ADAPTING TRANSFORMERS TOWARDS A BALANCED METAGAME THROUGH RETRAINING AND LAYER-PRUNING.

a thesis submitted to

The University of Kent

in the subject of computer science

for the degree

of msc by research.

By

Alan J. Osys

June 2026

Abstract

With the recent strides in machine learning and a growing interest in using Transformers in many different areas, the question has arisen of what can this architecture do for improving the quality of games with procedurally generated characters, such as the Pokemon game. In this context, the overall aim of this thesis is to produce sets of Pokemon (metagames) where the individual Pokemon in a set have more balanced win rates (computed in battles) and more diversity (different Pokemon property values), when using a Transformer model to generate Pokemon.

In order to address that aim, this thesis offers two main contributions, involving two proposed methods for improving the Pokemon metagame generated by a Transformer. The first contribution is the approach of pruning layers of a Transformer model, whilst the second contribution is the development of a multi-objective genetic algorithm for selecting Pokemon that optimise the balance and diversity of a Pokemon metagame. The computational results have shown that layer-pruning is paramount in preserving Pokemon diversity over time – every Transformer model produced without using layer pruning was significantly outperformed (regarding diversity) by models generated using pruning. In addition, the combination of layer pruning with a multi-objective genetic algorithm further improved the results, achieving overall the best results in terms of maximising diversity and balanced win rates over time.

Acknowledgements

I would like to thank Anna Jordanous for being very supportive and encouraging during the time when we were writing the paper on layer-pruning and for being an incredible first supervisor. I would like to thank my second supervisor Alex Freitas for his continued support with the genetic algorithms side of this thesis and for his insightful ideas on how to improve the project. I would also like to thank the AIDA group for taking me in during my first year at the university. All of my academic interests and drive can be attributed to these weekly meetings that opened my eyes to research. And lastly my parents since without their support I would not have the ability to do this degree.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This Introduction chapter is structured as follows. Section 1.1 describes the motivation for this research. Section 1.2 presents the overall aim and the two specific objectives of this research. Section 1.3 describes the research contributions. Finally, Section 1.4 briefly describes the structure of the remainder of this thesis, by mentioning the contents of each chapter.

1.1 Motivation

Game design is an ever-growing field, with new and constantly evolving games being made. In this thesis I focus on Player versus Player games (PvP). In PvP games, two or multiple players are pitted against each other in an environment, with some tools, or entities, such as Pokemon, to battle. In First Person Shooter (FPS) games, these entities are guns and various weapons, which all have their own variables and statistics, used to eliminate the enemy team. In Fighting Games (FG), these entities are the characters which have various moves with internal variables that need to be balanced, such as frame data and damage.

One of the most popular PvP games is Pokemon, with a vast variety of entities which are the Pokemon themselves. I decided to focus on Pokemon, because of

the easy to use Pokemon Showdown API which allows me to simulate battles, a readily available dataset, and my extensive knowledge of competitive Pokemon that I have acquired through first hand experience as a player.

Each of these games, as well as many more genres out there that have entities used to battle, need to have a dedicated balance team to adjust these entities frequently so that each player has a fair chance at winning while using a variety of different entities. This is called game balancing and is the main area under investigation in this thesis.

A metagame is the set of entities used by the players that have proven, through playing and testing, to give them the best chance of winning. A metagame also encapsulates how these entities are used by different players, a way of letting the players express themselves and adapt to the metagame created by the player-base (S. Reis, Novais, et al. 2023). A metagame must be diverse to be balanced, this is what makes it compelling, as Richard Garfield, the creator of Magic the Gathering, once said: "I believe that a compelling metagame is what separates a hobby from a game". Having a single entity that has an overwhelming win rate against other entities cannot be considered balanced, as the game slowly starts revolving around either playing it or beating it, since it gives you the best chance of winning. Game balancing is difficult and often requires full teams of designers and play-testers to adjust these entities. Often the teams are unable to predict how each different entity will interact, which causes some to slip through the cracks and dominate the metagame.

My research aims to automate the process of creating a balanced set of entities that will develop into a metagame. This makes the balance team's job easier by having a system that uses the win rate as the metric, a metric present in every PvP game and one that is easily extracted, to create the entities and update the metagame over time.

1.2 Research Aims and Objectives

The overall aim of this project is to produce Pokemon metagames that are more balanced and more diverse, when using a Transformer model Vaswani et al. 2023 to generate Pokemon. In this context, balance refers to the balance of win rates among the different Pokemon generated by a Transformer model, when those Pokemon compete in battles. That is, there should be no major difference between the win rates of different Pokemon, because such major difference would lead to an "un-balanced", uninteresting metagame. Diversity refers to the diversity of the Pokemon metagame generated by a Transformer model, i.e. those Pokemon should have substantially different sets of values for properties describing a Pokemon (like their type, items, abilities etc. - see Section 2.2).

In order to achieve that overall aim, this thesis has two specific objectives:

- First, to test a neural network-pruning technique for pruning layers of a Transformer model, in order to increase the diversity of Pokemon generated by the Transformer.
- Second, to develop a Genetic Algorithm for selecting a set of Pokemon that maximise the diversity and the balance of a Pokemon metagame.

1.3 Research Contributions

This thesis provides two contributions. First, it uses layer-pruning in Transformers to improve the diversity of the metagame generated. This is a new use of layer-pruning in Transformers, as previous related work has used layer-pruning to decrease computational time, as will be discussed in Section 2.7. The thesis reports statistically significant results demonstrating the effectiveness of layer-pruning for improving diversity.

Second, this thesis proposes a multi-objective genetic algorithm (GA) for selecting the set of Pokemon that maximise diversity and balance in a metagame. More precisely, the thesis proposes two versions of this multi-objective optimisation (MOO) GA: one where the two objectives to be optimised (diversity and balance) are combined via a Weighted Sum, and another based on the concept of Pareto Dominance (to be discussed in Section 2.8.6).

The unique way of utilising layer-pruning has yielded very positive results, where Pruning models significantly outperformed models that did not use pruning such as the Weighted Sum with Pruning approach which outperformed almost every other tested model, which are discussed more in-depth in Chapter 4. The other contribution of using MOO GAs as selection algorithms was able to significantly increase the balance of the generated metagames, with every GA-based model significantly outperforming the models using randomly selected datasets.

1.4 Structure of the Thesis

The remainder of this thesis is structured as follows. Chapter 2 provides a review of the literature, focusing on the Pokemon game and machine learning concepts relevant for this research. Chapter 3, on the proposed methodology, describes the proposed machine learning methods (involving neural-network pruning and a genetic algorithm), as well as the dataset used in the experiments and the experimental setup. Chapter 4 reports the computational results obtained by the proposed methods, and an analysis of those results. Finally, Chapter 5 concludes the thesis, providing a summary of its main contributions and some suggestions for future work.

Chapter 2

Literature Review

In this chapter I will describe the previous research done in the relevant areas of this thesis as well as giving background to terminology and concepts needed for this work. The areas of interest are Procedural Content Generation, Pokemon, Game Balancing, Repercussions of Unbalanced Games, Measuring Diversity, Transformers, Genetic Algorithms and Significant Tests. All of these help achieve the aim of guiding a Transformer towards creating a more balanced and diverse metagame through multi-objective optimisation retraining and layer-pruning.

2.1 Procedural Content Generation in Games

Procedural Content Generation (PCG) is a widely explored area in which a system generates some content without human help (Togelius et al. 2011). Even before the development of Transformers, there are many examples of how PCG can be used to create new and interesting content in games. Some of the most well-known examples are the world and creature generation in No Man's Sky (2016), world generation in Minecraft (2009), and the entirety of the world and story of Dwarf Fortress (2006). In all of these games, PCG plays a crucial role, defining large aspects of the games. PCG can be used to generate a world map, cards

(Mahlmann et al. 2012), blending game concepts to create new games (Gow and Corneli 2015), and even the backstory of a world Mason et al. 2019. With the rise of Transformers, their utilisation in PCG was inevitable, as they were able to generate game levels (Mohaghegh et al. 2023), interactive stories (Freiknecht and Effelsberg 2020) and even play games (Upadhyay et al. 2019).

PCG is becoming more widely adapted as the scale of games grows and players' desire grows to explore bigger and more varied worlds. PCG is perfect for this as it reduces the costs and minimises effort, while being able to produce very intricate worlds, characters and more. This comes with the issue of uninteresting content being generated (Hendrikx et al. 2013). Evaluating procedural content is a widely explored area for this reason (Osborn et al. 2019).

2.2 What is a Pokemon

In the Pokemon games, Pokemon are unique entities that have a large pool of moves they can use, a pool of abilities, individual EVs (which are variables attached to each Pokemon that determine their prowess in certain aspects such as physical attack or defence and stand for Effort Values), and natures to choose from, and access to a pool of items.

Pokemon can have one or two of 18 different types: Normal, Fire, Water, Grass, Electric, Ice, Fighting, Poison, Ground, Flying, Psychic, Bug, Rock, Ghost, Dragon, Dark, Steel, and Fairy. The types are responsible for assigning weaknesses and resistances to the Pokemon. Each type has assigned weaknesses which means when hit by a move type they are weak to they will take double damage, this is called a Super Effective hit as seen in Figure 7. Resistance allows the Pokemon to instead take half damage from certain move types. For example if a Pokemon is a rock and ground type it will take four times the damage from water moves due to both

rock and ground being weak to it.

Each Pokemon's move pool (an average of 58 different moves per Pokemon) consists of both physical attacks and special attacks, as well as moves that deal no damage but instead serve a purpose such as status effects as seen in Figure 4 and Figure 5. Out of the large move pool, 4 can be chosen to use in battle as seen in Figure 3. Moves are chosen by the players for each of their Pokemon before a battle starts.

Out of the pool of abilities, that is unique to each Pokemon (an average of 2 abilities), one can be chosen alongside an item of the player's choosing. Items are chosen from a large pool that every Pokemon shares. Items give the Pokemon a specific property such as Choice Specs seen in Figure 1 giving the Pokemon greatly increased Special Attack, but once the player chooses a move, they cannot use a different move until the Pokemon switches out.

A Pokemon has the following Effort Values (EVs): Health, Attack, Special Attack, Defence, Special Defence, and Speed. A distribution of up to 508 points can be spread out across all of the 6 EVs for each of their Pokemon, 252 maximum per EV, before a battle starts.

Nature affects the EVs of a Pokemon by decreasing one EV by 10% in order to increase another EV by 10%.

In a regular game of Pokemon, 2 players each have a team of 6 Pokemon. They send one Pokemon out, and every turn both players choose either one of their moves or to switch out to a different Pokemon. Both players make a decision at the same time. Unlike in a game of chess or go, where the players move after another. This makes Pokemon a game where you need to predict what your opponent will do at any given moment.

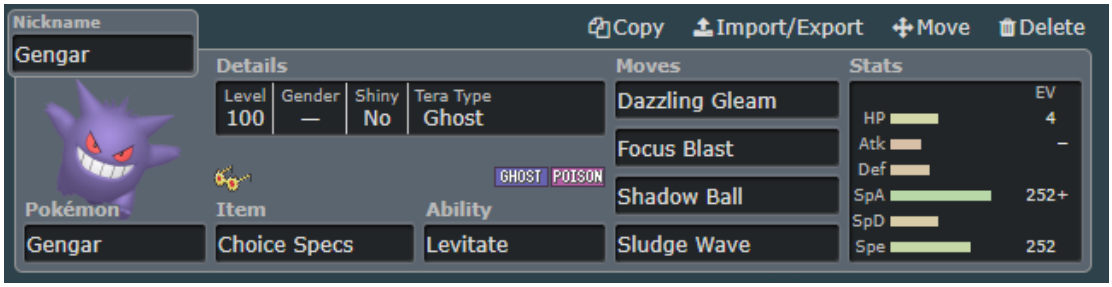

Figure 1: Pokemon example: This is an example of a Pokemon. The same example is outlined in Table 2.

