that the novelty of the behavior exhibited by the NEAT agent was particularly appealing to the participants, as evidenced by their responses. The FSM agent performed comparably in several aspects and was even rated higher in terms of interestingness in addition to challenging the participants the most, inducing substantial neuronal activity. However, the novel behavior of the NEAT agent appear to offer a more enriching experience overall. This conclusion is further supported by the activation observed in the amygdala and nucleus accumbens, areas of the brain responsible for learning, emotional response and reward processing. However, the FSM implementation gave impressive results amongst the participant, arguing that it is still a viable solution for certain standards of game AI even though NEAT offers more value to the gaming experience.

This also affirms our choice of FSM through research question 1, as FSM was a highly relevant agent for this study both in terms of FSMs history in video games, as well as its performance. The MA-POCA agent also proved to be an interesting candidate for comparison, as both the FSM and NEAT agent achieved better overall results than the MA-POCA agent. The nature of the MA-POCA agent, being based on reinforcement learning, provided a proper comparison to advanced AI in games with another learning algorithm. It is worth noting that other types of agents could be explored for future studies, but for the purpose of this study, we believe these agents were very suitable for comparisons.

6.3 Discussing the results of the fMRI study

The results found in Section 5.4 open for some interesting discussions. When discussing the different results from the fMRI study, it is important to note that there are differences in the whole brain analysis, and the ROI analysis data. The whole brain analysis indicates that the FSM agent has higher mean brain activation, both in voxel size and t-value. Observing results from regions of interest such as the amygdala and accumbens show very favorable results for the NEAT agent. This Section will discuss the differences in data and elaborate on our hypotheses regarding the results.

The figures presented in Section 5.4 show clear indications that certain agents are favoured over others. These results also confirms our findings from observations during play, self-reports and performance data from the game. Figure 5.4.5 along with Table 5.4.2, resulting from the whole brain analysis, shows that the NEAT agent induced more neuronal activity than MA-POCA. While the FSM agent induced the most neuronal activity out of all the agents.

Questionnaire analysis revealed that most participants favoured the NEAT agent on average as seen in Section 5.2.2, even though the FSM agent was the one that induced the most neuronal activity in the LOC and SPC as elaborated in Section 5.4.1. The increase in activation of LOC and SPC can be explained by the increase of challenge for the FSM agent. Where the demand for visual processing and spatial awareness were the highest. Participants favoured the challenge of the FSM agent, while favouring the NEAT agent according to self-report on the aspects of liking, mastering, balance between challenge and joy, and entertainment value as seen in Figure 5.2.2. The Cohens D analysis in Section 5.2.3 revealed that players favouring NEAT had a lower ball count on average than players favouring FSM. This reflects the findings that participants favouring NEAT has less gaming experience, and thus has a harder time of exploiting the game mechanics to the fullest.

The general increased neuronal activity for the FSM agent while the NEAT agent inducing the most activity in the amygdala and nucleus accumbens could be explained by the NEAT agent being more emotionally engaging, provocative or rewarding to play against. This is partly supported by the findings of Katsyri et al. [65], where activity in the nucleus accumbens was found to be associated with participants perceived pleasure. However, Ulrich et al. [62] found a decrease of amygdala neuronal activity during flow states which were argued to decrease negative arousal as discussed in Section 6.1.1. The general high FSM neural activity can be explained by its behavior requiring more attention and cognitive resources to interpret or understand. Playing against the NEAT agent might not demand as much brain resources, allowing the brain to focus more on the emotional and reward-processing aspects (involving the amygdala and nucleus accumbens). These findings answers **research question 2**, where the three agents which each exhibited distinct behaviors induced trending differences in neuronal activity. It was not just differences in the strength as mentioned in 5.4.1, but also significant differences in brain regions activated, particularly for NEAT. The neuronal activity in the amygdala and nucleus accumbens can suggest that the agent evolved with NEAT contributes positively to the game by providing a unique behavior increasing motivation and inducing stronger feelings for the player. This is supported by NEAT receiving the highest mean rank for strategy uniqueness, lowest frustration rank and highest rating for both sense of mastering and balance between challenge and joy. FSMs high frustration ranking and lower balance between challenge and joy while scoring the highest for challenge and unpredictability as seen in Figure 5.2.2 and 5.2.1, further confirms that the challenge of the FSM agent were too high for some participants. This was identified by analyzing the game log data in Section 5.2.1, where players won 35.3% of the time against FSM while winning 64.5% of the time against NEAT.

When observing Figure 5.4.1, we can see regional differences in brain activations. Notably, some brain activations are not displayed here, such as the pre-frontal cortex activation displayed in 5.4.1 in the top left brain image. This is due to different slices being presented, and not all local activity is able to be displayed in a single slice. These figures do still show a clear trend in brain activation. The

FSM agent has the highest t-value, with the significant score at a peak of 18.2. We speculate that these scores correlate with the challenge level of the FSM agent.

The results presented in Section 5.2.1 allude to correlations between activation in the amygdala and heightened sense of survival. Participants also rated the FSM as both most challenging to play against and almost equally frustrating to play against as the MA-POCA agent in 5.2.1. We believe this rating of frustration stems from two entirely different perceptions. Many players reported that the MA-POCA agent bored them to the point of frustration in G, while on the contrary the FSM was so frustrating in terms of being too difficult. This could mean that participants have a heightened sense of survival when confronted with the FSM agent, resulting in more significant activation in the amygdala. The significant activation in amygdala when confronted with the NEAT agent could stem from survival as well, but is also likely influenced more by excitement. This correlates to the questionnaire results in 5.2.1 and 5.2.2 where participants consistently rated the NEAT agent as most interesting, balanced, and unique.

When observing brain activity in the occipital lobe (the rearmost part of the brain) Figure 5.4.5, minimal differences in activation are observed. This is where the visual cortex is located, which is the primary area responsible for processing visual information. We argue that the lack of difference in activation in this particular area shows that the study was performed in a thorough and unbiased manner. Having a lack of activation in areas responsible for visual processing means participants processed the same information visually.

FSM scored the highest for challenge in the questionnaires, while the in-game win percentage for participants against FSM was 35.3%. This is consistent with the findings of FSM exhibiting increased neuronal activation revealed by the whole brain voxel based analysis. The increased activation over the other agents in the middle and superior frontal gyruses suggests that the participants were experiencing higher levels of attentiveness, decision making, cognitive control and coordination of movements.

6.4 Limitations

The development phase of this project profited from the preliminary process explained in Section 1.3 performed half a year prior to this project. Spending half a year to perform research, investigate relevant technologies and discussing with peers how to proceed with the project saved several hours of development and planning. In spite of this, the project still suffered from unexpected challenges that proved to be demanding and time consuming. This section will elaborate further on how some of these challenges limited the project and discuss how these could be handled differently or possibly avoided.

6.4.1 Limitations of the eye tracking system

The first, and most evident limitation to this project, is the lack of usable data from the eye tracking. Valuable time was dedicated to make the eye tracking system work, but there is no data to show for this. Through discussions with other scientists at St. Olavs Hospital, we learned that we are not alone in struggling with this system. Some of the issues presented here are specific to the setup at St. Olavs Hospital, but are valuable insights to prevent issues with the system nonetheless.

Firstly, the calibration of the eye tracking system was the source of much frustration when performing the study. In the logs presented in F, it is clear that calibration was an issue throughout the entire study. Along with the radiographers, we made several attempts to fix the calibration issue. By the end of the study, it became apparent that the culprit behind most calibration errors and difficulties was that the screen located behind the bore of the magnetom terra had slanted slightly. This resulted in participants not seeing the lowest of the calibration dots. We had previously tested how much of the screen participants could see during gameplay, though checking to see if players could see the calibration dots didn't cross our minds. A process has been initiated at St. Olavs to adjust the screen so participants see the screen properly. This requires clearance and takes time, since screws and hinges attached to the magnetom terra need to be adjusted. Another way to fix this issue is to look into a narrower calibration sequence, to ensure that players can observe all calibration dots, though this could affect the quality of the eye tracking data.

Ultimately, this had no effect on this project, due to the eye tracking data not being saved properly. In hindsight, we could have investigated whether the eye tracking data was being saved properly and checked the files that were being saved. By attempting to analyze one of the saved EDF files, the faulty recording that stopped once the game were launched could be uncovered. However, it is possible that we could not have solved this fault as WebLink was relatively fresh software and can contain bugs. We could have consulted SR-Research to mitigate the fault before the end of the trial to solve this. Executing the fMRI study was an intense process, which was adjusted to fit inside the time frame several times. Given the very restricted time available in the 7T office, all time was dedicated to executing the study as effectively as possible. To avoid this happening in future work, we would suggest checking the data received from the pilot study.

6.4.2 fMRI study and agent limitations

Since the magnetom terra is a highly expensive and popular device for fMRI research, hours available for booking are limited. Our studies were booked in one and a half hour sessions, meaning there was a limit to the time frame given for each participant. Initially, we estimated that players could play against each agent for 15 minutes, with breaks for baseline and questionnaires in between agents. This was highly optimistic, and after the pilot participants this was quickly changed. Given more time in the fMRI machine, we could potentially allow more play against each agent, or have longer and more frequent baselines throughout the experiment.

During the planning of the fMRI study, we estimated that in the 90 minute long time slot, 45 minutes would be allotted to play against agents. This turned out to be a gross overestimation, due to the many procedures required to prepare and conduct the study, as described in Section 4.6. After three studies, we eventually got the timing right, and the remaining studies were well conducted without exceeding the time allotted. However, the first three studies exceeded the time slot, which led to the first three participants playing against only two of the agents. We include this incomplete data in our results, since incomplete data is still important data, though ideally they should have played the same three agents as all other participants. Since the order of play was shuffled, the three agents have still seen equal amounts of play.

The results from a ROI and whole brain analysis can differ quite significantly. A ROI analysis delimits the area of interest, enabling smaller brain regions activity to be more significant. This is why the accumbens and amygdala were found to have higher activation for NEAT than FSM and MAPOCA, while the whole brain analysis displayed no increased activation for NEAT over FSM. Such small brain regions can disappear on whole brain analysis due to their small number of voxels not exceeding the threshold to be classified as active. Additionally, whole brain peak analysis also picks up stronger neuronal activity from the cerebral cortex, which is the layer on top of the cerebrum, than neuronal activity deeper into the brain. This is due to the cerebral cortex being closer to the head spool used during fMRI. ROI analysis takes the mean of activation of the brain regions analysed, while whole brain analysis captures the peak activation.

The specific implementation of all agents have weaknesses and strengths could be a result of the training, implementation or environment design. The FSM agent was developed for this project, and was intermittently plagued with staying in corners. The MA-POCA agent also had tendencies of staying in the corner, with a total of 584 seconds spent in or close to a corner compared to 459 seconds by the FSM agent. The low popularity of the MA-POCA agent was surprising, as it had undergone substantial training by its creators. However, the fact that it was trained in a team instead of alone might have contributed to its clumsiness in a one versus one environment. The time spent in a one versus one situation during its training may have been limited. Arguably, the skills learnt playing with and against multiple other agents transfers over to such a one versus one situation.

The order of the agents the participants played against was not systematically random, there was not a system for randomization put in place before the experiment. Instead, the order of agents to be played against was shuffled by starting with the previously last agent, then secondly the previously first agent was played against, and lastly the previously second agent was played against. Then when one full rotation had happened, the order was shuffled by swapping the order of certain agents into a order that was somewhat unique.

Another drawback of the fMRI study, was the available controllers. Due to being made to be MR compatible, they were not as fluid and movable as regular gaming controllers or joysticks. This might have contributed to the participants needing additional time getting used to the controller in addition to playing outside the scanner. The participants might also have gotten tired at the end of the trial, even though no remarks about it were made.

6.5 Future work

This section will provide suggestions for further research, elaborating on potential improvements to the methodology and potential applications of the findings.

6.5.1 Further development of the dodgeball environment and the agents

The project has some clear directions where further development or changes could positively impact the results and project as a whole. One such direction is fixing known bugs that were left in due to difficulties in solving them and time being prioritized in other directions.

Through the quantitative data results and observations during the study, the difference in agent difficulty was clear. This doesn't necessarily invalidate anything, but is a point of interest that was not explored through this particular study. By balancing the agents to match each others difficulty level, one could arguably achieve clearer distinction between agent behavior, not clouded by difficulty as an obstacle. Figure 5.2.1 displays that through the questionnaire, participants found the FSM agent to be the most challenging to play against, and found the MA-POCA agent least challenging. NEAT, being the "most balanced agent" while also portraying the most unique behavior, could have achieved its good results by offering good balance. This would potentially put players in a higher state of flow during play against NEAT. Exploring the territory of balancing agents to have the same adjustment of difficulty, while still keeping their respective behaviors is an interesting continuation of this project.