In previous literature, Pokemon has been used mainly as an example of a very complex game for Reinforcement Learning agents to learn (Pleines et al. 2025). Pokemon battles are very complex, each Pokemon has the possibility of using such a large number of moves, items and abilities, each of them changing the outcome of the game that standard algorithms like the ones seen in chess cannot predict best moves. This is why there is a lot of work developing agents that can play at human or above human level, making Pokemon a great benchmark for very complex games. Unlike both of these, this thesis focuses on the entities themselves and the battles are only used to extract the win rates of the entities.

Figures 1-8 are examples of different scenarios that can occur in a standard game of Pokemon. As these figures show, there are many different strategies a player can choose to use, which all combine to make a metagame. Each player must be aware of all of these strategies and their impact on the metagame. Strategies like the ones seen in Figures 4 and 5, commonly referred to as 'Stall' tend to be seen as very annoying [1] to play against. Thus the Pokemon balancing team has been staying away from defensive, status-based strategies and instead improving the aggressive and big damage dealing strategies. This can be seen in Figure 3, as they are more flashy and fun for people to play, play against and watch in tournaments.

[1] https://www.reddit.com/r/stunfisk/comments/11ww9fz/thoughts_on_stall/ Last accessed August 21st 2025

Figure 2: Pokemon interface example: This is an example of what a Pokemon battle interface looks like. Two Pokemon are battling against each other, each player is able to either use one of their Pokemon's 4 attacks or they can switch to one of their other 5 Pokemon. The opponent has to also choose what to do. Once both players select an option the priority goes as follows: Switching always comes first, the move of the faster Pokemon is executed, the move of the slower Pokemon is executed. If a player chooses to switch out their Pokemon, they are unable to attack that turn and will have to wait until the next turn, which can leave their Pokemon vulnerable.

Randomness is a key component of Pokemon that adds to the excitement of the game. Figures 6 and 8 both illustrate this randomness that is very much part of the game's identity. That being said, there is a lot of strategy and working around the randomness. Minimising risk while maximising reward is particularly present when creating your team. Picking moves that do not have 100% accuracy

Figure 3: Pokemon attack example: This is an example of what happens when an attack is chosen. An animation plays and the Pokemon that was attacked has its health depleted by the amount of damage it took. In this particular example the Ursaring dealt 100% of the Weezing's health in one attack, meaning Weezing is knocked out and can no longer be used in this battle. This was due to using Sword's Dance move the previous turn, which doubled the physical attack power of the Ursaring as well as the Ursaring having a status condition (an effect which stays on the Pokemon until it is healed and can either deal damage or reduce the EVs of the Pokemon afflicted) placed on them. More on status conditions is shown in Figure 4. The move Facade deals more damage if the user is afflicted with a status condition, which allowed the Ursaring to have the capability of depleting most of the opposing Pokemons health one attack.

is the player's choice, but it allows them to have a much bigger reward when the move does not miss and instead it deals the expected damage at all times. Picking a variety of Pokemon to maximise the chance of having a Super Effective advantage (dealing super effective damage increases the amount of health lost by the Pokemon hit), which can be seen in Figure 7, is very important and a large part of the game as it allows a Pokemon to deal increased damage against a favourable match up. This increases the diversity of teams the players create as having a variety of types will allow a player to be more prepared for different

Figure 4: Pokemon status condition example: In this example, the Lugia is afflicted with Paralysis status condition which causes it to move slower while also having a 50% chance at failing to execute a move. The opponents Sandy Shocks Pokemon is built as a defensive Pokemon that utilises Paralysis and slowly reduces the opposing Pokemon's health. There are many status conditions, but their utility and reason for being used is outside the scope of what is needed to understand this thesis.

teams and not get caught out by a single Pokemon that is able to deal Super Effective damage to their whole team.

2.3 Game balancing

As introduced in Chapter 1, game balancing is concerned with having an entity's win rate as close to 50% as possible. This gives both players using it a fair chance at winning and the more skilled player will tend to win. If a metagame has more entities with a win rate approaching 50%, it is likely to have more entities present in the competitive environment, as players will have a broader choice of entities with similar win rates. This, in turn, directly translates to better player and

Figure 5: Pokemon special condition example: In this example the opposing Persian is afflicted with Leech Seed which each turn depletes 12.5% of their health while healing the allied Pokemon the same amount. This is another strategy similar to the one outlined in Figure 4, but this can be used alongside a status condition like paralysis or sleep.

skill expression, due to the actions taken in game having higher impact than the entities chosen. Luck will always have an impact on any game. In a balanced environment the luck factor can have amplified impact on the outcome, however, the same luck is present in unbalanced metagames and can skew the outcome in the favour of the person using the best strategy even further.

Becker and Görlich 2020 states that no clear definition of game balance has been agreed on, however from their findings it is clear that balance requires both variety and the lack of centralising entities which the metagame revolves around. The work presented in David Stephens 2024 gives a better example of balance that I will be using. Their definition of balance in multiplayer games states that all game elements, in this case all the entities, should be relevant and usable, with no single strategy dominating. The aim of balancing is to avoid making the game monotonous and predictable. This definition fits well with the research aim of this

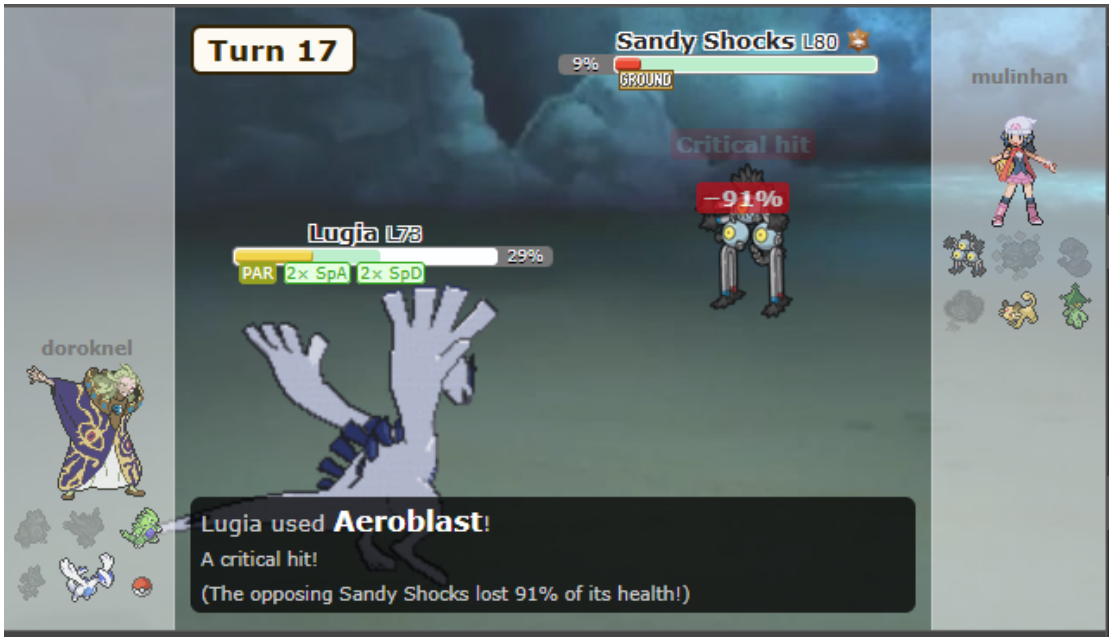

Figure 6: Pokemon critical hit example: In this example the move I used critically hit, every damaging move has approximately 4% chance of hitting critically. This makes the move deal double its usual damage. This adds a big element of luck to the game, where sometimes a Pokemon that should lose gets a critical hit and wins through sheer luck.

thesis, which is to create a balanced and diverse metagame, without overwhelmingly powerful entities and with as many usable entities as possible.

Balancing game elements (entities) such as fighting game characters, cards in card games, and Pokemon in Pokemon games, is vital to the success of these genres (S. Reis, L. P. Reis, et al. 2021). If there is a centralised pool of very strong entities in a given game, it means the metagame will revolve around these entities, creating a very boring and stale environment (Pfau and Seif El-Nasr 2024). Understanding how different entities can interact with each other in the game, both as teammates and as opponents, is a very difficult task to tackle (Aha et al. 2005), (S. Reis, Novais, et al. 2023). Autonomous game balancing has been explored using reinforcement learning which focused on adapting the opposing bot to the players level, in a fighting game, to create a more fun experience, the limitation of this

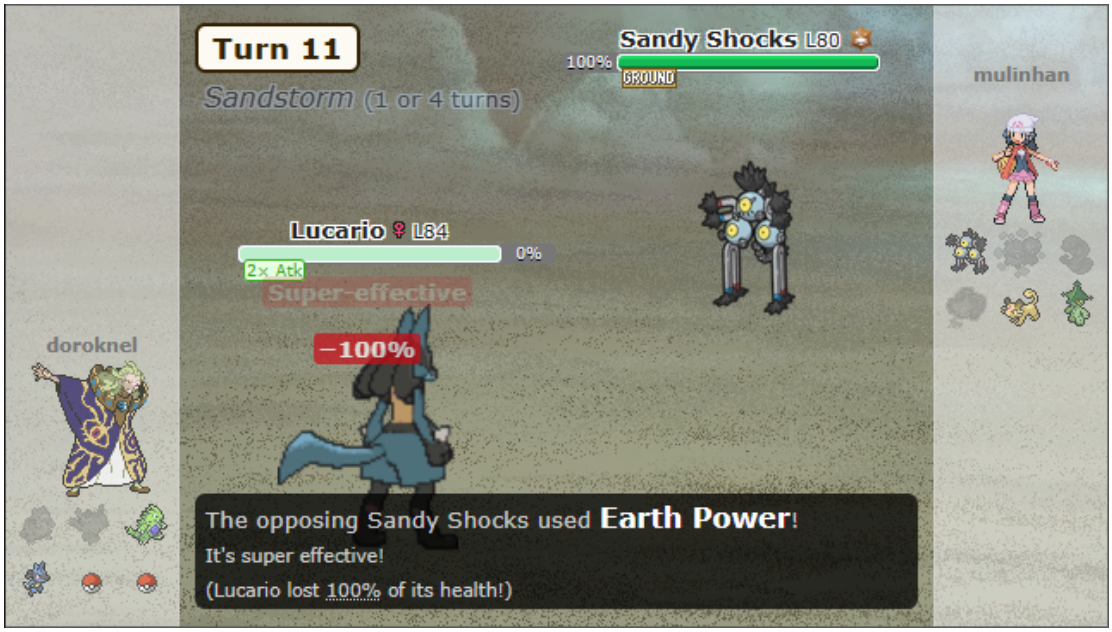

Figure 7: Pokemon super effectiveness example: This example shows the importance of typing in a game of Pokemon. Each type has weaknesses and strengths and some even have immunities. When a Pokemon is hit with a super effective move, the damage it takes will be doubled. This means having a variety of move types on a Pokemon, as well as many different types of Pokemon on a team, is very important so as to not get caught out by an unfavourable match up.

approach is that it only works for non-player characters and cannot be used in a multi-player environment(Aha et al. 2005) and user-based evaluation that takes player feedback, which can be slow and unreliable as well as being difficult to acquire unlike in-game statistics such as win rate or play-rate (Andrade et al. 2006).

No literature can agree on how a balanced game is defined. Every game is different and balance varies greatly on many aspects, such as the amount of different entities, and how the entities' win rate is calculated. In Pokemon, since each entity is placed on a team with 5 other Pokemon, the win rates of individuals are affected by their teammates. A win rate of 50% is widely considered balanced, since when someone uses an entity with a 50% win rate they have a 50% chance of winning, thus their skill is what determines wins or losses. Very rarely can a

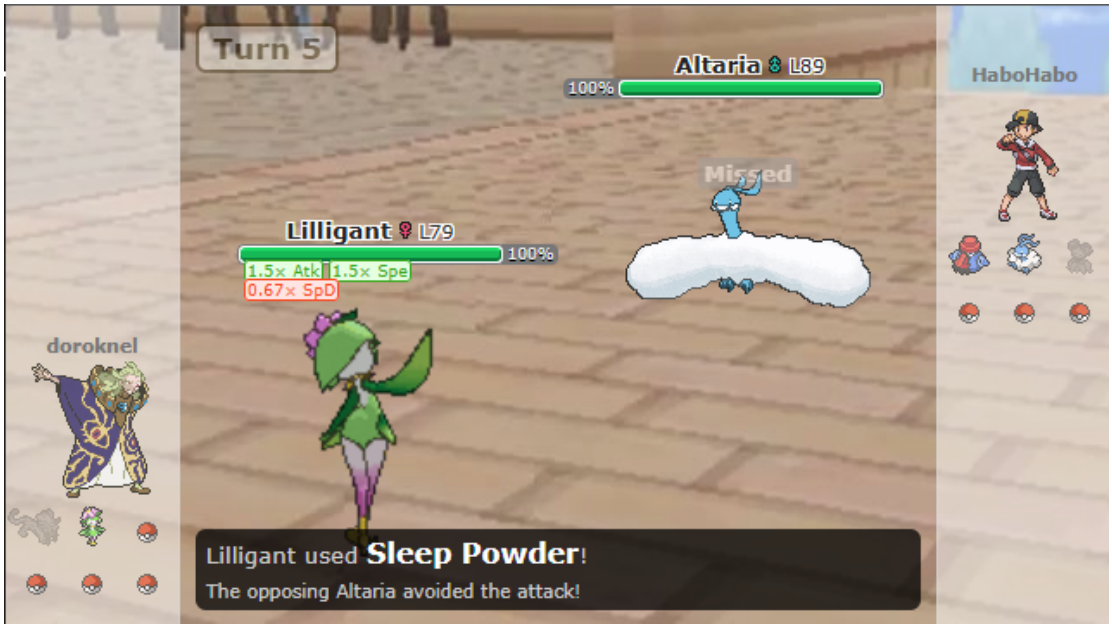

Figure 8: Pokemon miss example: This example shows when a move is not 100% accurate. Every move has accuracy and some can miss. This adds yet another layer of luck much like the one exemplified in Figure 6. A famous example of this is when a top player Aaron Zheng lost a game in the semi finals are Vancouver regionals, because he missed two 85% accurate moves in a row and his Pokemon was subsequently hit critically.

win rate of 50% be achieved, especially in a game where each Pokemon has other Pokemon on the team helping and skewing the win rate, so it is useful to use a larger window around 50% for balance.

Quantitative balancing has been explored in works such as DeLaurentis et al. 2021 where they agree a 50% win rate is considered balanced while proposing new approaches such as Monte-Carlo Tree Search algorithms for the task of balancing based purely on quantitative data. Every game is different which means the quantitative approaches will change from game to game. They are important to explore due to the nature of being able to statistically prove balance. The main limitation of this is that a lot of the data is difficult to extract for other games. This means the approach needs to change drastically if it were to be used for fighting games for example.

Autonomous game balancing has been explored mostly through the use of autonomous agents as presented in Politowski et al. 2023, where using bots allows the development team to see how the game would be played and adjust it accordingly without the need for human testers. This is a good approach for many games, however the human ingenuity and unpredictability is what breaks entities, meaning autonomising the players will not prepare the team for any eventuality. The authors of DeLaurentis et al. 2021 take this a step further and autonomise the full process. The bots play against each other and then further deep learning is used to assess the games. The main limitation of this approach is that for every new game this deep learning approach would have to be retrained on, certain games are more complex and could prove to not be able to be assessed and balanced this way. Lara-Cabrera et al. 2014 presents a framework where the terrain is automatically balanced with the player in mind using evolutionary algorithms, by taking in-game statistics which can be adapted from game to game making this a lot more easily adaptable than the previous work. The limitation of this approach, however is that it is only used to balance the terrain rather than the entities themselves making it only applicable to Real-Time Strategy games. There have been other studies such as Rupp et al. 2024 which have explored the same idea of evaluating the fun and balance of game levels through player surveys. Acquiring player feedback is important, but difficult at times meaning the limitation of this approach is the need for player feedback which is unable to be automatically assessed and adapted into the game.

Many works have explored the generation of balanced game entities such as the work done by Zaidan and Góes 2016, where a evolutionary algorithm balances hearthstone cards through simulations similar to the approach presented, and expanded upon, in this work. Game balance through generating new game elements as outlined in Babin and Katchabaw 2025 attempts to change the fundamental aspects of the game through reinforcement learning with the goal of improving

the balance while also only requiring the win rate as a variable. Some even generated Pokemon teams themselves (S. Reis, Novais, et al. 2023), similar to the approach proposed in this work with the same goal of optimising the balance of teams, within the same domain of Pokemon, however, the method is vastly different. The lack of the machine learning approach as well as the goal of diversity being exploration of different teams as opposed to heterogeneity of the metagame sets the works apart. The approaches above differ from mine in the key aspect of transformers coupled with a genetic algorithm working together to achieve the goal of heterogeneous balance. As seen, little work has been done on optimising the game entities used to play the game themselves for the purpose of creating a balanced and diverse metagame. Moreover very few works discuss the issues with autonomous balancing creating homogenous metagames and none address both.

2.4 Repercussions of an Unbalanced Game

In this section I will outline examples of entities that were widely considered to be unbalanced and infamous in their respective games and why they needed to be rebalanced.

- **Leroy, Tekken 7** - Tekken is a 3D fighting game with many different characters. These characters, much like Pokemon, have different moves that could be more or less balanced. Leroy was the first DLC (downloadable content) character and it was quickly apparent that he introduced imbalance into the game. Compared to other characters, Leroy had higher damage output and better frame data, which allowed him to have less risky attack patterns. The distaste for the character culminated ultimately in a Japanese Tournament called Evo in 2020[2]. At this tournament 6 out of the top 8 players in the tournament decided to choose Leroy, as seen in Figure 9, and many

[2]https://m.sports.naver.com/esports/article/439/0000019559?sid3=79e Last accessed August 21st 2025

players stated in interviews that: 'If you want to win, just pick Leroy'. This is an example of a character that was not properly adjusted before release. As a DLC character which came later in the game, the likelihood was the game developers wanted to see how the character was to be adjusted after players tested it. Unfortunately this period bled into one of the most influential tournaments, ruining it. Leroy was later rebalanced by increasing the risk factor in his combos (combinations of actions) and reducing his damage output greatly.

Figure 9: Tournament standings showing meta imbalance. (Tekken): The top 8 of Evo where Leroy took 6/8 spots ultimately winning the tournament.

- **PePe, Yu-Gi-Oh** - Yu-Gi-Oh (YGO) is a card game made and supported by Konami, where players summon creatures with special effects that allow them to summon more creatures. The goal of the game is to reduce your opponent's life points to zero. The game is infamous for having long combos,

with certain decks needing entire spreadsheets to understand different combo sequences. In 2015, a deck was released called Performapal Performages (PePe). In the 40 card deck, 16 of the cards were able to start the specific combo that led to an unbeatable board. A combo which the opponent could very rarely stop once established. A large number of tournaments in Japan and China had a vast over representation of PePe as seen in this metagame report[3]. Having a 60% representation across every tournament, as seen in Figure 10, signifies a clearly unbalanced entity that made the metagame revolve around it. A more common usage percentage for a metagames best deck is closer to 25% as seen in the meta of August 2025. [4] This prompted Konami to release an emergency banlist which banned 3 cards, that allowed the combo to be executed, from the deck, thus rebalancing it. Once again, a company wanted to make the new deck the best, possibly to incentivise commercial gain, an approach that they have continued to use throughout the life-span of this game.

- **Meta Knight, Super Smash Bros. Brawl** - Super Smash Bros. Brawl (Brawl) was a GameCube platform fighter game. Two players fight each other until one of them falls off of the stage 3 times. Meta Knight, a character that was fundamentally different than other characters in Brawl. In this game any time you ran you had a slight chance of tripping, where your character would fall and be vulnerable for a second. This meant many characters chose to approach slowly, which reduced their chance of tripping, or tried to jump in, which meant they cannot trip and be left vulnerable, however, being above your opponent meant that you were in a disadvantaged state which left you vulnerable to attacks. Meta Knight, however, had 5 jumps unlike every other character who had 2 jumps. Meta Knight also had

[3]https://roadoftheking.com/ocg-2015-10-metagame/ Last accessed August 21st 2025

[4]https://www.yugiohmeta.com/articles/tournaments/tcg/weekly-roundup/2025/august/4 Last Accessed 12th September 2025

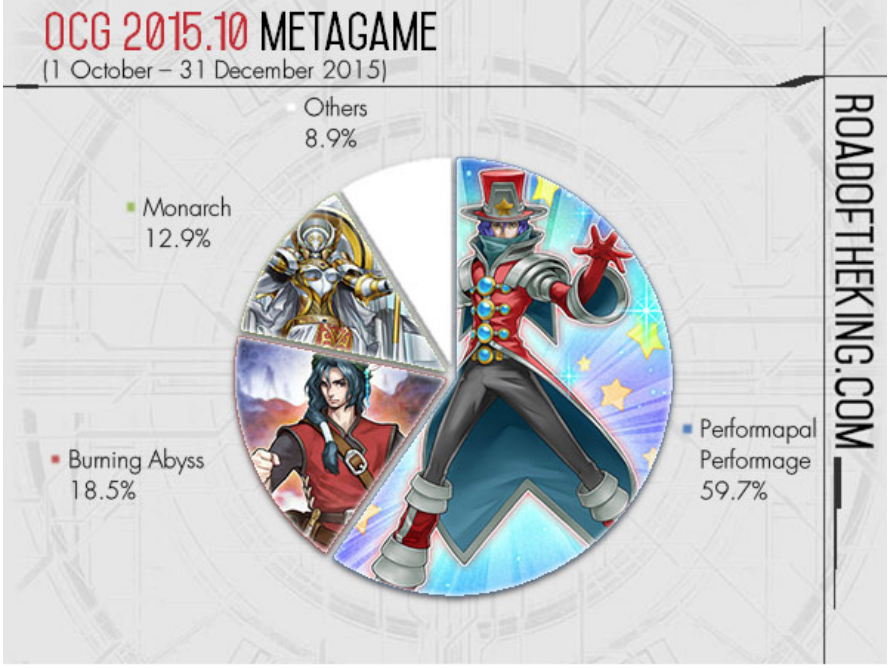

Figure 10: Usage percentage of a powerful deck depicting meta imbalance. (Yu-Gi-Oh): A graphic depicting the usage percentage of PePe in premier tournaments. 60% is a very substantial amount, as in more than half of every game you will see at least one person using the deck.