In retrospect of performing the study, there are also more points of data that could provide interesting insights to the agents' behavior. One such data point is to log the movement data of both player and agent. Superimposing the movement data on an image of the game map would give re-traceable data to examine both how players and agents approach different game scenarios. Additionally, observing the distance between player and agent would add to the fMRI brain data. Being close to an opponent can induce moments of panic or frantic behavior, in contrast being very distant from your opponent should be more relaxing. Investigating the brain response in moments where the participant witnesses the agent on screen could also be valuable information, this could be done by logging whether or not the enemy is rendered on the screen. Lastly, logging whether the player hit the opponent in the front or from the back, gives insight into the nature of the situation which might effect brain activity. If the opponent is hit from the back, it suggests that the participant has control over the situation and is not pressed by the opponent, risking a shot in return.

6.5.2 Further refinement of the NEAT evolution framework

The NEAT evolution framework was not perfect, and neither was the NEAT implementation itself. Even though we got satisfactory results from our evolution process, there were some aspects that could see improvement. First and foremost, the opponents used to challenge the genomes during evaluation should have been regulated by an algorithm instead of manually. The genome used as challenge (i.e. the genome used to evaluate the NEAT population against it) should have been picked from the best genomes from a set range of recent generations, either by chance or a set previous generation. A sliding window from earlier generations encompassing recent evolution history with the best genomes resulting from them could be used as the pool from which to pick a new challenge by chance. This could have led to improved results, with a more generalized behavior. Some of the genomes indicated that they had learnt specific strategies to beat certain opponents, which did not translate well in gameplay against humans due to overfitting on certain agents weaknesses. This could have been one of the reasons seemingly high fitness for certain genomes not reflecting into gameplay behavior.

Further experiments on more optimal NEAT configurations is always encouraged, as it might significantly impact the evolution result. Starting out evolution with more hidden nodes could have positively impacted the evolution time, however the bug not allowing this in NEAT-python has to be fixed or a different NEAT library must be used to test this.

More significant experimentation with evolution can be done by testing the performance of applying multi-modal NEAT by evolving a ANNs or CTRNNs for movement while evolving separate ANNs or CTRNNs for shooting and aiming. The search space for the task in this project might have been complex enough to make evolving to solve it inefficient, requiring severe luck during search to find desireable solutions. Having separate ANNs or CTRNNs could require significantly less evolution time to reach a point where the pair of genomes responsible for controlling their specific mechanics exhibited satisfactory behavior. The Lamarckism direction of GAs could also be explored by for example incorporating Q Learning during evaluation, like explored by Kopel et al. [86].

For automatic selection amongst multiple best genomes from a range of generations, ELO based tournament amongst them could be beneficial to try out. It would be a more automatic way of selecting a genome which phenotype lead to the most wins against many different other genomes.

6.5.3 Future fMRI studies

As mentioned in Section 6.4.1, the eye tracking system explored in this project resulted in no data of value due to several limitations. The potential in data received from the EyeLink system could provide valuable insights in player responses and player observations. EyeLink already has integrated functionality with Unity, allowing data collectors to log which game object the participant was looking at in real time. Moments when the participant is looking at a certain bush to spot an enemy, or looking for balls scattered around the map, are great events to relate and analyze along with fMRI data.

There are other physiological data points that could provide valuable insights in the players state of mind, but were not explored in this project. Checking participant respiration rate and pulse both give great data that could correlate to brain activity or the performance data.

Even though participants were quick to adapt to the tethyx controller and the control scheme used for our study, the game is definitely in need of a more conventional way to control and interact with the environment. The controls presented in Section 4.3.1 are all limited in some way and are not the most suitable for this kind of game. An investment in a regular console gaming controller with dual thumbsticks, shoulder buttons and a normal button layout would make the game more approachable for any kind of participant. These types of controllers already exist in fMRI safe versions, but due to budget restraints it was not selected for this project.

This project was limited by the given time frame of a Master thesis, and took 6 months to complete, along with the preliminary project explained in Section 1.3. 13 participants playing each agent for 8 minutes has provided us with valuable data for this exploratory project, but is not sufficient to conclude upon the true value of NEAT as a gaming opponent. Further research in this field requires a study of higher quantity, to be able ascertain more conclusive results.

[1]

6.6 Ethical Considerations

The study was conducted in an anonymous manner to protect private information of participants in the study. The study was approved by the regional ethics committee (reference number: 469486).

[1]https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md

CHAPTER
SEVEN

CONCLUSION

Applying neuroevolution to video game opponents, specifically through techniques like NEAT, can potentially generate more unique and captivating opponents than traditional agents. Our findings, presented in Chapter 5, support this notion, showing a favorable response to the NEAT agent in comparison to the FSM and MA-POCA agents. Our exploratory fMRI study revealed that during play against the agent evolved with NEAT, participants exhibited a trending higher mean activation in the amygdala and nucleus accumbens - commonly associated with emotion processing, learning and pleasure - compared to the other agents. This increase in activation was measured via BOLD signal changes, a reliable indicator of neuronal activity.

These results offer a compelling argument for the application of neuroevolution techniques, such as NEAT, in game development. The fact that engagement with agents evolved with NEAT can drive greater neuronal activation in brain areas associated with emotions, learning and pleasure suggests that the application of NEAT has the potential to enhance the player's holistic gaming experience. This is also supported by self-reporting, where players rated the NEAT agent as having the most unique strategy, highest sense of mastery, highest entertainment value, least frustrating to play against, and most balanced in terms of joy and challenge. Additionally, the data suggested that the participants were the most likely to experience a state of flow while playing against the NEAT agent as elaborated in Section 6.1.1. However, these findings are preliminary and further research is necessary to fully understand the implications of our study. A more quantitative study would provide enough data to confirm the exciting findings from this thesis.

REFERENCES

[1] Tyler Wilde. *The freakily photorealistic 'bodycam FPS' we glimpsed last year now has a trailer and Steam page.* https://www.pcgamer.com/unrecord-announcemen/. Accessed on May 30th, 2023. 2023.

[2] Fernando Fradique Duarte et al. "A Survey of Planning and Learning in Games". In: *Applied Sciences* (2020).

[3] Hao Wang, Yang Gao, and Xingguo Chen. "Rl-dot: A reinforcement learning npc team for playing domination games". In: *IEEE Transactions on Computational intelligence and AI in Games* 2.1 (2009), pp. 17–26.

[4] Noah Wardrip-Fruin and Pat Harrigan. *First person: New media as story, performance, and game.* Mit Press, 2004.

[5] Kazimieras Mikelis. "Designing, Building, Testing and Shipping AAA-Style Video Game AI". In: (2021). URL: https://mikelis.net/designing-building-testing-and-shipping-aaa-style-video-game-ai/.

[6] Jeff Orkin. "Three states and a plan: the AI of FEAR". In: *Game developers conference.* Vol. 2006. CMP Game Group SanJose, California. 2006, p. 4.

[7] Techopedia. *How Are Finite State Machines Used in AI.* 2022. URL: https://www.techopedia.com/finite-state-machine-how-it-has-affected-your-gaming-for-over-40-years/2/33996.

[8] LabXChange. *AI in Video Games: Toward a More Intelligent Game.* 2022. URL: https://www.labxchange.org/library/pathway/lx-pathway:9b43804e-4fec-40fa-9c12-6c3bd8500d2b/items/lx-pb:9b43804e-4fec-40fa-9c12-6c3bd8500d2b:html:2c5b5b93.

[9] JuegoStudio. *AI in Gaming (FSM, MCST,VR and AR are used in gaming).* 2018. URL: https://www.juegostudio.com/blog/ai-concepts-fsm-mcst-vr-ar-in-gaming.

[10] J. Griesemer C. Butcher. *The Illusion of Intelligence: The Integration and Level Design in Halo).* 2002. URL: https://www.youtube.com/watch?v=xp468IY99ag&ab_channel=BungieHaloArchive.

[11] Game Developer. *Good AI is Predictable).* 2016. URL: https://www.gamedeveloper.com/design/good-ai-is-predictable.

[12] Katie Reddemann. "Evolving Neural Networks in NPCs in Video Games". In: *UMM CSci Senior Seminar Conference.* 2015.

[13] Christopher Berner et al. “Dota 2 with large scale deep reinforcement learning”. In: *arXiv preprint arXiv:1912.06680* (2019).

[14] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. “Deep blue”. In: *Artificial intelligence* 134.1-2 (2002), pp. 57–83.

[15] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[16] Kai Arulkumaran, Antoine Cully, and Julian Togelius. “Alphastar: An evolutionary computation perspective”. In: *Proceedings of the genetic and evolutionary computation conference companion.* 2019, pp. 314–315.

[17] Shweta Bhatt. *Reinforcement Learning.* [Online; accessed December 12, 2022]. 2018. URL: `https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292`.

[18] Marcus Andersson. “How does the performance of NEAT compare to Reinforcement Learning?” 2022.

[19] Kenneth O. Stanley and Risto Miikkulainen. “Efficient evolution of neural network topologies”. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No.02TH8600)* 2 (2002), 1757–1762 vol.2.

[20] D. E. Goldberg and J. Richardson. “Genetic algorithms with sharing for multimodal function optimization”. In: *Proceedings of the Second International Conference on Genetic Algorithms.* Ed. by J. J. Grefenstette. San Francisco, California: Morgan Kaufmann, 1987, pp. 148–154.

[21] William M. Spears. “Speciation Using Tag Bits”. In: 2007.

[22] Faustino J. Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. “Accelerated Neural Evolution through Cooperatively Coevolved Synapses”. In: *J. Mach. Learn. Res.* 9 (2008), pp. 937–965.

[23] Sebastian Risi and Julian Togelius. “Neuroevolution in Games: State of the Art and Open Challenges”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 9 (Oct. 2014). DOI: `10.1109/TCIAIG.2015.2494596`.

[24] Oliver Bown and Sebastian Lexer. “Continuous-Time Recurrent Neural Networks for Generative and Interactive Musical Performance”. In: Apr. 2006, pp. 652–663. ISBN: 978-3-540-33237-4. DOI: `10.1007/11732242_62`.

[25] Andrew C. Slocum, Doug Downey, and Randall D. Beer. “Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention”. In: 2000.

[26] Niels van Hoorn et al. “Robust player imitation using multiobjective evolution”. In: *2009 IEEE Congress on Evolutionary Computation* (2009), pp. 652–659.

[27] Andrei Faitas. “Using the NEAT algorithm for the discovery of Continuous-Time Recurrent Neural Networks with CPG-like behaviour”. 2021.

[28] Stig E. Forshult. “Magnetic Resonance Imaging MRI - An Overview”. In: 2007.

[29] Stuart Currie et al. “Understanding MRI: basic MR physics for physicians”. In: *Postgraduate medical journal* 89.1050 (2013), pp. 209–223.

[30] Edwin A. Takahashi. “How Does Magnetic Resonance Imaging Work?” In: *Essential Radiology Review: A Question and Answer Guide*. Ed. by Adam E. M. Eltorai, Charles H. Hyman, and Terrance T. Healey. Cham: Springer International Publishing, 2019, pp. 543–544. ISBN: 978-3-030-26044-6. DOI: 10.1007/978-3-030-26044-6_163. URL: https://doi.org/10.1007/978-3-030-26044-6_163.

[31] Donald W McRobbie et al. *MRI from Picture to Proton*. Cambridge university press, 2017.

[32] YaleMedicine. *Functional MRI of the Brain*. URL: https://www.yalemedicine.org/conditions/functional-mri-imaging-the-brain (visited on 10/19/2022).

[33] EyeWare. *What is eye tracking*. URL: https://eyeware.tech/blog/what-is-eye-tracking/ (visited on 11/11/2022).

[34] Charlene Ianthe Jennett et al. “Measuring and defining the experience of immersion in games”. In: *Int. J. Hum. Comput. Stud.* 66 (2008), pp. 641–661.

[35] Maurizio Mauri et al. “Why Is Facebook So Successful? Psychophysiological Measures Describe a Core Flow State While Using Facebook”. In: *Cyberpsychology, behavior and social networking* 14 12 (2011), pp. 723–31.

[36] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. *Flow: The psychology of optimal experience*. Vol. 1990. Harper & Row New York, 1990.

[37] Positive Psychology. *Mihaly Csikszentmihalyi: Father of Flow and Positive Psychology*. https://positivepsychology.com/mihaly-csikszentmihalyi-father-of-flow/. Accessed: 22.09.22. 2016.

[38] PENELOPE SWEETSER and PETA WYETH. “GameFlow: A Model for Evaluating Player Enjoyment in Games”. In: (2005).