moves that covered every possible angle in any direction, whether he was airborne or on the ground. This caused Meta Knight to have a positive or neutral advantage with every other character in the game[5]. He was the only character that never had to interact with the ground if the player chose not to. Those aspects, mixed with Meta Knight's incredible damage output on his very safe and almost risk-free moves, meant he was dominating the top 8 at every major tournament[6]. many of these tournaments having 6 or 7 Meta Knights as can be seen in Figure 11. The community called to ban the character to preserve game integrity. 2005-2008 which was the time of Brawl, balancing was not possible since they would have to re-release the game with new patches (changes to improve the game); this was not possible

[5]https://www.eventhubs.com/tiers/ssbb/character/metaknight/ Last accessed August 21st 2025

[6]https://liquipedia.net/smash/Major_Tournaments/Brawl Last accessed August 21st 2025

as anyone that wanted to play would have to re-buy the game any time a new patch occurred. The reasons for Meta Knight not being banned have been lost to time, as most of the discussions were on forums that are no longer accessible, the general consensus is that best player in the world (Mew2King) stated he would stop playing if Meta Knight was banned. With the Japanese Brawl community agreeing with him, the character was left alone[7]. After the release of the new Smash Bros. game, Brawl was forgotten, largely due to Meta Knight, unlike its predecessor Smash Melee which is still played to this day. A game with no rebalancing will often encounter problems like these, which is why it is so important to have a dedicated development approach to fix these issues as they happen.

- **Juggernauts, League of Legends** - In League of Legends (LoL) a team of 5 players fights another team of 5 to destroy each other's base to win. There are 5 roles a player can choose from which all have their use in a team[8]. Usually you want a ranged attacker, a support character paired with the ranged attacker, an assassin that helps other teammates, a strong leading character and lastly a defensive character that is able to take many hits but struggles with dealing damage. In 2015, Riot Games put out a patch which buffed (improved) 3 characters, one of which was a character called Gankplank. This new patch heavily increased Gankplank's damage, which allowed him to be both quite defensive and deal the most damage on the team of 5. At the 2015 World Championship[9] Gankplank had a ban rate of nearly 100%, as seen in Figure 12, which meant every team disallowed the character from being used in their games (each team can ban

[7]https://www.ssbwiki.com/Meta_Knight_(SSBB)Most_historically_significant_players Last Accessed 12th September 2025

[8]https://leagueoflegends.fandom.com/wiki/Champion_classes Last Accessed 12th September 2025

[9]https://gol.gg/champion/champion-stats/26/season-ALL/split-ALL/tournament-World%20Championship%202015/ Last accessed August 21st 2025

Place	Prize	Player
1st	$1,455	Mew2King
2nd	$620	Dojo
3rd	$310	DieSuperFly
4th	$205	Lee Martin
		place 5 to 8
5th-6th	$205	Tyrant
		CO18
7th-8th	-	Domo
		Melee1

Figure 11: Tournament standings showing meta imbalance. (Smash Bros. Brawl): An example of a tournament dominated by Meta Knight; many were exactly like this. Some tournaments tried to ban Meta Knight for that tournament only. No official global ban was placed upon the character, however.

up to 5 champions). There were only 4 games where Gankplank was not banned. In the games where Gankplank was not banned, he was picked 100% of the time. In those 4 games, the players using Gankplank won all of them, almost single-handedly due to Gankplank's overwhelming damage. A 100% win rate at the highest level of play is unheard of in most, if not all, multiplayer games. This created a lot of controversy [10], which then

[10]https://www.reddit.com/r/leagueoflegends/comments/6le8zv/how_did_this_happen_the_balance_disaster_
Last Accessed 12th September 2025

caused Riot to adjust Gankplank's damage significantly. This showed that even huge games with massive development teams make mistakes, and could make use of autonomous balancing systems to help with their balancing decisions.

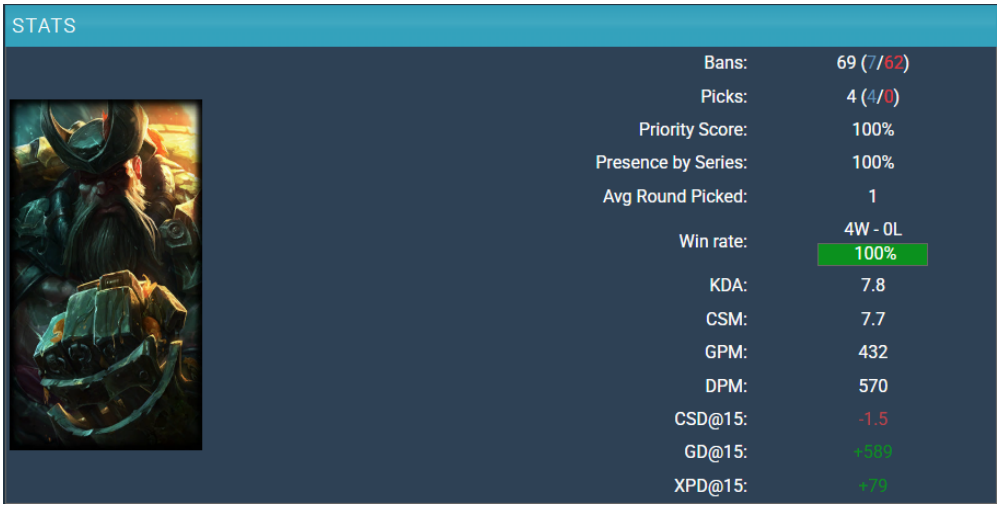

Figure 12: Tournament usage statistics depicting meta imbalance. (League of Legends): The statistic breakdown of Gankplank's performance at Worlds 2015. The fact that he had a 100% win rate, pick rate and priority score meant that the entire tournament revolved around this champion. This causes a lack of diversity and forces a team to either ban him or hope to have a chance to pick him and have a guaranteed chance to win.

- **Izzet Cauldron, Magic the Gathering** - A very recent example of an unbalanced metagame created through a lack of foresight can be seen in the Magic the Gathering (MTG) standard format. MTG is another example of a card game much like YGO where two players summon creatures and use spells to reduce the opponent's life point to zero. On June 30th 2025, a banlist[11] caused a lot of changes in the standard format, with many strong cards, that were causing the metagame to revolve around them, banned. This banlist removed one particularly strong deck that had 4 out of the

[11]https://magic.wizards.com/en/news/announcements/banned-and-restricted-june-30-2025 Last accessed August 21st 2025

8 spots in the top 8 of the August 9th 2025 major tournament in a deck called Izzet Prowess. In an attempt to balance the game, the development team decided to ban the main card that made this deck so strong. With Izzet Prowess gone, a new deck rose to success. Izzet Cauldron was too weak to compete with Izzet Prowess, but with that deck gone, alongside a couple other decks receiving adjustments, Izzet Cauldron was able to shine. In the Standard Arena Championship[12] Izzet Cauldron took 7 out of the 8 spots in the top 8, as seen in Figure 13. Due to having a very hard-to-interact-with gameplan, a lot of people have called for an emergency ban. Black Swan Events (unpredictable events that come as a surprise and has a major effect) (Nafday 2009) like these occur infrequently, but can cause catastrophic outcomes. After a balance patch or banlist, a grace period is needed for the metagame to stabilise. But if left too long without action, the player base can get frustrated with strategies that prove to be too strong and sales can take a big hit.

As seen in the above examples, balance is very important to both the players as they want to play a balanced and diverse game, and also to the developers, who want to keep their players happy and playing the game. The clear connection between an unbalanced entity and an undiversified metagame can be seen in all of these examples. As soon as there is an overarching best entity to use, the metagame quickly starts to revolve around it.

2.5 Measuring Diversity

In the examples seen in the previous section, one strategy caused the metagame to revolve around it, and caused a lack of diversity, which in turn made the game unbalanced. In this section I will describe ways of measuring this diversity in order

[12]https://www.mtgtop8.com/event?e=72302&f=ST Last accessed August 21st 2025

Figure 13: Tournament standings depicting meta imbalance. (Magic the gathering): UR Soul Cauldron in this graphic means Izzet Cauldron- it is another name for the deck. As seen here 7/8 of the top 8 is taken up by Cauldron, only one spot being taken by a different deck, which is specifically built to beat Izzet Cauldron.

to increase it. I will discuss some of the most widely used methods of measuring diversity, some that are particularly relevant and useful for this thesis.

Quality diversity (QD) uses local neighbourhoods in the search space, or niches, and these are used to perform the evaluation. The QD-score (Pugh et al. 2016), defined as the sum of fitness of the highest performing entities in each niche, combines the measure of individual fitness with the coverage of the search space. The goal of this measure is to fill a large number of niches with high-quality individuals to maximise coverage.

The sum of distances (SD) is another metric, this approach rewards the spread

of artefacts, however it does not penalise duplicates. This makes it a rather poor approach in this thesis as duplicate Pokemon and their sets are the main aspect I want to minimise. Duplicates in the metagame signify an unbalanced and undiversified metagame with centralising Pokemon.

A more narrow approach to measuring diversity in games has been explored in works such as Perez-Nieves et al. 2021, but each game is so different that most research in the area decide on using pre-existing measures of diversity rather than creating ones for specific games.

Lastly, the metric I ultimately decided to use for this thesis is the Vendi Score (VS) which is specifically designed as a measure of diversity for a machine learning context (Friedman and Dieng 2023). VS does not rely on class separation of entities, it instead uses distances between the artefacts to measure the diversity of the whole set. The VS is defined as the exponential of the Shannon entropy of the eigenvalues of a normalised similarity matrix.

$$VS(\mathbf{K}) = exp(-\sum_{i=1}^{N} \lambda_i log \lambda_i)$$

The ease of use and adaptability of this method makes VS the ideal measure of diversity for many tasks. I believe it is also ideal for Transformer generated content due to the tokenisation of the data which I will describe and explain in the next section.

2.6 The Transformer

In this section I will describe the Transformer architecture in detail, its structure, how learning looks alongside the benefits and limitations of using it for various

tasks.

2.6.1 Tensors

Tensors are central to many areas of mathematics and physics and have become the primary data structure in modern machine learning and deep learning frameworks including Transformers. A tensor is a mathematical object that generalises the concepts of scalars, vectors, and matrices to arbitrary dimensions. Formally, a tensor is defined as a multidimensional array of numerical values organised along one or more axes (also termed dimensions or modes). Their importance in machine learning comes from their ability to easily represent and manipulate data in a flexible and efficient way. The vast majority of current machine learning models use tensors because of this.

2.6.2 Tokens

```
[133, 941, 1089, 226, 331, 84, 86, 1, 119, 1, 1, 2015, 309, 652, 1098, 238]
['Abomasnow', 'Grass', 'Ice', 'Assault Vest', 'Brave', '248', '252', '0', '8']
['0', 'Snow Warning', 'Blizzard', 'Earth Power', 'Ice Shard', 'Aurora Veil']
```

Figure 14: Tokenised Pokemon example: An example of a tokenised Pokemon. The bottom 2 lines are usually conjoined, but were split for visual purposes. As can be seen in this Figure, each token is assigned a number based on a vocabulary and then later these are turned into tensors for the Transformer to train on.

A token is a piece of data that has been encoded as a number and later translated into a tensor. This is done so that machine learning models, which work using tensors, can be used in Natural Language Processing tasks. Previous work has encoded both words and parts of words as tokens, to be used for tasks such as translation (Sennrich et al. 2016).

2.6.3 Softmax

The softmax is a function that turns raw output scores into a probability matrix. This is done by taking the exponential of each output and normalising these values by dividing by the sum of all the exponentials. This turns the Transformer into a probabilistic model, which allows it to have a variety of answers to the same questions, as well as allowing the increase or decrease in diversity that is crucial in this thesis. In the Transformer, the probability matrix represents the probability of each token being the next one in the sequence. All of the probabilities in the matrix will add up to 1, for each row and each column.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

The terms in Equation 1:

- z_i = The element i in input z consisting of K real numbers.
- e^{z_i} = The exponential function applied to z_i making all values positive.
- z_j = The element j in input z used to find the sum of the entire input.

2.6.4 What is a Transformer

The Transformer is an example of a machine learning architecture. It excels at Natural Language Processing tasks as well as Time-Series data prediction and much more. The architecture is able to learn both the token representation of the data that is put into it and the positional representation. This allows it to learn the importance of the placement of tokens, as well as their meaning. This is unlike other models that it is often compared to, such as Recurrent Neural Networks and Long-Short-Term-Memory models.

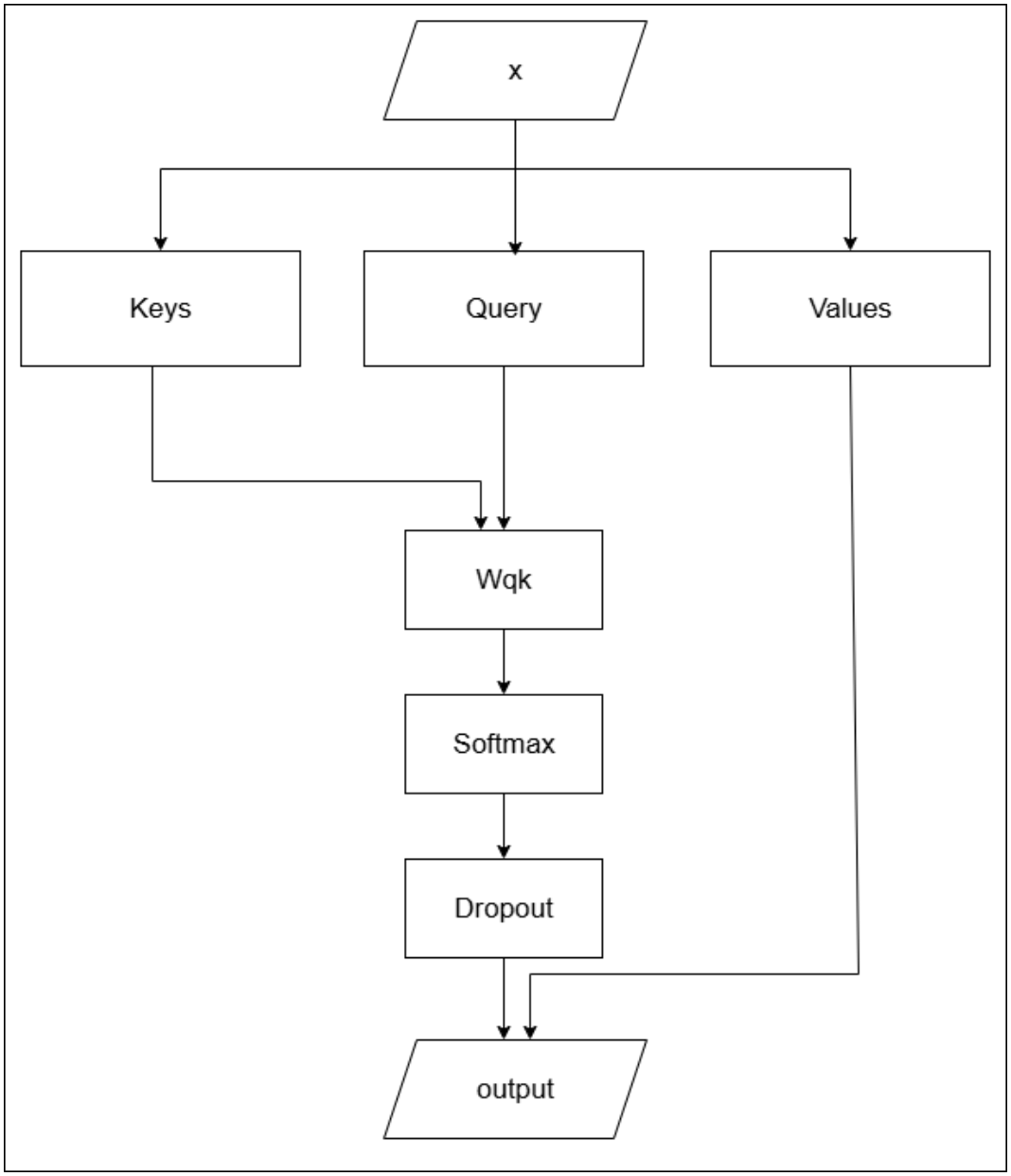

Figure 15: Attention head: A graphical representation of a single head of attention. Input x is parsed into Keys, Query and Values. W_{qk} are attention weights which are then put through a softmax to give the probabilities of each token being output. A systematic Dropout is applied to prevent overfitting, that is then combined using matrix multiplication as the output of an individual head.

2.6.5 Structure

Attention heads are responsible for assigning attention scores to the data that is parsed through them. Using these attention scores, the Transformer is able to discard the data that is considered noisy or less useful for learning.
The structure of an Attention head is as follows:

- A linear layer used for the queries of size E,H, where E represents the amount of embedding neurons and H represents the head size, both hyper parameters that are chosen. Seen in Figure 15 in the middle after the output.
- The Query and Values are linear layers of size E,H and are duplicates of the query as seen in Figure 15 as the second layer.
- These are followed by a softmax function that creates a probability matrix and at the end a dropout is applied. Dropout works by turning certain nodes off during training to prevent the network from becoming reliant on certain nodes and force the rest of the network to be trained.

The Attention heads work by parsing data x through the Key layer giving K and the Query layer giving Q. They are both matrices. K is transposed, and the values are exponentiated by -0.5, and then multiplied by Q giving W_{qk} as seen in Figure 15. A mask is applied to remove any illegal connections. W_{qk} is parsed through a softmax function which gives a matrix of probabilities that represents the likelihood each token has of being the next one in the sequence. A dropout is applied, removing certain parts of the matrix to reduce overfitting. The initial datapoint x is now parsed through the Value layer giving V which is then multiplied by the probability matrix given by the softmax and returned as the output.
The structure is as follows:

- A module list of size N of Attention heads, where N denotes number of

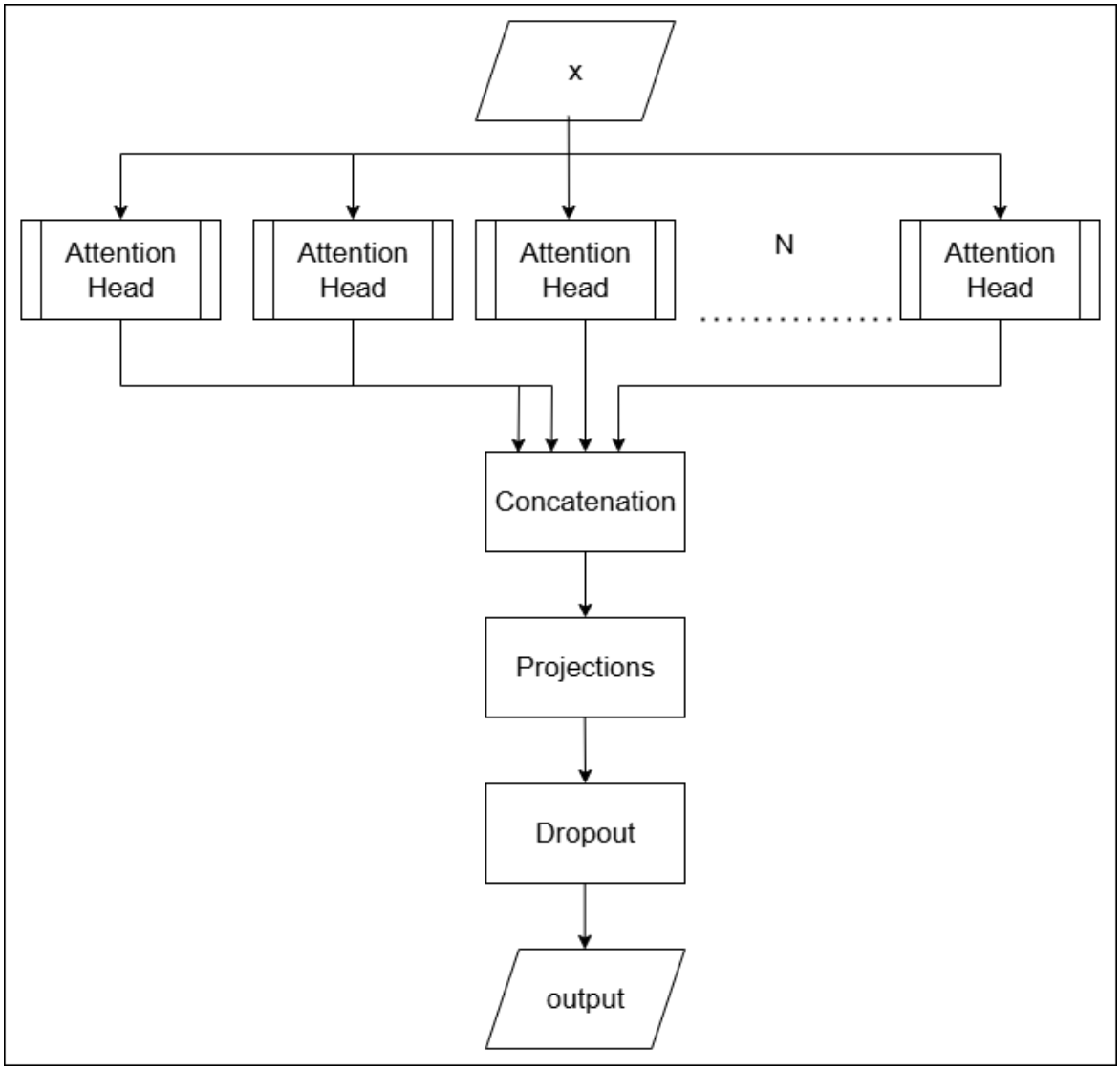

Figure 16: Multi-headed attention example: A graphical representation of the Multi-Headed Attention mechanism. Input x is parsed through each Attention head and later concatenated together. The output of the concatenated attention scores is parsed through the projection layer, which is a linear layer. A systematic Dropout is applied to prevent overfitting and then it is output.

heads, which is a hyper parameter. This can be seen in Figure 16 as the second layer.

- This is followed by a linear layer of size $H \times N$,E (Projections).
- This module is finished with a Dropout layer to prevent overfitting.