[39] Richard Huskey et al. “Does intrinsic reward motivate cognitive control? a naturalistic-fMRI study based on the synchronization theory of flow”. In: *Cognitive, Affective, & Behavioral Neuroscience* 18 (2018), pp. 902–924.

[40] Steven Pace. “Immersion, flow and the experiences of game players”. In: May 2008.

[41] Jessica Lowell, Kir Birger, and Sergey Grabkovsky. “Comparison of NEAT and HyperNEAT on a Strategic Decision-Making Problem”. In: 2010.

[42] Alan McIntyre et al. *neat-python*.

[43] Mats Eriksen Bjerke. “Comparing neuroevolutionary methods through competitive fencing”. 2022. URL: https://www.duo.uio.no/handle/10852/95615.

[44] Matthew J. Hausknecht et al. “A Neuroevolution Approach to General Atari Game Playing”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 6 (2014), pp. 355–366.

[45] Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen. “Evolving Neural Network Agents in the NERO Video Game”. In: 2005.

[46] Uriel Mandujano and D. Hugh Redelmeier. “Evolving robots to play dodgeball”. In: 2014.

[47] Jason M. Traish and James R. Tulip. “Towards adaptive online RTS AI with NEAT”. In: *2012 IEEE Conference on Computational Intelligence and Games (CIG)* (2012), pp. 430–437.

[48] Fernando Ishikawa et al. “Playing Mega Man II with Neuroevolution”. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (2020), pp. 2359–2364.

[49] Matheus G. Cordeiro et al. “A Minimal Training Strategy to Play Flappy Bird Indefinitely with NEAT”. In: *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* (2019), pp. 21–28.

[50] Giovanna Martínez-Arellano, Richard John Cant, and David Woods. “Creating AI Characters for Fighting Games Using Genetic Programming”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 9 (2017), pp. 423–434.

[51] Markus Wittkamp, Luigi Barone, and Philip Hingston. “Using NEAT for continuous adaptation and teamwork formation in Pacman”. In: *2008 IEEE Symposium On Computational Intelligence and Games* (2008), pp. 234–242.

[52] John Reeder et al. “Interactively evolved modular neural networks for game agent control”. In: *2008 IEEE Symposium On Computational Intelligence and Games* (2008), pp. 167–174.

[53] Joan Marc Llargues Asensio, Juan Peralta, and P. Cortez. “Evolving Artificial Neural Networks applied to generate virtual characters”. In: *2014 IEEE Conference on Computational Intelligence and Games* (2014), pp. 1–5.

[54] Steffen Priesterjahn et al. “Evolution of Human-Competitive Agents in Modern Computer Games”. In: *2006 IEEE International Conference on Evolutionary Computation* (2006), pp. 777–784.

[55] Teofebano Kristo and Nur Ulfa Maulidevi. “Deduction of fighting game countermeasures using Neuroevolution of Augmenting Topologies”. In: *2016 International Conference on Data and Software Engineering (ICoDSE)* (2016), pp. 1–6.

[56] Michael D. Ferrell et al. “An fMRI Analysis of Neural Activity During Perceived Zone-State Performance”. In: *Journal of Sport & Exercise Psychology* 28 (2006), pp. 421–433.

[57] Kazuki Hirao. “Prefrontal hemodynamic responses and the degree of flow experience among occupational therapy students during their performance of a cognitive task”. In: *Journal of Educational Evaluation for Health Professions* 11 (2014).

[58] Isak Andersson. “Brain activity during flow: A systematic review”. 2022. URL: `https://www.diva-portal.org/smash/get/diva2:1687530/FULLTEXT01.pdf`.

[59] Martin Klasen et al. “Neural contributions to flow experience during video game playing.” In: *Social cognitive and affective neuroscience* 7 4 (2012), pp. 485–95.

[60] Jari Kätsyri et al. “Just watching the game ain’t enough: striatal fMRI reward responses to successes and failures in a video game during active and vicarious playing”. In: *Frontiers in Human Neuroscience* 7 (2013).

[61] Martin Klasen et al. “Think Aloud during fMRI: Neuronal Correlates of Subjective Experience in Video Games”. In: *Fun and Games* 1 (2008), pp. 132–138.

[62] Martin Ulrich et al. “Neural correlates of experimentally induced flow experiences”. In: *NeuroImage* 86 (2014), pp. 194–202.

[63] K. Saito, Naoki Mukawa, and M. Saito. “Brain Activity Comparison of Different-Genre Video Game Players”. In: *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)* (2007), pp. 402–402.

[64] Uijong Ju and C. Wallraven. “Manipulating and decoding subjective gaming experience during active gameplay: a multivariate, whole-brain analysis”. In: *NeuroImage* 188 (2019), pp. 1–13.

[65] Jari Kätsyri et al. “The opponent matters: elevated FMRI reward responses to winning against a human versus a computer opponent during interactive video game playing.” In: *Cerebral cortex* 23 12 (2013), pp. 2829–39.

[66] René Weber et al. “Theorizing Flow and Media Enjoyment as Cognitive Synchronization of Attentional and Reward Networks”. In: *Communication Theory* 19 (2009), pp. 397–422.

[67] Michael I. Posner and Steven E. Petersen. “The attention system of the human brain.” In: *Annual review of neuroscience* 13 (1990), pp. 25–42.

[68] Dimitri van der Linden, Mattie Tops, and Arnold B. Bakker. “Go with the flow: A neuroscientific view on being fully engaged”. In: *The European Journal of Neuroscience* 53 (2020), pp. 947–963.

[69] Christopher G. Davey, Jesús Pujol, and Ben J. Harrison. “Mapping the self in the brain’s default mode network”. In: *NeuroImage* 132 (2016), pp. 390–397.

[70] Debra A. Gusnard et al. “Medial prefrontal cortex and self-referential mental activity: Relation to a default mode of brain function”. In: *Proceedings of the National Academy of Sciences of the United States of America* 98 (2001), pp. 4259–4264.

[71] Marcus E. Raichle et al. “A default mode of brain function.” In: *Proceedings of the National Academy of Sciences of the United States of America* 98 2 (2001), pp. 676–82.

[72] Martin Ulrich, Johannes Keller, and Georg Grön. “Neural signatures of experimentally induced flow experiences identified in a typical fMRI block design with BOLD imaging.” In: *Social cognitive and affective neuroscience* 11 3 (2016), pp. 496–507.

[73] AbuHasan Q, V Reddy, and W Siddiqui. “Neuroanatomy, Amygdala”. In: *StatPearls*. Internet. [Updated 2022 Jul 19]. [Online; accessed June 21, 2023]. Treasure Island (FL): StatPearls Publishing, Jan. 2023. URL: `https://www.ncbi.nlm.nih.gov/books/NBK537102/`.

[74] Johannes Schmid and Rishabh Verma. “Nucleus accumbens”. In: *Radiopaedia.org* (2018).

[75] Vincent-Pierre Berges et al. *ML-Agents plays DodgeBall.* Unity Blog. July 12, 2021. URL: https://blog.unity.com/engine-platform/ml-agents-plays-dodgeball (visited on 07/12/2021).

[76] Siemens Healthineers. *Magnetom Terra).* 2023. URL: https://www.siemens-healthineers.com/magnetic-resonance-imaging/7t-mri-scanner/magnetom-terra.

[77] Current Designs. *Fiber Optic Response Devices fMRI Gamepad).* 2023. URL: https://www.curdes.com/gamepad.html.

[78] *Tethyx Joystick image.* Accessed: 2023-05-20. 2023. URL: https://www.imaging.psu.edu/facilities/3t-mri/fmri-joystick.

[79] *Response grips image.* Accessed: 2023-05-20. 2023. URL: https://www.imaging.psu.edu/facilities/3t-mri/fmri-response-grips.

[80] Evelyn Zuniga et al. “How Humans Perceive Human-like Behavior in Video Game Navigation”. In: *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems.* CHI EA ’22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391566. DOI: 10.1145/3491101.3519735. URL: https://doi.org/10.1145/3491101.3519735.

[81] Edward L. Deci and Richard M. Ryan. “Intrinsic Motivation and Self-Determination in Human Behavior”. In: *Perspectives in Social Psychology.* 1985.

[82] Oscar Esteban et al. “Analysis of task-based functional MRI data preprocessed with fMRIPrep”. In: *Nature Protocols* 15 (2019), pp. 2186–2202.

[83] Dominic Holland, Joshua M. Kuperman, and Anders M. Dale. “Efficient correction of inhomogeneous static magnetic field-induced distortion in Echo Planar Imaging”. In: *NeuroImage* 50 (2010), pp. 175–183.

[84] Anderson M. Winkler et al. “Permutation inference for the general linear model”. In: *Neuroimage* 92 (2014), pp. 381–397.

[85] Edward T. Bullmore et al. “Global, voxel, and cluster tests, by theory and permutation, for a difference between two groups of structural MR images of the brain”. In: *IEEE Transactions on Medical Imaging* 18 (1999), pp. 32–42.

[86] Marek Kopel and Tomasz Hajas. “Implementing AI for Non-player Characters in 3D Video Games”. In: *Intelligent Information and Database Systems.* Ed. by Ngoc Thanh Nguyen et al. Cham: Springer International Publishing, 2018, pp. 610–619. ISBN: 978-3-319-75417-8.

APPENDICES

APPENDIX

A GITHUB REPOSITORIES

All code and related to this thesis and in this document are included in the Github repositories linked below.

Github repository link to the NEAT algorithm implementation

- `https://github.com/KristianTve/DodgeBallEANN`

Github repository link to the dodgeball Unity environment modified for fMRI gaming

- `https://github.com/Hallahallan/Dodgeball-Bio-fMRI`

Github repository link to the original dodgeball environment made by ml-agents

- `https://github.com/Unity-Technologies/ml-agents-dodgeball-env`

Full size image of the NEAT algorithm topology

- `https://raw.githubusercontent.com/KristianTve/DodgeBallEANN/ebed3fc0c12740f4eb5279846b9ffde868b56633/best_genome.gv.svg`

APPENDIX

B

PER PARTICIPANT GAME PERFORMANCE AND QUESTIONNAIRE DATA ANALYSIS

In the following tables, a simple analysis of the game performance data and questionnaire data can be seen. The win percent denote the ratio between player wins and opponent wins for the most balanced agent for that particular participant. The most balanced agent were the one with where the participant had a win percentage closest to 50%. The lowest and highest player accuracy denote against which agent each participant had their lowest and highest accuracy.

Participant	1	2	3
Most balanced (closest to 50%)	FSM	NEAT	NEAT
Win Percent	66,67	50,00	66,67
Lowest Player Accuracy	MA-POCA (0,39)	FSM (0,27)	M-POCA (0,33)
Highest Player Accuracy	FSM (0,46)	NEAT (0,33)	NEAT (0,47)
Lowest Agent Accuracy	MA-POCA (0,16)	NEAT (0,47)	MA-POCA (0,03)
Highest Agent Accuracy	FSM (0,42)	NEAT (0,59)	NEAT (0,36)
Best General Impression	FSM	NEAT	NEAT
Most Challenging	FSM	FSM	NEAT
Most Unpredictable	NEAT	FSM	MA-POCA
Most Unique Strategy	NEAT	NEAT	NEAT
Most Frustrating	MA-POCA	FSM	MA-POCA
1st/3rd person game experience	5	3	5
Joystick experience	2	1	2
Highest avg. rated during play	FSM	NEAT	NEAT

Participant	4	5	6
Most balanced (closest to 50%)	NEAT	NEAT	NEAT
Win Percent	50,00	55,56	55,56
Lowest Player Accuracy	NEAT (0,21)	FSM (0,20)	NEAT (0,29)
Highest Player Accuracy	MA-POCA (0,31)	MA-POCA (0,51)	MA-POCA (0,37)
Lowest Agent Accuracy	MA-POCA (0,24)	MA-POCA (0,28)	MA-POCA (0,24)
Highest Agent Accuracy	NEAT (0,53)	NEAT (0,46)	FSM (0,50)
Best General Impression	FSM	FSM	FSM
Most Challenging	NEAT	FSM	FSM
Most Unpredictable	FSM	FSM	NEAT
Most Unique Strategy	NEAT	FSM	NEAT
Most Frustrating	MA-POCA	MA-POCA	NEAT
1st/3rd person game experience	4	4	3
Joystick experience	2	1	1
Highest avg. rated during play	FSM	FSM	FSM

Participant	7	8	9
Most balanced (closest to 50%)	FSM	NEAT	NEAT
Win Percent	57,14	63,64	60,00
Lowest Player Accuracy	FSM (0,18)	FSM (0,29)	NEAT/FSM (0,22)
Highest Player Accuracy	NEAT (0,34)	NEAT (0,41)	MA-POCA (0,32)
Lowest Agent Accuracy	MA-POCA (0,13)	MA-POCA (0,25)	MA-POCA (0,14)
Highest Agent Accuracy	NEAT (0,24)	FSM (0,55)	NEAT (0,30)
Best General Impression	FSM	NEAT	NEAT
Most Challenging	FSM	FSM	NEAT
Most Unpredictable	FSM	FSM	FSM
Most Unique Strategy	FSM	NEAT	NEAT
Most Frustrating	MA-POCA	FSM	NEAT
1st/3rd person game experience	5	3	4
Joystick experience	2	1	1
Highest avg. rated during play	FSM	NEAT	NEAT

Participant	10	11
Most balanced (closest to 50%)	NEAT	NEAT
Win Percent	71,43	75,00
Lowest Player Accuracy	FSM (0,33)	FSM (0,23)
Highest Player Accuracy	MA-POCA (0,46)	MA-POCA (0,52)
Lowest Agent Accuracy	MA-POCA (0,25)	MA-POCA (0,26)
Highest Agent Accuracy	FSM (0,47)	FSM (0,52)
Best General Impression	FSM	NEAT
Most Challenging	FSM	FSM
Most Unpredictable	NEAT	FSM
Most Unique Strategy	FSM	FSM
Most Frustrating	NEAT	FSM
1st/3rd person game experience	3	4
Joystick experience	2	1
Highest avg. rated during play	FSM	NEAT

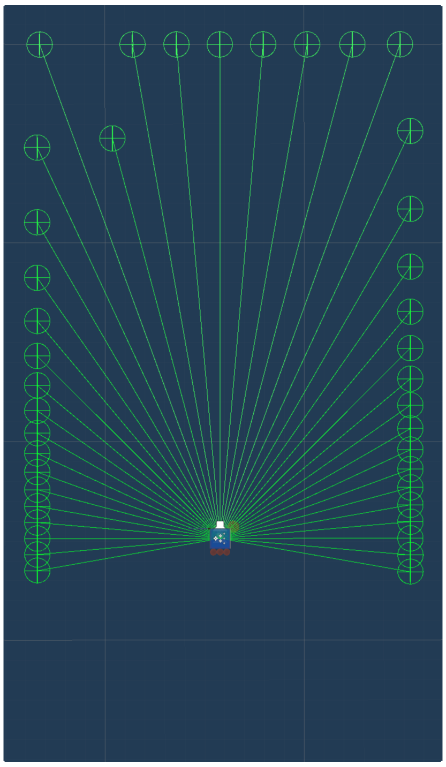
Participant	12	13
Most balanced (closest to 50%)	FSM	FSM
Win Percent	36,36	50,00
Lowest Player Accuracy	FSM (0,19)	NEAT (0,20)
Highest Player Accuracy	MA-POCA (0,41)	MA-POCA (0,43)
Lowest Agent Accuracy	MA-POCA (0,18)	MA-POCA (0,28)
Highest Agent Accuracy	NEAT (0,41)	FSM (0,41)
Best General Impression	NEAT	NEAT
Most Challenging	FSM	FSM
Most Unpredictable	FSM	NEAT
Most Unique Strategy	NEAT	NEAT
Most Frustrating	FSM	FSM
1st/3rd person game experience	4	2
Joystick experience	1	1
Highest avg. rated during play	NEAT	NEAT

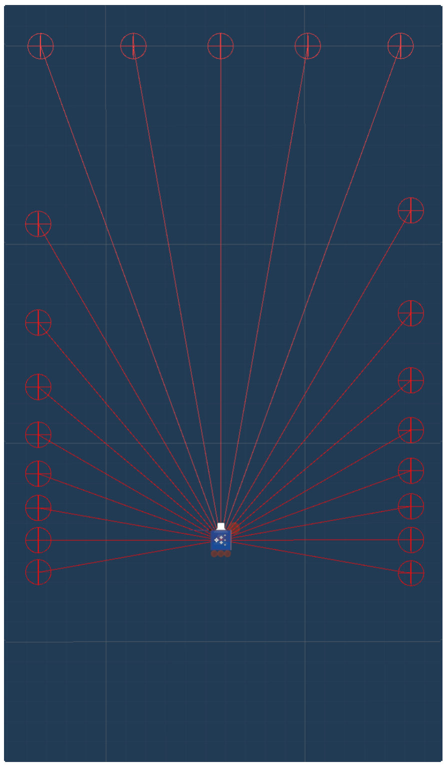
APPENDIX

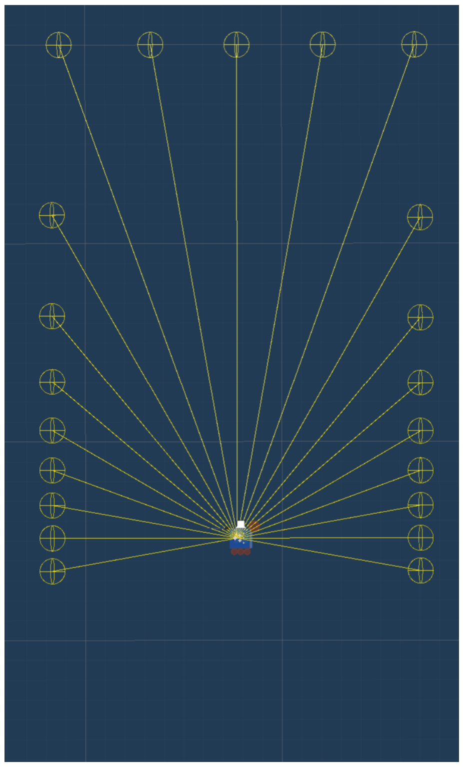
C

NEAT

Full NEAT Configuration	
Parameter	**Value**
fitness_criterion	mean
fitness_threshold	2.2
pop_size	72
reset_on_extinction	False
activation_default	tanh
activation_mutate_rate	0.05
activation_options	tanh
aggregation_default	sum
aggregation_mutate_rate	0.0
aggregation_options	sum
bias_init_mean	0.0
bias_init_stdev	1.0
bias_max_value	30.0
bias_min_value	-30.0
bias_mutate_power	0.5
bias_mutate_rate	0.7
bias_replace_rate	0.1
compatibility_disjoint_coefficient	1.0
compatibility_weight_coefficient	0.5
conn_add_prob	0.8
conn_delete_prob	0.5
enabled_default	True
enabled_mutate_rate	0.01
feed_forward	True
initial_connection	partial_direct 0.5
node_add_prob	0.8
node_delete_prob	0.5
num_hidden	0
num_inputs	364
num_outputs	5
response_init_mean	1.0
response_init_stdev	0.0
response_max_value	30.0
response_min_value	-30.0
response_mutate_power	0.0
response_mutate_rate	0.0
response_replace_rate	0.0
weight_init_mean	0.0
weight_init_stdev	1.0
weight_max_value	30.0
weight_min_value	-30.0
weight_mutate_power	0.5
weight_mutate_rate	0.8
weight_replace_rate	0.1
compatibility_threshold	3.0
species_fitness_func	mean
max_stagnation	15
species_elitism	2
elitism	2
survival_threshold	0.2


(a) AgentRaycastSensor


(b) BallRaycastSensor


(c) WallRaycastSensor

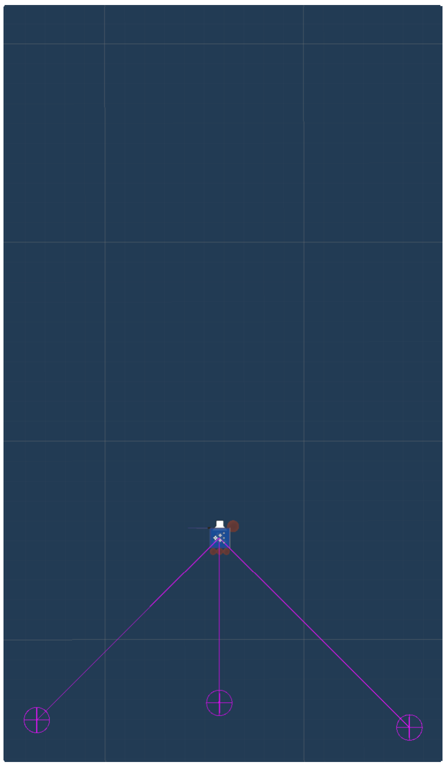

(d) BackRaycastSensor

Figure C.0.1: All of Raycasts shown individually

Correlation and K-means analysis parameters

- **mapoca:** boolean flag
- **fsm:** boolean flag
- **neat:** boolean flag
- **enemy_health:** Average enemy health
- **player_health:** Average player health
- **ball_count:** Average ball count
- **damage_per_opponent_hit:** Ratio of damage taken to damage dealt to enemy
- **enemy_accuracy:** Float 0-1: how many enemy throws resulted in damage
- **player_accuracy:** Float 0-1: how many player throws resulted in damage
- **win_pct:** float (range: 0-1):
- **took_damage:** Int: how many times the player took damage
- **player_throws:** Int: how many times the player threw a ball
- **impression_rank:** integer (ranked 1-3 where 1 is best)
- **challenge_rank:** integer (ranked 1-3 where 1 is best)
- **unpredictable_rank:** integer (ranked 1-3 where 1 is best)
- **unique_strategy_rank:** integer (ranked 1-3 where 1 is best)
- **frustration_rank:** integer (ranked 1-3 where 1 is best)
- **likeness_grade:** float (rating: 1-5, where 5 is best)
- **interesting_grade:** float (rating: 1-5, where 5 is best)
- **mastering_grade:** float (rating: 1-5, where 5 is best)
- **challenge_joy_balance_grade:** float (rating: 1-5, where 5 is best)
- **entertainment_grade:** float (rating: 1-5, where 5 is best)
- **fitting_challenge_grade:** float (rating: 1-5, where 5 is best)
- **game_familiarity:** integer (rating: 1-5, where 5 is very familiar)
- **joystick_familiarity:** integer (rating: 1-5, where 5 is very familiar)
- **age:** integer
- **experience:** float (range: 1-5, where 5 is much experience)
- **general_liking:** integer (rating: 1-5, where 5 is best)
- **clear_general_difference:** integer (rating: 1-5, where 5 is best) Perceived difference between the three agents.
- **motivated_to_play:** integer (rating: 1-5, where 5 is best) How much the player were motivated to play against the agents and improve their skill.

APPENDIX

D

BUGS AND ISSUES

NEAT bugs

It was not possible to start evolution with NEAT with more than zero hidden nodes set in the configuration file. Setting a value for "hidden_nodes" of more than 0, resulted in the genome size diverging from the pre-defined population size in the configuration. After a couple of generations, the number of genomes supplied to the evaluation function could be one less than the defined size or double the defined size. This was impossible to handle with the training setup with Unity, and could not be trusted to not destabilize the training. This bug was researched on forums, and had been present for three to four years before this project, with responses from the development team. This is the reason why the "hidden_nodes" parameter were set to 0.

Agent Bugs

One very notorious bug that needs fixing lies in the FSM agents behavior. A simple workaround was done by making the FSM a "territorial" agent that patrols in its own territory, being the square half of the map where it spawns. This prevents the bug from happening during normal play, though we observed the bug happening during play as well. If the FSM agent moves over to the other square half of the map, it will break its behavior. This results in the agent moving backwards from the center of the stage, ending up in a corner. This corner-bug breaks immersion, and tends to get negative comments by the participants. To account for this, data was logged as to whether the agent collided with invisible hitboxes in the corners during play. This way data could be quantified or excluded to determine the agent's rationality during play.

The MA-POCA agent has similar corner activity, but is different in nature. Pinpointing the exact reason for this behavior is difficult, but through observations, some hypotheses can be drawn. MA-POCA doesn't necessarily get stuck in corners, but seems to misinterpret the fact that it is walking into a wall. It doesn't seem to recognize that its path is blocked, and continues moving straight ahead, resulting in it getting stuck in a corner eventually. However, unlike the FSM, which breaks its behavior, MA-POCA tends to get out of the corner given enough time. This seems to be a fault in how MA-POCA is trained, but it is difficult to know certainly what causes this.

fMRI Bugs / Issues

The per-participant estimated SAR value were wrong, resulting in the protection measures being set in place to avoid too much radiation to be surpassed. To work around this, the number of slices which where scanned of the brain was reduced for one of the experiment days. This issue was later fixed with a system reboot. It was theorized that this was due to some caching issue or bug within the system.

Eye Tracking Bugs / Issues

There were no saved EDF files after finishing trial, were stored at the host PC however. This is theorized to be due to the disconnect of the HDMI cable between the play against each agent. As of submitting this thesis, there is no conclusion to the cause of this issue, though it is in progress.