All of the Attention heads are concatenated and have data x parsed through them. This is followed by parsing the data from the multiple heads into the Projections layer which then has dropout applied and returned as the output. Multi-Headed Attention allows the Attention heads to be trained in parallel, greatly increasing the efficiency of the Transformer and setting it apart from other architectures that are used for the same tasks.

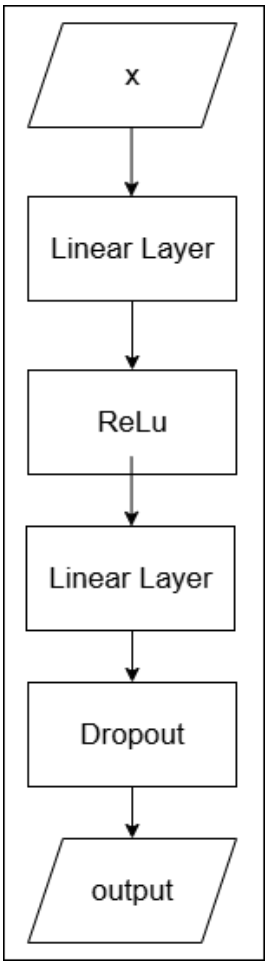

Figure 17: Feed forward network example: A graphical representation of the Feed Forward Network. Input x is parsed through a linear layer with the ReLu activation function. It is then parsed through another linear layer. A systematic Dropout is then applied to prevent overfitting and then it is output.

The Feed Forward network is a Sequential network with the following structure:

- Linear layer of size E,$4 \times E$.
- A ReLU activation function.
- Linear layer of size $4 \times E$,E.
- Lastly a Dropout layer.

The Feed Forward network can be seen in Figure 17 and inside a Transformer serves a crucial role in enhancing the model's representational capacity. While the self-attention mechanism allows each token to incorporate context from other tokens, the Feed Forward network operates independently on each token to perform non-linear transformations that help the model learn richer features. It works by parsing data x and returning the output of the network. The ReLu activation function causes the network to output x whenever x is higher than 0 and zero when it is not. ReLu is a very common activation function and is given by this formula:

$$\text{ReLU}(x) = \max(0, x)$$

The Block module is used to put the previously created structures together.
The structure of the Block module is as follows:

- The Multi-Headed Attention structure of size N,(E//N).
- The Feed Forward structure of size E.
- Two Layer Normalisation Layers both of size E.

The Block module can be seen in Figure 18 and works by calculating the attention scores of data x that was parsed through the first Normalisation Layer and

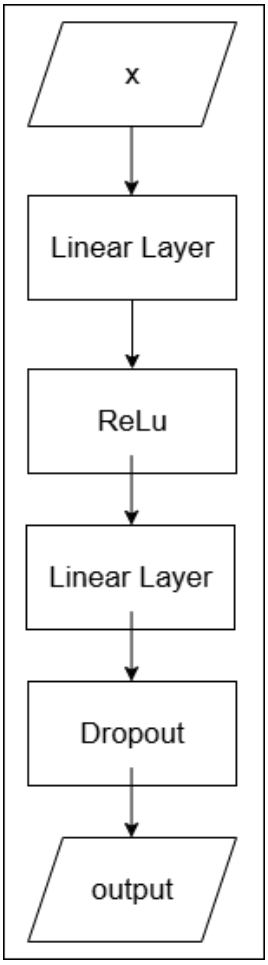

Figure 18: Block example: A graphical representation of the Block module. Input x is parsed through Layer Normalisation to normalise the output of the previous component. This is then parsed through a Multi-Headed Attention module, the output is concatenated with original input x. This is parsed through another Layer Normalisation then parsed through the Feed Forward Network. The output is concatenated with original input x and then output to the next block or the final Layer Normalisation.

concatenating it with data x, giving X. X is then parsed through the second Normalisation Layer, the output of that is parsed through the Feed Forward network and concatenated with X to give the final output which is then returned.

Now that every component of the Transformer has been outlined, the full structure, which can be seen in Figure 19, of the Transformer is as follows:

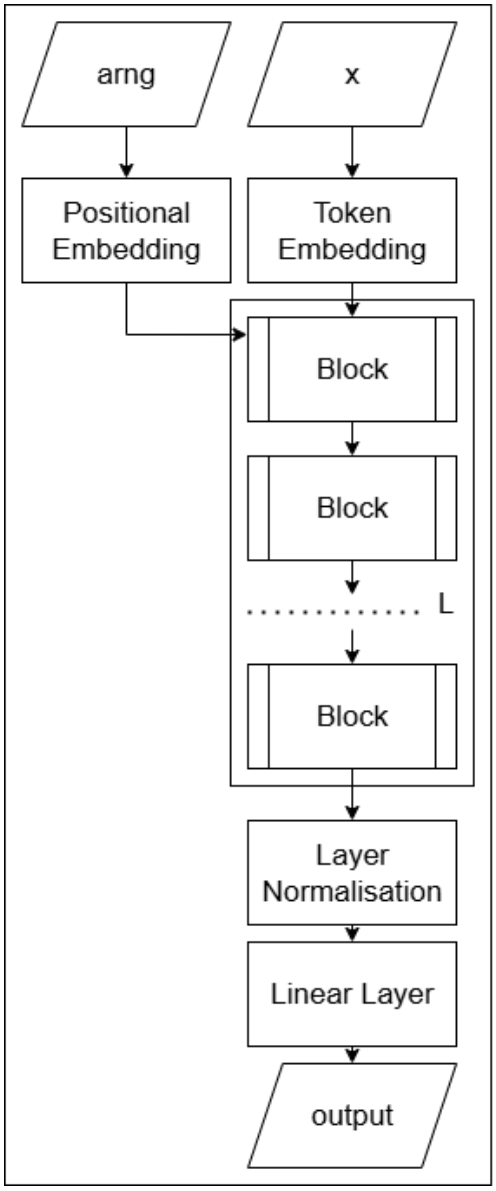

Figure 19: Transformer example: A graphical representation of the fully built Transformer. Input x is first parsed into the Token Embedding Table. An array is arranged that is used to determine the position of each token in the sequence, this is then parsed into the the Positional Embedding Table. The output of both embedding layers are concatenated and parsed into the sequential arrangement of block modules. The block modules parse the data into the next block. The last block parses the data into a Normalisation Layer, this is then parsed into a linear layer and output.

- First a Token Embedding Table is created of size V, E, where V represents the size of the vocabulary. This is followed by a Positional Embedding Table of size B, E where B is your block size which is the amount of tokens that are being learned in parallel and how much the Transformer will use as context.
- Blocks which are structured as a Sequential layer of length L, which is a hyperparameter responsible for the amount of layers to be created.
- Layer Normalisation of size E.
- Linear layer of size E, V.
- First input x is parsed into the Token Embedding Table, and concatenated with an arranged array, of size T which is the second dimension of the shape of the input. The input x is then parsed through the Blocks, here pruning occurs and is explained in section 3.3. The output of the blocks is parsed through Layer Normalisation, which is then parsed through the linear layer.
- Cross entropy loss is then calculated between the output and the next token in the sequence for each Attention head.

2.6.6 Learning

The Transformer learns by breaking the training data down and giving it to the multi-headed attention mechanism simultaneously in batches. This allows the Transformer to learn a large amount of data in parallel. Given the data, the Transformer will give a softmax matrix of probabilities, each probability corresponds to a token. The token is sampled from the probability matrix, and then a cross-entropy loss is calculated between the picked token and the target token. This loss is calculated for every Attention head that picked a token, which allows for very fast learning.

2.6.7 Inference

To generate something using a Transformer you require a context. A context is an initial set of tokens that the Transformer uses as a start of the sequence to generate more. The Transformer takes the context and, given then tokens in the context and the positions of those tokens, the next token is generated. Then using the context plus the newly generated token, the next token is generated and repeated for M, which denotes the maximum amount of tokens to be generated.

2.6.8 Why the Transformer

The Transformer is one of many machine learning-based approaches to both content generation and time-series data analysis and prediction. For example, Recurrent Neural Networks (RNN), Long-Short-Term-Memory(LSTM) models and Markov Chains are all systems that can be used for time-series data. The main benefit of Transformers across those models is the greatly decreased training time due to the ability for the Attention head to be trained in parallel. This allows the attention mechanism of the Transformer to learn complex relations between the tokens and the importance of their positioning (Karita et al. 2019, Lakew et al. 2018, (Zeyer et al. 2019)).

Architectures that have been largely replaced by Transformers:

- RNNs which work by sequentially taking data points and using them through recurrent connections to influence the next data point. The main benefit of this network is that it is able to learn small connections between tokens, making it suitable for time-series data. The main drawback is that it forgets data points quite quickly, making it unsuitable for large strings of connected data. This means in NLP tasks where there are large sequences of related text, Transformers outperform RNNs by being able to remember a much larger context window.

- LSTMs which work similarly to RNNs, but they have memory cells which allow them to have short term memory, much like RNNs, but also long term memory that allows the architecture to remember larger sequences. It however gets outperformed by the Transformer due to being unable to be trained in parallel, much like RNNs. The Transformer is also able to learn positional representation which the other architectures do not, this gives it an edge and allows it to learn and preserve the importance of token positions.

- Diffusion models are an example of another generative architecture, they learn how to de-noise pictures which allows them to generate pictures from noise. This technique has now been adapted to work with Transformers by Peebles and Xie 2023.

RNNs and LSTMs were state-of-the-art before Transformers in NLP and time-series tasks. The big problem that Transformers are able to combat is the vanishing gradient problem detailed in Bengio et al. 1994 that comes from the networks losing data over time. The network is unable to store all the data parsed through it over time, thus the longer a sequence is, the less relevant it will be to the beginning of the sequence. Transformers can have an arbitrarily long context window that stores the relevant information. This allows the Transformer to stay on the original topic even after millions of tokens. The Transformer also outperforms the previous state-of-the-art models by parallelising learning and having substantial batches of data processed at the same time. This greatly reduces training time.

For content generation, other approaches include Diffusion models, which sometimes can be Transformer-based (Peebles and Xie 2023), and Generative Adversarial Networks. These have their own merits, however, are outclassed by the Transformer due to parallelisation during training and the attention mechanism, as previously mentioned, and the ease of scaling-up Transformers.

However, the Transformer architecture has its drawbacks. While the training can be parallelised due to taking large batches of all the data at once, inference needs to be sequential, much like an RNN, which causes higher inference time. The high inference time is due to the inability of the Transformer to generate two or more tokens in advance. This means it must go through the very large model many times with increasingly larger contexts, which can take a long time.

By far the biggest drawback of Transformers is hallucinations, which are caused by the softmax probabilities. A token that does not semantically fit within a given sentence may still receive a relatively high probability due to anomalies or idiosyncrasies in the training data, whether stemming from noise or even legitimate but rare patterns resulting in its occasional, yet unexpected, generation by the model. This can cause the system to spiral and start generating nonsense. A lot of work is being done on the prevention of hallucinations (Zhang et al. 2025, Wu et al. 2025).

2.6.9 Tokenisation

Transformer networks have been proven to reliably learn contextual and positional relationships between datapoints, which puts them ahead of other state-of-the-art architectures. Transformers require the data to be tokenised by encoding them with a vocabulary.