FMRI COMMON BRAIN NETWORK ACTIVATION FOR ALL PARTICIPANTS ACROSS ALL AGENTS

Brain region	**Hemisphere**	**C #**	**C size**	**t-value**	**X**	**Y**	**Z**
———————————————————-*MAPOCA*———————————————————-							
–*Positive effects*–							
Lateral_occipital_cortex	L	10	14002	11.4	-15	-78	52.5
Superior_parietal_cortex,7A	L	10	14002	11	-12	-60	57
Superior_parietal_cortex,7P	R	10	14002	10.7	15	-69	60
Supplementary_motor_cortex	R	9	6815	12	6	-3	63
Premotor_cortex,BA6	R	9	6815	8.29	24	-6	48
Superior_frontal_gyrus	R	9	6815	7.86	30	-1.5	61.5
Middle_frontal_gyrus	R	9	6815	7.78	30	3	49.5
Visualcortex_v1	R	8	2081	9.28	12	-82.5	4.5
Cerebellum,VermisVI	R	8	2081	6.29	1.5	-76.5	-24
Cerebellum,VermisVI	L	8	2081	6.21	-4.5	-72	-15
Intracalcarine_cortex	L	8	2081	6.14	-18	-70.5	7.5
Lateral_occipital_cortex	R	7	1410	7.76	48	-66	7.5
Middle_temporal_gyrus	R	7	1410	7.23	42	-58.5	3
Brocas_area	L	6	713	12.2	-52.5	7.5	27
Premotor_cortex,BA6	L	6	713	6.23	-55.5	4.5	42
Inferior_frontal_gyrus	L	6	713	5.18	-58.5	10.5	15
Middle_frontal_gyrus	R	5	507	7.59	31.5	28.5	36
Frontal_pole	R	5	507	6.35	33	36	27
Insula,anterior	R	4	456	7.39	34.5	22.5	6
Inferior_frontal_gyrus	R	4	456	5.15	45	21	18
Frontal_operculum_cortex	R	4	456	4.74	42	16.5	3
LingualGyrus,anterior	L	3	424	5.79	-27	-51	-6
Parahippocampal_cortex,posterior	L	3	424	4.95	-18	-43.5	-10.5
Occipital_fusiform_gyrus	L	3	424	4.28	-27	-67.5	-4.5
Thalamus,Prefrontal	R	2	313	11.2	12	-18	12
Thalamus,Temporal	R	2	313	5.86	10.5	-12	19.5
Thalamus,Premotor	L	1	296	6.39	-16.5	-16.5	4.5
Thalamus,parietal	L	1	296	5.86	-15	-25.5	10.5
Thalamus,Prefrontal	L	1	296	5.81	-13.5	-22.5	10.5
–*Negative effects*–							
Cingulate_gyrus,posterior	L	6	1371	9.03	-9	-51	30
Precuneous_cortex	L	6	1371	8.04	-1.5	-63	33
Cingulate_gyrus,posterior	L	6	1371	6.52	-10.5	-39	33
Precuneous_cortex	R	6	1371	6.31	1.5	-64.5	34.5
Frontal_pole	L	5	915	6.94	-13.5	64.5	33
Frontal_pole	R	5	915	6.54	4.5	70.5	19.5
Middle_temporal_gyrus	R	4	482	9.78	70.5	-39	-1.5
Inferior_parietal_cortex,Pga	R	3	431	6.33	43.5	-58.5	40.5
Lateral_occipital_cortex	R	3	431	4.89	46.5	-60	52.5
Superior_temporal_gyrus	L	2	350	7.07	-67.5	-36	3
Middle_temporal_gyrus	L	2	350	5.46	-45	-36	-3
Orbitofrontal_cortex	L	1	348	7.14	-25.5	28.5	-19.5
Frontal_pole	L	1	348	3.86	-28.5	40.5	-12

———————————————————*FSM*———————————————————-

–*Positive effects*–

Lateral_occipital_cortex	L	8	36145	18.2	-16.5	-75	57
Superior_parietal_cortex,7A	L	8	36145	18.1	-13.5	-60	57
Precuneous_cortex	R	8	36145	16.6	4.5	-58.5	61.5
Superior_parietal_cortex,7P	R	8	36145	15.8	9	-76.5	55.5
Cerebellum,VI	L	7	2061	10.8	-9	-73.5	-18
Cerebellum,VermisVI	L	7	2061	10.7	-1.5	-72	-18
Cerebellum,V	L	7	2061	8.61	-1.5	-61.5	-6
Occipital_fusiform_gyrus	L	7	2061	8.4	-18	-73.5	-16.5
Cerebellum,IIV	R	7	2061	7.23	1.5	-51	0
LingualGyrus,posterior	L	7	2061	6.02	-7.5	-79.5	-15
Brocas_area	L	6	1530	9.46	-58.5	10.5	27
Inferior_frontal_gyrus	L	6	1530	9.16	-54	9	22.5
Middle_frontal_gyrus	L	6	1530	8.5	-55.5	9	39
Premotor_cortex,BA6	L	6	1530	7.12	-55.5	3	40.5
Lateral_occipital_cortex	R	5	1018	14.4	52.5	-72	4.5
Middle_temporal_gyrus	R	5	1018	6.11	45	-58.5	6
Middle_frontal_gyrus	R	4	970	8.2	30	28.5	37.5
Frontal_pole	R	4	970	7.93	34.5	39	43.5
Thalamus,Premotor	L	3	959	8.82	-16.5	-19.5	10.5
Caudate,posterior	L	3	959	8.37	-12	-9	19.5
Thalamus,Prefrontal	L	3	959	8.16	-10.5	-22.5	4.5
Thalamus,Occipital	L	3	959	7.16	-16.5	-28.5	16.5
Thalamus,parietal	L	3	959	5.59	-22.5	-30	7.5
Thalamus,Primarymotor	L	3	959	5.45	-22.5	-15	3
Middle_frontal_gyrus	L	2	734	10.7	-33	30	28.5
Frontal_pole	L	2	734	8.87	-33	40.5	12
Thalamus,Prefrontal	R	1	717	9.75	10.5	-16.5	9
Thalamus,parietal	R	1	717	8.18	13.5	-24	12
Thalamus	L	1	717	5.77	-3	-30	-1.5
	–Negative effects–						
Precuneous_cortex	L	2	2159	13.5	-1.5	-63	27
Precuneous_cortex	R	2	2159	11.3	3	-60	28.5
Cingulate_gyrus,posterior	L	2	2159	6.9	-1.5	-31.5	39
Cingulate_gyrus,posterior	R	2	2159	6.72	4.5	-49.5	13.5
Middle_temporal_gyrus	L	1	1362	11	-64.5	-46.5	0

————————————————————————-*NEAT*————————————————————————-

	–Positive effects–						
Lateral_occipital_cortex	L	7	12335	14.4	-22.5	-76.5	33
Superior_parietal_cortex,7A	L	7	12335	10.5	-19.5	-64.5	64.5
Lateral_occipital_cortex	R	7	12335	9.37	27	-82.5	34.5
Superior_parietal_cortex	L	7	12335	9.22	-31.5	-52.5	69
Premotor_cortex,BA6	L	6	4901	11.2	-45	-6	51
Superior_frontal_gyrus	L	6	4901	9.92	-21	3	60
Middle_frontal_gyrus	L	6	4901	9.32	-34.5	-1.5	67.5
Brocas_area	L	6	4901	7.7	-60	10.5	28.5
Superior_frontal_gyrus	R	5	2958	9.62	22.5	-4.5	49.5
Supplementary_motor_cortex	L	5	2958	6.4	-6	-3	57
Premotor_cortex,BA6	R	5	2958	6.29	27	0	46.5
Lateral_occipital_cortex	L	4	889	7.96	-48	-75	-3
Lateral_occipital_cortex	R	3	770	7.93	43.5	-61.5	0
Middle_temporal_gyrus	R	3	770	6.69	46.5	-57	7.5
Cerebellum,IIV	R	2	641	9.1	6	-52.5	-4.5
Cerebellum,V	L	2	641	4.65	-1.5	-63	-7.5
Thalamus	L	1	513	7.57	-1.5	-28.5	-3
Callosal_cortex	L	1	513	7.42	-3	-31.5	-3
Thalamus,parietal	R	1	513	6.93	13.5	-21	13.5
Thalamus,Prefrontal	R	1	513	6.89	12	-18	12
Thalamus,Prefrontal	L	1	513	5.44	-9	-16.5	-1.5
	–Negative effects–						
Cingulate_gyrus,posterior	L	3	669	5.91	-7.5	-54	28.5
Precuneous_cortex	L	3	669	5.8	-1.5	-61.5	33
Cuneal_cortex	R	3	669	4.19	0	-78	33
Superior_temporal_gyrus	L	2	515	7.79	-61.5	-34.5	0
Middle_temporal_gyrus	L	2	515	7.62	-61.5	-31.5	-1.5
Central_opercular_cortex	R	1	475	6.48	52.5	-10.5	13.5
Secondary_somatosensory_cortex	R	1	475	6.31	63	0	10.5

————————————————————————-*Baseline 1*————————————————————————

	–Positive effects–						
Lateral_occipital_cortex	L	7	2811	8.49	-18	-84	27
Occipital_pole	L	7	2811	5.79	-25.5	-90	16.5
Inferior_frontal_gyrus	L	6	2156	11.4	-55.5	12	15
Brocas_area	L	6	2156	9.35	-55.5	7.5	25.5

Superior_frontal_gyrus	L	6	2156	8.32	-24	-1.5	46.5
Premotor_cortex,BA6	L	6	2156	7.23	-40.5	-7.5	54
Frontal_operculum_cortex	L	6	2156	7.22	-42	15	7.5
Central_opercular_cortex	L	6	2156	6.85	-45	6	10.5
Occipital_pole	R	5	1146	8.11	18	-88.5	18
Lateral_occipital_cortex	R	5	1146	7.08	31.5	-85.5	12
Visual_cortex,v2	R	5	1146	4.75	21	-90	31.5
Lateral_occipital_cortex	R	4	1044	8.8	45	-61.5	0
Middle_temporal_gyrus	R	4	1044	6.1	49.5	-49.5	3
Lateral_occipital_cortex	L	3	1000	9.23	-16.5	-75	54
Superior_parietal_cortex,7A	L	3	1000	7.12	-10.5	-60	58.5
Inferior_parietal_cortex	L	2	541	7.57	-54	-37.5	42
Supramarginal_gyrus	L	2	541	5.5	-60	-28.5	42
Postcentral_gyrus	L	2	541	5.17	-61.5	-19.5	37.5
Visualcortex_v2	R	1	455	5.97	25.5	-93	-7.5
Visual_cortex,V3	R	1	455	5.9	24	-87	-10.5
Lateral_occipital_cortex	R	1	455	5.52	37.5	-87	-7.5
	–Negative effects–						
———————————————————*Baseline 2*———————————————————-							
	–Positive effects–						
	–Negative effects–						

Table D.0.1: Brain regions denote which part of the brain the activity was measured in, with hemisphere denoting whether the left (L) or right (R) hemisphere of the brain was activated. C stands for Cluster and is abbreviated to make space for the table. The t-value is the maximal value measured. X, Y, and Z pinpoint the brain coordinates for the peak signal.