The individual characters of a text can be used as tokens. This improves the generalisation of the model and prevents overfitting, however it also subjects the network to generating words that do not exist. If, however, the words being generated need to be correct at all times and are very specific, tokenising words is also possible.

2.7 Neural Network Pruning

Pruning is a technique used in many parts of machine learning. Removing decision tree branches that are deemed not important is an example of this. Pruning parts of a network involves setting the weights of a layer to 0 or not including it in the calculations by temporarily removing it (Shim et al. 2021). Pruning parts can give an understanding of what these parts are responsible for doing when they are present. It also increases the efficiency of a model by removing computations that would need to be calculated. In a Transformer network, it is possible to prune many parts without greatly affecting performance. These parts are as follows:

- Multi-layered perceptron (MLP) layers such as the feed forward layer (He et al. 2024)
- Attention layers (He et al. 2024)
- Block which encapsulates both the MLP and Attention layers (He et al. 2024, Shim et al. 2021).

As outlined in He et al. 2024, attention layers display the largest amount of redundancy, which in turn increases homogeneity in what is generated. The work also proved pruning blocks is the most impactful and useful. Further to their work which involved pruning after training, they outline the potential for future work exploring the possibility of retraining models while pruning, which could affect the Transformer architecture further.

Shim et al. 2021, He et al. 2024, Yang et al. 2024, Huang et al. 2024 and Lu et al. 2024 describe pruning the various parts of the Transformer, such as the layers or blocks of attention, to optimise training and reduce computation. These methods have been successful in reducing computation time while not impeding the results, by excluding unnecessary parts of the model in the inference step.

Layer-pruning is used as of now to decrease computational time and increase the efficiency of models (Peer et al. 2022 Kwon et al. 2022, Han and Tang 2025, Kim et al. 2022). My work however focuses on increasing the diversity of what is being generated much like the work of Peeperkorn et al. 2024. To my knowledge this work is the first to investigate if pruning layers of a Transformer can increase diversity.

2.8 Genetic Algorithms and Multi-Objective Optimisation

In this section I describe Genetic Algorithms, how they can be used and their purpose in this thesis. I also describe and explain the two approaches used in this thesis for multi-objective optimisation for specifying the fitness function for my Genetic Algorithms.

2.8.1 Genetic Algorithms

A Genetic Algorithm (GA) is an iterative search and optimisation method, which works with a population of individuals, where each individual is a candidate solution to a given problem, and at each iteration (called a generation in a GA), individuals are evaluated by a fitness function (which measures the quality of their candidate solution), and then individuals are selected with a probability proportional to their fitness, and selected individuals undergo operators like crossover and mutation to generate new individuals.

2.8.2 Single Point Crossover

Single point crossover works on the principle of selecting one point in two parents and swapping the bits at those points, thus creating two new offspring which can

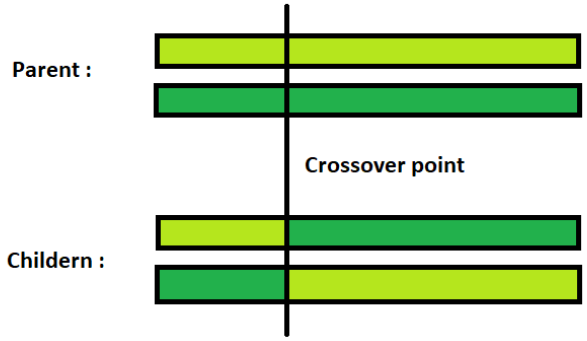

Figure 20: Crossover example: The parents can be seen at the top and the generated offspring can be seen at the bottom.

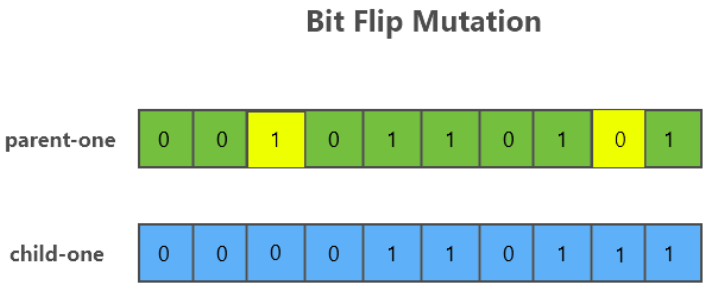

Figure 21: Bit flip example: The parent can be seen at the top, the highlighted bits are ones that will be flipped in the offspring which can be seen at the bottom.

be seen in Figure 20.

2.8.3 Bit Flip Mutation

Bit flip mutation works by changing a bit's value from 0 to 1 or from 1 to 0, which can be seen in Figure 21.

2.8.4 Tournament Selection

Tournament selection works by creating small groups from the dataset. In these smaller groups the individuals compete against one another - individuals with

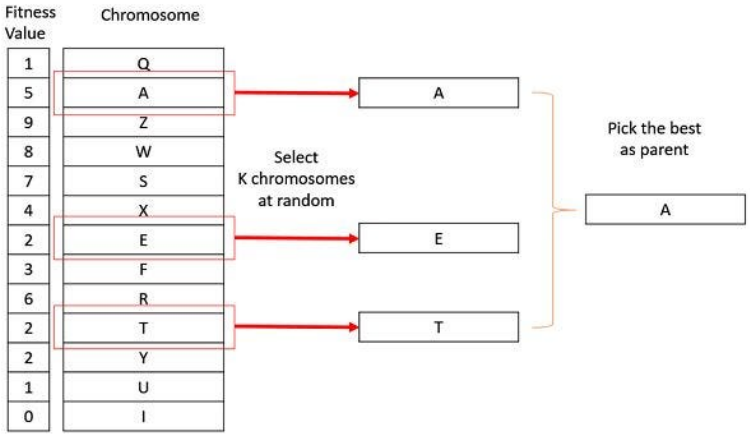

Figure 22: Tournament selection example: The chromosomes (individuals) that are highlighted are the ones randomly chosen as the smaller group to compete against each other. In this example, individual A had the highest fitness which meant it was selected as the next parent.

higher fitness win and are then chosen as one of the parents, as seen in Figure 22.

2.8.5 Multi-Objective Optimisation

Multi-objective optimisation (MOO) refers to an algorithms that tries to find a solution that either maximises or minimises a number of objectives simultaneously. It is important to this thesis due to having two objectives that need to be maximised simultaneously - balance and diversity. Two common approaches to MOO are Pareto Dominance and Weighted Sum.

2.8.6 Pareto Dominance

The Pareto Dominance (**fandel_tutorial_2004**, Fonseca and Fleming 1995) approach is primarily used for two objectives, as the number of non-dominant solutions increases quickly, which makes it more difficult to get good results. Those objectives act as axes on which the algorithm will place various solutions. Pareto Dominance works as follows: A solution s_1 dominates a solution s_2 when the

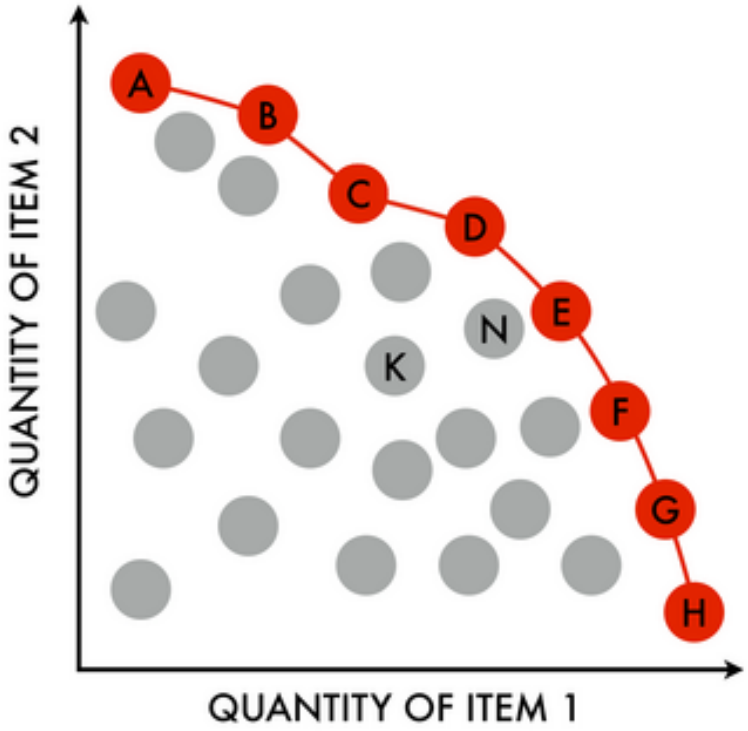

Figure 23: Pareto front example: An example of the Pareto Dominance approach with two objectives to be maximised. The grey dots are dominated solutions and the red dots indicate the Pareto front which are the non-dominated solutions.

following two conditions are satisfied: (a) s_1 is not worse than s_2 with respect to all objectives; and (b) s_1 is better than s_2 with respect to at least one objective, see Figure 23. A solution s is said to be non-dominated when no other solution dominates s. The Pareto front consists of all non-dominated solutions. The Pareto front consists of solutions deemed efficient. Solutions on the Pareto front balance both of the objectives in an optimal way. A solution needs to be selected from the Pareto front, this is usually done by calculating the hypervolume of the Pareto front and removing solutions to see their impact on the hypervolume and picking one that maximises the impact.

2.8.7 Weighted Sum

The Weighted Sum approach is another multi-objective optimisation approach, which can be used for any number of objectives. By assigning arbitrary weights to each objective and combining them into a single number, the Weighted Sum approach maximises or minimises a single number for each candidate solution. Unlike Pareto Dominance where two or more objectives are taken into account simultaneously, Weighted Sum compresses this problem to just one number to optimise. It relies on the human to select the appropriate weights for the objectives. This makes Weighted Sum have a degree of error on the human part and may need optimisation itself. The weights can be difficult to choose in some scenarios, which can make Weighted Sum quite inconvenient. It can greatly underperform if the weights are not carefully chosen. This is not the case for Pareto Dominance which treats the objectives separately, this therefore makes it the more desirable approach when appropriate weights are not easy to identify.

2.9 Significance Tests

In this section I explain what a significance test is, what it is purpose is for this thesis and explain the specific test I used to evaluate my results.

2.9.1 What is a significance test and its purposes

A statistical significance test (Cox 1982) is a formal analytical procedure used to determine whether the observed effect or difference in a dataset or set of results is unlikely to have occurred by random chance alone. Its primary purpose is to assess the strength of evidence against a null hypothesis, which typically represents a baseline assumption such as having no effect or difference. By calculating a p-value, the test quantifies the probability of observing results at least as extreme as those in the data, assuming the null hypothesis is true. If this p-value falls

below a predetermined threshold (commonly 0.05), the result is considered statistically significant, suggesting that the observed pattern is unlikely to be due to random variability. In empirical research, statistical significance testing is critical for validating hypotheses, guiding inference, and supporting conclusions drawn from experimental or observational data.

2.9.2 Wilcoxon Signed-Rank Test

The Wilcoxon signed-rank (Woolson 2008) test is a non-parametric statistical significance test used to compare two models on multiple datasets. The main advantage of being a non-parametric test is that it makes no assumption of normal distribution, unlike the paired t-test.

The null hypothesis for this test is that the medians of two models' performance, in this context generative performance, are equal.

To perform a Wilcoxon signed-rank test, the difference (D_i) is calculated between two data points ignoring their signs. The tied absolute differences are assigned the average of their respective ranks. Any pair in which $D_i = 0$ is excluded from the analysis. The absolute values $|D_i|$ are ranked in ascending order.