APPENDIX

E

CORRELATION MATRIX WITH P-VALUE ANALYSIS

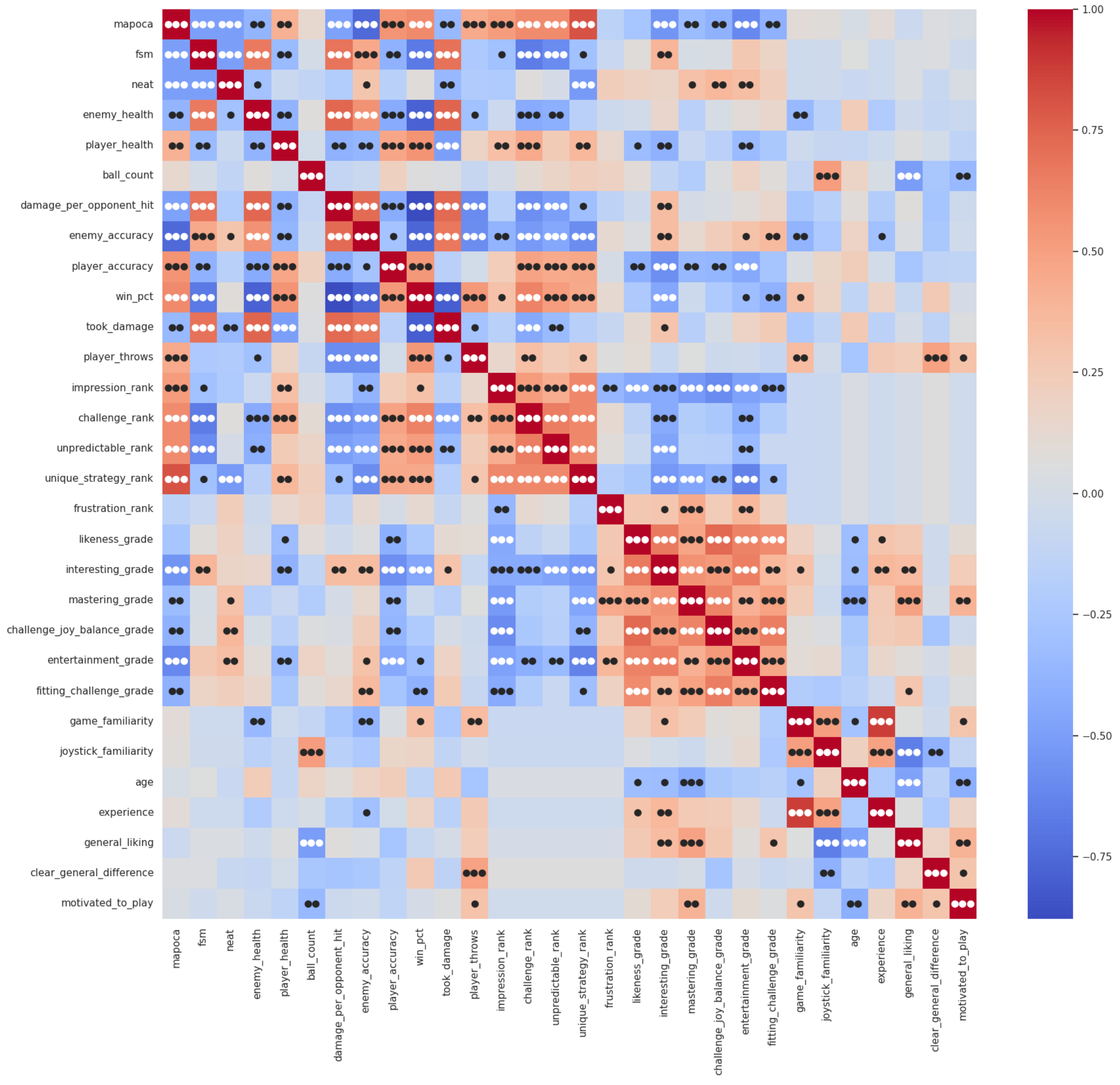


Figure E.0.1: Correlation matrix with P-value analysis. P values between 0.05 - 0.1 are marked with one dot, 0.01 - 0.05 with two dots, and lastly below 0.01 with three dots. P-values below 0.05 are regarded as statistically significant.

APPENDIX

F

NOTES

F.0.1 17.04

Retro
First run was messy, but it went alright. This was a sort of pilot test, so given the circumstances this went quite well. We had some delay in the setup and during the study, so measures will be made to prevent this in future studies.

Setup
Setup needs to be quicker, we need to set finite timers for all parts of the study, to ensure a consistent completion time. Firstly, participants will be introduced to the planned study briefly, before being sent into the test playing room while other preparations are made. Here they are asked to re-dress into the fMRI coat, sign the agreement paper and the fMRI compatibility paper. The remaining time will go to test playing to ensure that they know all the controls and rules of the game. They will then require a couple of minutes to get properly prepared for the fMRI machine, with the controller setup being the most time consuming part.

In the fMRI machine
After today, we have decided that 10 minutes is enough time to play against each agent. This is due to feedback and observations from the players, which lost interest after 10 minutes of play. This also alleviates more time for other essential time consumers, such as preparation scan, questionnaire and baseline.

At the beginning of the fMRI scan, each participant must go through a scan lasting approximately 8 minutes. This is the scan that creates the initial images of the brain, and is essential. While this scan is running, participants will calibrate the eye tracking software to save time.

Questionnaires will be carried out right after play to get the most immediate response while the participant remembers it. Afterwards, baselines will be played for 3 minutes in between each of the agents to allow for brain signals to relax.

Finally the participants need to “cool down” for one minute. This is a sort of post process for the fMRI imaging.

This plans for approximately 60 minutes of runtime in the fMRI machine, which means that the preparations need to be swift to complete it in the given timeframe of 1 hour 30 min.

Suggested time frame after first run:

Pre-scan preparations 0-5: Participants arrive, are introduced to an agenda of the study and informed about essentials.

5-15: Participants are placed in test play room, asked to re-dress, sign forms and play the game to practice.

15-25: Participants are placed in the fMRI machine, strapped and ready with controls.

In the fMRI machine 25-35: Initial scan with brain mapping and eye tracking.

35-40: Initial Baseline play(Note, baselines are 3 minutes, but setup and change of scenes takes time

40-50: Players play against first agent

50-55: Questionnaire about first agent

55-65: Players play against second agent

65-70: Questionnaire about second agent

70-80: Players play against third agent

80-85: Questionnaire about third agent

F.0.2 19.04

Retro

This study was much better than the previous one, but was also characterized by unexpected issues. The 7T machine wasn't working at the beginning of today's session, which meant it needed to be reset. This led to the participant being put in the machine at approximately 8:50, 50 minutes after arrival. This was also partly due to the eyelink equipment not working properly.

With 40 minutes to play against agents, the participant ended up playing 6 minutes against the NEAT agent and the MA-POCA agent. This is a testament to the amount of time needed for things unrelated to the game, which means we need to plan more time for other things. We therefore reduce the time for playing against agents yet again from 10 minutes to 8 minutes. We will also try to reduce time expended in other areas to alleviate the time pressure. One suggestion was to invite the player to the office from 7:45, a quarter earlier, to have more time to complete the arrivals. This will be tested on friday.

Issues

Eyelink

The eyelink software wouldn't connect to the host. Meaning that we didn't get to see where the patients were looking, only recordings of their eyes. We will attempt to implement the eyelink this Friday, and if it doesn't work we will drop it.

Machine

When we arrived in the morning, the fMRI machine didn't work. This is another issue that is out of our control, but is detrimental to the study. In today's study we had to cut down the agent pay time to 6 minutes and only play against 2 of 3 agents. It's good that we got some data from the session, but we got a lot less data than wanted. The issue with the machine was something the radiologists hadn't seen before, but was fixed by a complete reset of the system.

F.0.3 21.04

Notes during the study

Recorded today's session while running it to try to measure time usage in fMRI machine. We also asked the participant to arrive 15 minutes earlier, which they complied to. These 15 minutes proved to help a lot, as they had much more time to get introduced to the study.

Retro

Only thing cut from today's session was the final questionnaire, which may be cut from this point on as it can be done after extracting the patient. We barely had time to complete the session, but today marked the first day we got complete data from a study, which is great.

Lap	Lap times	Overall time	Activity
01	1:27	1:27	Initial 1 minute scan and talking
02	10:22	11:50	TN recording 8 minute anatomic scan
03	1:23	13:14	Adjusting images
04	1:14	14:28	Shimming
05	2:46	17:15	Baseline(Faulty, screen not on)
06	1:46	19:02	Fixing problem
07	3:22	22:24	Baseline
08	8:14	30:39	Playing against first agent (MA-POCA)
09	2:31	33:10	Questionnaire for first agent (MA-POCA)
10	8:13	41:32	Playing against second agent (NEAT)
11	1:58	43:30	Questionnaire against second agent (NEAT)
12	8:23	51:54	Playing against third agent (FSM)
13	4:18	56:12	Final 1 minute scan and baseline
14	1:02	57:14	Extracting participant

Table F.0.1: Timing fMRI study

F.0.4 24.04

Notes during study

Today's first study is no exception to unexpected issues. Eyelink is a little painful, but now runs from our end. We observe that the participant is laying slightly off center. At least the coil interferes with his eye, so some of the data is perceived by the eye tracker as missing.

There was also an unexpected error where the scanner wouldn't start when starting at the first baseline. The radiographers explained this as impossible to detect pre-emptively, as it had something to do with the structure of the brain, so we received an error.

This occurred again during the second study, which is still caused by a strange occurrence.

The first participant had 64 slices in their brain, which the radiographers speculated could be the issue. This is because the normal amount of slices in humans is 82 slices. The second participant had 70 slices, which may further prove that this was the case of the error.

It is still unknown whether this caused the error, but if it did, it is out of our control. We cannot anticipate nor see this in patients until the fMRI machine gives off the error signal.

Second participant had great results with the eyelink. Calibrated with 5 dots this time, since the previous participant had the most amount of dots. We theorize that these dots are outside the field of vision for the patient, meaning they do not see them and some of the calibrations throw off the rest.

F.0.5 25.04

Notes during study

First participant has issues with calibrating the eye tracker. LY goes out of bounds, and has trouble with the bottom calibration. We suspect this is caused by a partially closed eyelid, which makes the tracker interpret the LY axis wrong.

Observed something peculiar, particularly with this participant. Participants tend to answer very high on the first question in the questionnaire, and then rate the agents lower and lower if they weren't impressed with it. It might've been worthwhile to ask the first question again at the end of the questionnaire.

Observed something interesting that has previously been observed as well. Players tend to play a lot more intensively when playing against the pink agent compared to the red agent. In this run we started with the red agent and followed by the pink. When the agents change, the player's behavior changes dramatically. Sometimes players passively observe the red agent as it runs around a lot and doesn't necessarily shoot at them. When playing against the pink agent we observe more dodging and activity

Second participant had similar issues with the eye tracker. The radiographers went into the scanner room to check for anomalies. There doesn't seem to have been any adjustments to the screen, but it seems that the participants cannot see the bottom part of the calibration. This worked fine in yesterday's experiment when we got it to work for the first time, so this is strange.

F.0.6 26.04

Notes during study

First participant had great eye tracking data. All runs were successful and representative of the agent's behavior. On time.

Second participant had problems with eye tracking calibration again. After asking participants we are now certain that they do not see the lowest dot on the calibration. This renders the eye tracking almost useless.

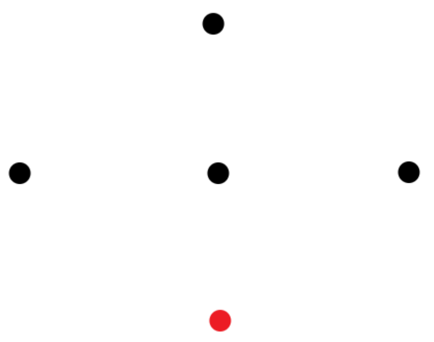
We observe some irregularities in the play of the agents. Second participant rated pink agent the lowest it has ever been rated, but not without reason. During observations of play, the agent was very passive and circled around the player most of the time. These irregularities in play can cause different perspectives of the agent's intelligence, and is worth noting in the paper.

Another observation is that players that play passively against agents, meaning they explore the game more than trying to win, tend to give the agents a poor score. Players that play to win tend to think of the agents as more competitive.

F.0.7 27.04

Notes during study

Before today's first participant, I ran through a visual presentation of how the eye tracking calibration will look like. This is an attempt to "cheat" the fault of participants not seeing the lower calibration dot, and get some good eye tracking data. Below is a representation of the eye calibration where the black dots are visible dots and the red dot is outside the field of view of the participant.



The participant was asked to measure the distance between the middle dot and the top dot, and try to estimate where the lower dot was. The participant was also informed that we can try this indefinitely, so preferably he should keep attempting until a satisfactory result has been achieved.

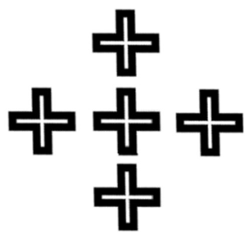
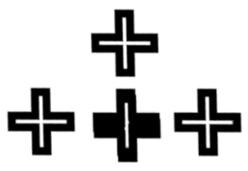
To no end. It is possible that it is not possible to trick the eye tracker this way, though it happened to an earlier participant that they simply tried to look in that direction which resulted in the best eye tracking data we have so far.

After turning off randomized order and spending some extra time communicating with the participant, we managed to get calibration in order with a sequential calibration order.

Now we are getting great eye tracking data.. finally..

The rest of the study went as planned, we haven't had problems with time since 21.04, every run from then on out has seen participants play against all agents and complete questionnaires within the time frame.

Second participant was informed of the same calibration routine. We seem to have gotten a decent calibration for this participant as well.

	
What we want the calibration to look like	What it looks like when lower dot is not properly calibrated

Observed extremely good play for the second participant. This is reflected in the answers on the questionnaire.

F.0.8 28.04

Notes during the study Final day

Participant has some issues with eyelink, calibration is still difficult.

Observed that the participant has struggles with the control scheme. This may affect play and results.

Interestingly this participant seems to play very passively. We observe a lot of hiding behind bushes and waiting for a chance.

Orange had worst results observed possibly ever, which is a little sad to see. It got stuck alot in the corners. Luckily we account for this in the game logs, but this is of course reflected in the questionnaires.

APPENDIX

G

PARTICIPANT COMMENTS FROM POST-FMRI QUESTIONNAIRE

Participant 1:

- The controls in the environment at the level with the pink agent unfortunately didn't work, so it's number 3. The orange one was very engaging to play against compared to the red one - it was much less passive and behaved more interestingly.

- The controls didn't work properly with the pink one, which was quite frustrating. The red one was so passive that I started to get bored.

- The orange one was incredibly aggressive; it attacked every single chance it got. The red one was very passive and almost never attacked. The pink one behaved very unpredictably and it was never entirely clear when it was going to attack.

- All the agents tended to get stuck in the bottom corners. All the agents liked to throw multiple balls in quick succession. It makes sense that this was the most effective, but it was somewhat frustrating when you can't shoot as fast yourself due to the control.

Participant 2:

- (I did not play against red) I preferred playing against pink because I had time to try different tactics with the pink as it was roughly as good as me. And it was fun to win since I won about half of the time.

- The orange one was most accurate and harder to find in the grass. It often had many balls in hand while the pink one often didn't have any balls.

- The orange one had two different levels I felt. At the beginning, it was "really good" and the other times it played much worse and just stood in the grass without shooting.

- The pink one took a longer time to approach me as a player but it didn't always shoot even though it saw me. Sometimes it came after me and other times it hid.
- I got irritated by how good the orange one was at the beginning, and equally irritated when it suddenly got worse which I didn't like.
- The pink agent could also have been better at not going into the grass - which would have increased the difficulty further because I wouldn't have known where it was all the time.

Participant 3:

- Felt the pink agent were more realistic, and tried to dodge and use the trees as cover.
- Felt like maybe I was more engaged and focused when I was playing against the pink one, but that could be because I had become more accustomed to the equipment.
- Felt the red one was very random in everything it did and didn't feel like it behaved like a real player would.
- Felt the pink one used a more "human" strategy.
- It might have been because I wasn't entirely comfortable with the controller, but it was most frustrating to play against the red one.

Participant 4:

- Orange allowed for more varied strategies as it seemed to try different things itself. For red, it seemed like one had to "cheese" to win against it.
- Pink was the most challenging to play against because it was better at dodging and using dash-strafing. Orange was pretty balanced. Red, again, was just about "cheese", so the game was "solved" rather quickly.
- Orange employed slightly different strategies. Sometimes it would sit in the corner and wait for me, other times it would rush me down. Pink didn't have such "wild" strategy switches but moved quite varied. Red went for the same strategy almost every time.
- Pink was unique because it wasn't necessarily that varied. The strategy was largely based on active dodging.
- (Pretending it is a real player). It felt like it was "cheese or be cheesed", and that it was not the gaming skills that determined the outcome. (Red, in case of frustration).

Participant 5:

- Orange was the most "active". I found that red, in particular, was more passive.
- Orange was the most satisfying to beat. Red was a bit easy to beat.
- Unfortunately, I don't remember exactly which one, but I think it was orange. I seem to recall that it could move through the bushes. (unpredictability)
- It was a bit annoying that they could end up standing in the corner of the field.

Participant 6:

- I felt that orange was a challenging (=fun) level of difficulty. Red was a bit easier, but still fun to play against. Pink was difficult, but frustrating because it moved so erratically.
- Orange was good at shooting and moved sensibly. Pink was difficult because it was so unpredictable. Red was fairly calm and easier to both locate, hit, and dodge.
- I felt that orange and red followed a somewhat similar strategy, but with a variation in difficulty. Pink's, on the other hand, was something completely different, hence the most unique.
- Pink, because of its unpredictability. It was close between red and orange, but both sometimes felt frustratingly "dumb" in that they didn't actively fetch new balls when they were out. (question about frustration)
- I struggled a bit with the joystick in the beginning (on pink), but managed to position it better as I went along, which made it easier to aim.

Participant 7:

- Orange is the most varied and at the same time challenging. Red is much better than the pink agent, but it forces you into a monotonous strategy.
- Orange was unpredictable and good, while red was relatively predictable. Pink didn't do much, so it was easy to win against.
- I struggled to predict orange, while red was relatively easy to predict.
- It mainly comes down to unpredictability again. The more unpredictable, the more unique the agent feels. (Orange, question about unique strategy)

- Some of the frustration is about losing against the agents. Therefore, pink is the least frustrating because it was the easiest agent. Red feels more frustrating than orange because it forces a certain type of strategy, instead of allowing for multiple strategies.
- The orange agent felt the most fun, but it seemed like at times it would intentionally become weaker. If this is the case, and one figures it out, it would ruin much of the fun for me.

Participant 8:

- Orange was a bit too difficult, while I managed a bit better with pink.
- Orange moved a lot and quickly, and actively tried to get hold of balls to shoot at me. Pink did not actively try to get hold of any ball, while red was somewhat in between. Orange was also the only one that seemed to be more offensive than defensive.
- Orange moved a lot, and if I lost sight of it, it was hard to find it again.
- There was a very noticeable difference in the strategy of pink after playing against orange. The strategy of red was not as clear to me as orange and pink.
- Orange was much faster and much more accurate than I was. I often lost before I had time to think.
- I think orange would have been the most enjoyable agent to play against in the long run since the other two seemed to rather run away from you than come towards you.

Participant 9:

- Red did not have any particular play pattern. Orange did some things that were a bit strange (stuck in the corner), pink was the most challenging and therefore the most interesting to play against.
- Same as the previous answer really. Pink seemed the most strategic.
- A bit torn between pink and orange, really. I remember orange tricked me a bit once on where it was. I did not realize that it was behind me.
- Pink had the most "human" strategy. Orange seemed to be on its way to do things that were just weird, like being stuck in the corner. Red didn't have any special strategy.
- None were particularly frustrating, but if I have to rank them, I think the most frustration came from those that were the most challenging. For example, I felt a bit tryhard against pink.

- Since the first agent (red) was very simple, and the second agent (orange) was a bit harder, I felt like I got the difficulty level in "chronological" order. Therefore, I became a bit biased towards the last agent (pink) and thought that this one was going to be even more difficult. This might have made me a bit nervous/anxious and I really wanted to win.

Participant 10:

- Pink had no action pattern. Like all the others, it just shot when it saw you. Very boring. Red felt a bit like playing against a child. Orange felt "on".
- Pink was nothing, Red was no challenge and Orange was something you had to try.
- Always knew that Red ended up in its corner and was cowardly. One could say Pink was simple to predict as it just spun around.
- Randomness is unique.
- Felt that pink and red were more of a waste of time than orange.

Participant 11:

- The first one was very unpredictable, as it constantly moved around quickly. (FSM/Orange)

Participant 12:

- Very balanced agent. It was easier to improve against it. (Pink)
- The agent was very unpredictable. Sometimes it shot 3 balls in quick succession, while other times it was very easy to kill. (Orange)
- As mentioned previously, sometimes it shot 3 balls in quick succession, while other times it didn't shoot at all. (Orange) (Regarding unpredictability)
- The agent was more balanced. It kept its distance and shot from afar when there were no bushes between us. (Pink)
- Because it did so many odd things. Sometimes it was easy, and other times it was difficult. The red one was too easy. (Orange, regarding frustration)
- Very Fun!

Participant 13:

- No comments

APPENDIX

H

EYE TRACKING RESULTS AND EXPERIENCE

From the fifth participant onward, the eye tracking recording with WebLink seemed to be successful, where the data files were being saved for the initial participants. However, there was a struggle with obtaining a good calibration for each participant with the five point calibration setup. Participants usually missed the lower calibration dot leading to a skewed calibration in the Y axis. The participants were queried about the issue and mentioned that there were no lower dot visible. The screen were adjusted too low. Due to a complicated setup in the fMRI room, the screen could not be adjusted in time by the right personnel. A workaround was to instruct the participants to look down as far as the screen went if they did not observe a dot on the screen during the calibration phase.

There were only a total of two participants managing to be correctly calibrated. Of which none of them were instructed the workaround for the calibration, and both claimed to not observe the lower dot. None of the participants which were told about the work around managed to get a good calibration in spite of many attempts during the initial scan period.

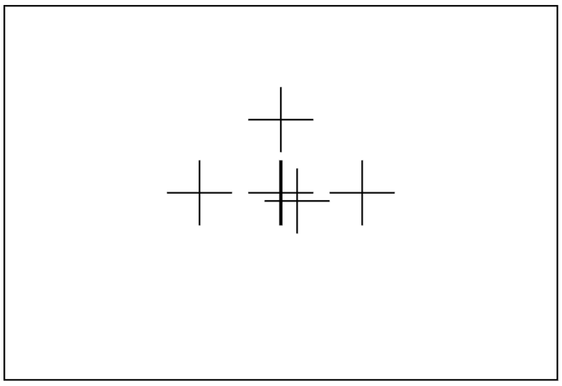
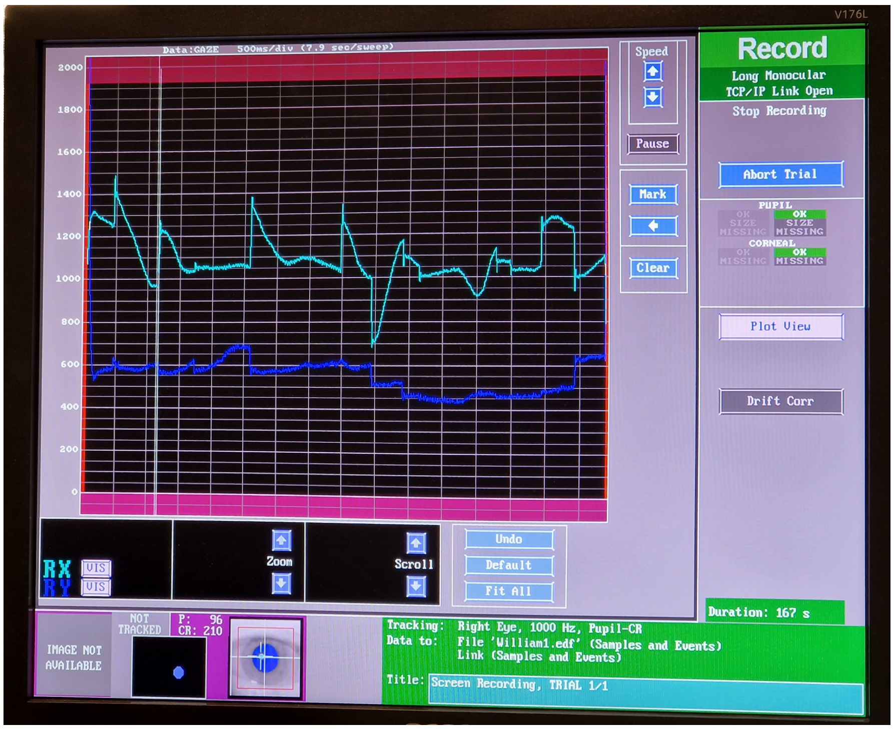


Figure H.0.1: Poor five point calibration due to a non-visible lower calibration dot.

Due to the poor calibration sequence, most of the eye tracking data raised concerns when observing eye movement during the fMRI study. In the operational room, a monitor is dedicated for all EyeLink related protocols. While the calibration clearly exhibited poor calibration on most participants, we still let the EyeLink run in hope of some salvageable data. Following is an example of how the data looks with excellent calibration.



```
Tracking:  Right Eye, 1000 Hz, Pupil-CR
Data to:   File 'William1.edf' (Samples and Events)
           Link (Samples and Events)
Title:     Screen Recording, TRIAL 1/1
```

Figure H.0.2: EyeLink monitor observations with excellent calibrations

The monitor displays the participants gaze in real-time. In the bottom left of the screen, the selected eye is displayed. In the case of this study, the right eye was selected at most times due to the positioning of the EyeTracker equipment. Due to shadows being cast on the left eye, the tracker had difficulties discerning the pupil from shadows, resulting in better data from the right eye. The pupils size and gaze are both recorded, with the gaze being displayed in the large line-graph in the top-left. The light blue line represents the X-axis of the gaze, while the dark blue represents the Y-axis. A white line moves from left to right, updating this graph with a time interval of 7.9 seconds. While this shows the ideal results, it is also important to recognize that this was not the case for the majority of the studies.

Presented above is a clear exhibit of how the poor calibrations shown in figure H.0.1 can affect the data. Due to the lower dot calibration error, EyeLink struggles to separate two different parts of the gaze. Discerning between the gaze points is impossible, resulting in the choppy line visible in H.0.3.

Another exhibition of a common challenge in Eyetracking is presented showing the poor data received when EyeLink cannot perceive the persons eye properly. Red sections represent the loss of recorded vision and mainly occurs in three different ways.

Firstly and most naturally occurring is if the participant is blinking or closing their eyes. This is completely natural, and cannot be avoided. Secondly, the threshold set for the participant makes EyeLink wrongfully assume that the eye is not detected. The threshold is set to discern the pupil from the cornea, and also the cornea from the rest of the observed details in the EyeLink camera. EyeLink automatically sets this threshold, although it can also be manually adjusted.

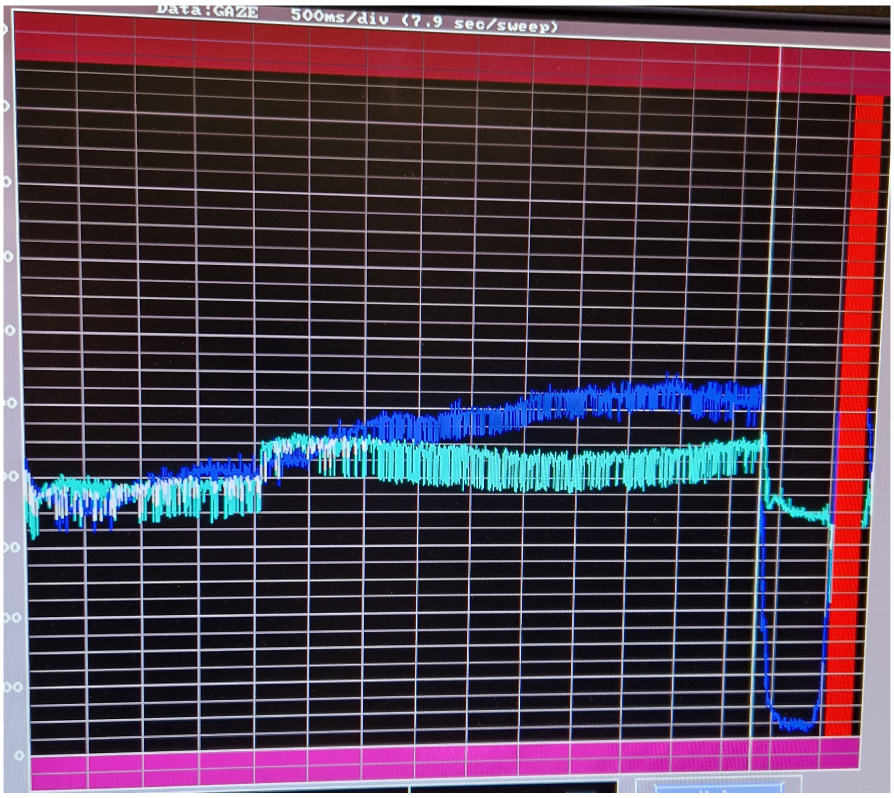


Figure H.0.3: EyeLink monitor observations with poor calibrations

There are still challenges that can be detrimental for the proper threshold, such as shadows creating large black fields close to the eye that are hard to discern from the pupils. This can also be challenging due to the large variations among the human eye. Eye size and eye openness can impact the quality of the data perceived by EyeLink. Finally, EyeLink can wrongfully assume that the participant is not looking anywhere inside the screen. This is mainly due to the wrongful calibrations, but has an even more severe impact on the data than the choppy lines. Another example is shown below where these two faults are both highly prevalent, invalidating the data further.

After the experiment were completed, the resulting data were investigated. There it was discovered that the important data files for analysis were completely missing after the first five participants of the second week of experiments. The fifth participant the second week also had partially missing EDF (EyeLink Data File) files, which suggests something happening to the WebLink and EyeLink software. Additionally, for the EDF files that were saved, the data tracking seemed to last only from start of the recording to the start of the game where it promptly stopped. Even though the video files were recorded throughout the session, the actual eye data recording would stop without any notification of this during the recording session. The absence of the EDF files and lack of data for the ones that existed, precluded any feasible analysis of the eye behavior during gameplay.

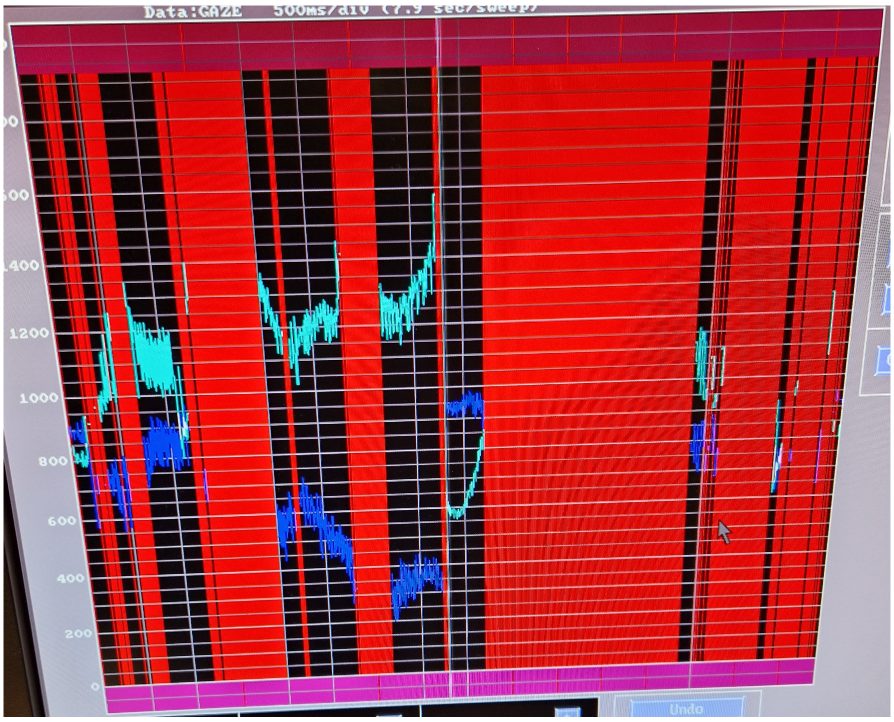


Figure H.0.4: EyeLink monitor observations with poor threshold

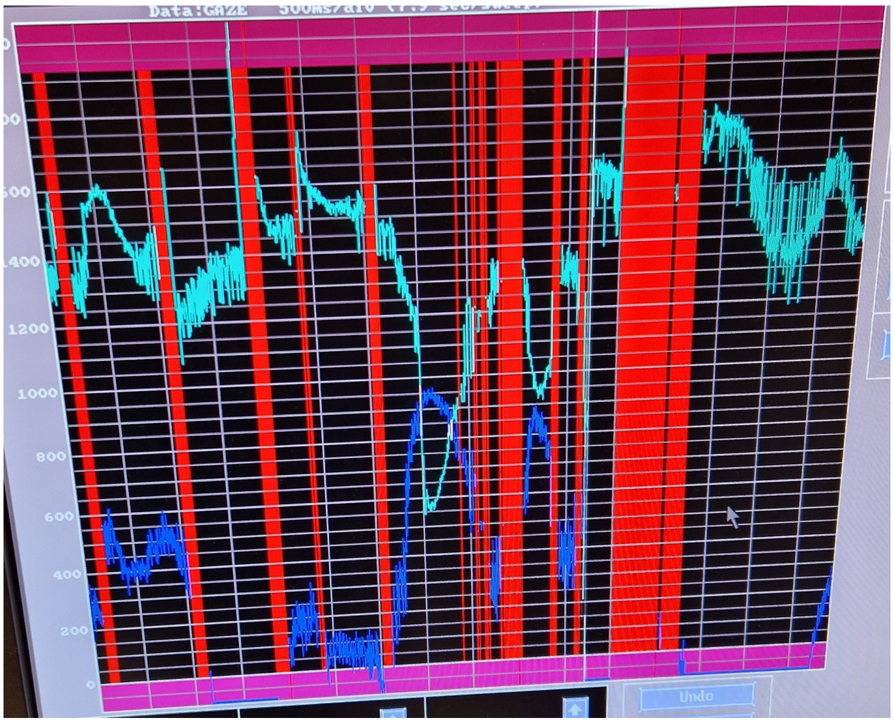


Figure H.0.5: EyeLink monitor observations of faulty data

APPENDIX

I

GAMEFLOW FRAMEWORK

Sweetser et al. [38] further extend Csikszentmihalyi's description of flow by a framework consisting of eight elements mapping flow aspects to video games. As the flow experienced in computer games can be described through Csikszentmihalyi's model of flow, the mapping of the GameFlow model is as follows in Table I.0.1

Games Literature	Flow
The Game	A task that can be completed
Challenge Player Skills	Perceived skills should match challenges and both must exceed a certain threshold
Control	Allowed to exercise a sense of control over actions
Clear goals	The task has clear goals
Feedback	The task provides immediate feedback
Immersion	Deep but effortless involvement, reduced concern for self and sense of time
Social Interaction	n/a

Table I.0.1: *The GameFlow framework by Sweetser et al*

From the GameFlow framework, we find that three of the aspects are particularly interesting for this project. Challenge and player skill define the balance between the skill level of the player and the challenge the player is facing. Keeping a good balance by adapting this challenge to the skill level keeps the player from getting an intrinsic reward without getting bored. Control is an important factor; without it, frustration can easily follow, ruining the flow experience. Immersion is also a highly relevant concept for this project, as immersion brings with it enjoyment as the player is invested in the game.

APPENDIX

J

PARALLELIZATION OF THE ENVIRONMENT FOR MULTI-CORE TRAINING

For faster training, the High-Performance computer at NTNU called Idun was at disposal. The server ran Linux which required a Linux build of the game. Due to Python only running on a single thread, running the python algorithm without parallel processing on Idun would be a waste of time. Especially since the single core performance on the CPUs running on the Idun platform, is outperformed by mid-range consumer CPUs.

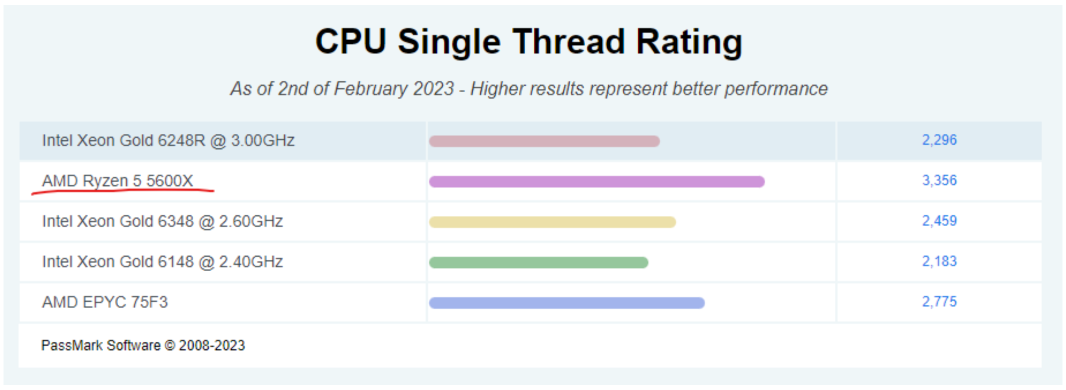


Figure J.0.1: Benchmarked single core performance of a mid range aging CPU at disposal for the project (underlined in red), against Idun CPUs

The NEAT algorithm supplies a multiprocessing class for running multiple parallel evaluation instances in each thread. Due to the fixed nature of Unity, where there only are X number of agents, running multiple evaluation functions did not seem feasible.

The biggest bottleneck in terms of time usage in the evaluation function was theorized to be the activation of the neural network to obtain action output for each agent. This was confirmed by running TimeIt tests in python, where the neural net activation was the part of the code requiring the most resources. To solve this, a possible solution was to assign different neural networks to separate threads. The main algorithm would then wait for all threads to finish before progressing further in the algorithm. This way actions can be obtained in a parallel fashion reducing time.

Typical python multithreading implementations does not actually result in parallelism, but there is only one thread running at a given time and all runs sequentially. Due to the NEAT implementation with neural network forward pass being a computationally expensive operation, this was not the preferred option.

The other alternative was to implement multiprocessing. This technique facilitates true concurrency in Python, though it's not without its drawbacks. It necessitates the duplication of the script's entire memory for each spawned subprocess, potentially leading to considerable memory consumption. Attempting multiprocessing showed a significant slowdown of 15 times the original run time, most likely bottlenecked by the serialization and deserialization for each of the sub-processes.

For the final training runs of this project, a HPC computer with a Ryzen 7 5800X was sourced. The 5800X had less cores than the Idun CPUs and had significantly stronger single core performance.