The test statistic W is the sum of the signed ranks calculated as such:

$$W = \sum_{i=1}^{n} sign(D_i) \cdot R_i \tag{2}$$

where R_i denotes the rank of $|D_i|$.

W represents the measure of how strongly the differences between paired samples lean in one direction and is used in calculating the p-value that ultimately denotes significance.

2.10 Literature Review Conclusion

To conclude the Literature Review, I explored different aspects of balance as well as its importance in games. I presented a background on Pokemon to help visualise the problem. I spoke about the metrics of diversity used in the current landscape that are more relevant to my field. I explained the Transformer architecture, how it works and its benefits over other architectures that could be used for the same task. I discussed genetic algorithms as well as the various approaches to multi-objective optimisation. Lastly I discussed significance tests which are paramount to statistically prove my findings to be important. In the next chapter I demonstrate how I performed the experiments to explore and achieve my aims.

Chapter 3

Methodology

In this section I describe how I used the Transformer to generate Pokemon, the purpose of the Transformer's hyperparameters and how they were set in the experiments, and the approaches used for the tokenisation and training procedures. Then, I describe the dataset used in the experiments. Next, I describe the two proposed methods that are the research contributions of this work, namely: (a) the proposed layer pruning approach, which prunes inconsequential Transformer layers during training and retraining; and (b) the proposed Genetic Algorithm for Pokemon selection. Recall that these methods are designed for achieving the overall aim of this research, i.e., improving the diversity and balance of a Pokemon metagame generated by a Transformer model. Finally, I describe how a statistical significant test was used for analysing the results of the experiments.

3.1 The Transformer's Setup for the Experiments

In this section I describe the Transformer in these experiments that was used to generate Pokemon and how it worked. I describe my choices for tokenising the Pokemon. I describe and explain the training and retraining of the Transformer that is used to guide it towards creating more balanced Pokemon. Lastly I describe

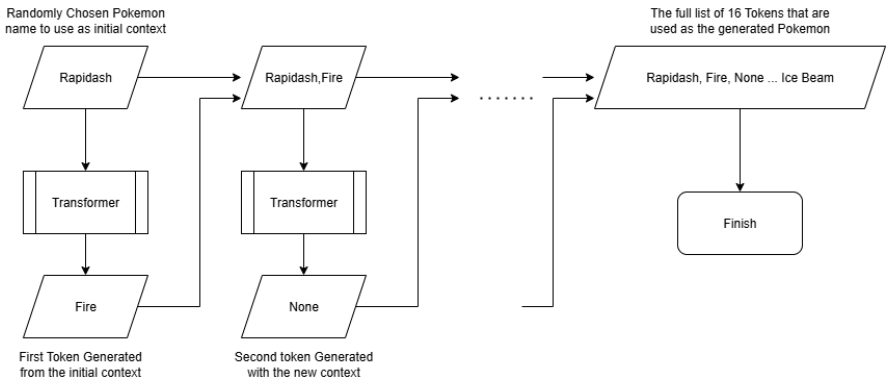

Figure 24: Process of generating Pokemon from the transformer: The process of generating a Pokemon. First a Pokemon name is input initially into the Transformer. The Transformer then sequentially generates the next tokens until 16 are generated, at which point the Transformer stops and the structure of the generated Pokemon is checked.

the hyperparameters of the Transformer used in these experiments.

3.1.1 Transformer to generate Pokemon

In this thesis the Transformer is used to generate a Pokemon by using the name of a Pokemon as a context. A single, randomly selected token that denotes a Pokemon name is chosen and used as the input. The Transformer then assigns this context an attention score and generates a probability matrix of all tokens that can be the next in sequence using the softmax function. In the first pass, the Transformer will give bigger probability to the type, as that is what follows after the name of a Pokemon (see Section 3.2). After the next token is chosen, this process is repeated with an increased context given to the Transformer as the Pokemon is sequentially created. Once 16 tokens are generated and the order and structure of the tokens is in line with the structure of a Pokemon, the Pokemon is added to the dataset to battle against other Pokemon generated this way.

The system is very precise and requires a specific format for the Pokemon to be in. To prevent erroneously generated Pokemon from being considered, the system

automatically discarded any Pokemon not in the correct format. An incorrectly formatted Pokemon is one with incorrectly ordered variables, such as an ability in the slot of a move. The Transformer later learned this format and was able to reproduce it almost flawlessly after some number, on average around the 10^{th}, of iterations of retraining, thus showing the Transformer's capability of becoming more deterministic through retraining while still providing diverse output, as illustrated in Table 1.

Table 1: Transformer learning example: Depiction of how the Transformer learns the structure of Pokemon over time through retraining.

Iteration	Number of erroneous Pokemon generated
1	525
2	218
3	82
4	68
5	61
6	35
7	8
8	8
9	8
10	7
11	5
12	5
13	2
14	3
15	1

200, correctly structured Pokemon were generated at the end of every iteration of retraining. The Pokemon were generated by giving the context of a random Pokemon name. The next 15 tokens were then chosen by the Transformer and the full sequence was used as one Pokemon for simulations. Any erroneous Pokemon were discarded. This was then repeated until the dataset was filled with 200 correctly

structured Pokemon.

3.1.2 Tokenisation

In these experiments every word and number, such as a move and EV (which are variables attached to each Pokemon that determine their prowess in certain aspects such as physical attack or defence and stand for Effort Values, see Section 2.2) respectively, is tokenised individually. This enables the network to avoid generating words that cannot exist, for example an ability that is misspelled, unlike networks that tokenise each individual letter and symbol. Tokenising full words also reduces the size of the network, as the network scales with vocabulary size. A smaller vocabulary also decreases inference time, due to a decrease in the number of passes through the network. A Pokemon is made up of 16 tokens as described in section 3.2.

3.1.3 Training and retraining

To train the Transformer I first specified whether the model will use pruning during training and retraining or not. In addition, I chose which approach is used for selecting a set of 100 Pokemon out of the set of 200 Pokemon generated by the Transformer, in order to create each new retraining dataset, for each iteration of retraining. I used three selection approaches: random selection and two versions of a multi-objective Genetic Algorithm (GA) for performing Pokemon selection, namely a GA with a weighted-sum fitness function and a GA with Pareto Dominance (see Section 3.5). Then the Transformer is trained on the dataset outlined in Section 3.2. The different experiment configurations are as follows:

- No Pruning, Random Selection.
- No Pruning, Pareto Dominance.
- No Pruning, Weighted Sum.

- Pruning, Random Selection.
- Pruning, Pareto Dominance.
- Pruning, Weighted Sum.

After 3100 iterations of training I stopped each model's training. This was to prevent the model from overfitting. I split the dataset into a training and a validation dataset (which the model does not train on and is instead used to check if the model is able to generalise to problems it has not seen) with a ratio of 90:10. The number 3100 was chosen because each model began to have an increase in validation loss after that, and to remove fluctuations between models and minimise the chance for a model to overfit, which would greatly impact the diversity of the output. An iteration of retraining is defined as follows:
After the initial training of the model, I generated a dataset of 200 Pokemon. I then used these 200 Pokemon to battle against each other, this is further described in section 3.4. By extracting the win rate of each individual Pokemon, Then one of the 3 aforementioned selection methods was used to create the retraining dataset of 100 Pokemon that is used as new input for the same Transformer and trained for 100 iterations. A small number of iterations was chosen due to the very small size of the dataset and to avoid overfitting on this small dataset.

3.1.4 Hyperparameters of the Transformer

The hyperparameters used are as follows:

- Learning rate = 3e-4
  - A relatively low learning rate chosen through trial and error. A low learning rate increases the training time needed to converge on an optimal solution. I aimed to minimise convergence through the learning rate and layer-pruning.

- Batch size = 64
  - The batch size determines how many sequences will be processed at once by the Transformer during training. The number 64 was chosen to optimise training time and is quite standard (McCandlish et al. 2018). The dataset used for training is quite small and the retraining datasets are very small, making 64 a appropriate candidate.
- Context size = 16
  - Context size determines the sequence length the Transformer is able to remember. The number 16 was chosen since that is the length of a Pokemon in the dataset. I did not want Pokemon to influence each other, which could occur with a larger context window.
- Number of Embedding Neurons = 400
  - This is a number chosen through trial and error. The number of embedding neurons increases the size of the network - with a small dataset, the network could overfit much quicker while having increased inference and training time.
- Number of Layers = 5
  - This is a number chosen through trial and error. A larger number of layers drastically increases model size, which meant that with a small dataset I needed to keep this number low. The use of the proposed layer-pruning approach required that the Transformer have enough layers so that there is opportunity to prune some layers and the unpruned layers can be effectively used for training. With a larger dataset, this could be increased which could drastically improve performance and increase the need for layer-pruning.

- Number of Attention heads = 5
  - This is a number chosen through trial and error. Having many heads does not increase training time drastically due to parallel training, but increases homogeneity in layers, thus more layers are pruned. With a limited number of layers, I needed to keep Attention head count low.
- Dropout rate = 0.2
  - This is a common value for dropout rate (Hinton et al. 2012). Dropout rate values are usually between 0.2 and 0.5. Due to the focus on increasing diversity through layer-pruning, I decided to keep this on the lower end. This number denotes the proportion of neurons turned off at different stages of learning to prevent overfitting through random selection, by preventing the network from being overdependent on specific nodes.
- Pruning threshold = 0.02
  - This threshold was chosen through trial and error. If the absolute value of the Cosine similarity is 0.02, the layer is pruned and not used in the epoch. A large pruning threshold prunes layers more frequently, this would be beneficial with larger networks, however my network is relatively small with few layers. At the start of training, very few layers are pruned. The number of layers being pruned increases as the network overfits, which is ideal in these experiments as retraining will cause the network to converge.
- Vocabulary size = 1456
  - In the dataset there are 1456 different tokens. Each of these tokens

have a chance at being generated. The Transformer learns to generate specific tokens at specific positions over the course of retraining. The split between Types:Items:Natures:Abilities:Moves:EVs is 19:113:25:224:820:253 respectively.

3.2 The Dataset Used in the Experiments

For these experiments, a relatively small dataset of Pokemon was used. This dataset consisted of 6819 different Pokemon - it was the only dataset available that was this extensive and was designed with competition in mind. Most Pokemon datasets focus on the images of the Pokemon themselves alongside their descriptions [1]. The chosen dataset, instead, contains the most competitively used teams from various tournaments and events taken place between February and August 2022. Each Pokemon consists of 16 variables which are explained later in this section.

The data used for the experiments were taken from a Pokemon dataset[2] and arranged in the following format:

Pokemon Name, Type 1, Type 2, Item, Nature, Health, Attack, Defence, Special Attack, Special Defence, Speed, Ability, Move 1, Move 2, Move 3, Move 4

This format ensures that the Pokemon will be generated top-down, from the most important feature, typing, to ones with most variance, such as movepool.

Each individual, type, move, item, stat, et cetera was tokenised to prevent the Transformer from generating tokens that cannot exist, such as misspelled moves

[1] Most other datasets found on kaggle such as https://www.kaggle.com/code/ahmedayad20/pokemon have no competitive aspect to them and many focus on the images such as https://www.kaggle.com/code/trolukovich/predicting-pokemon-with-cnn-and-keras

[2] https://www.kaggle.com/datasets/giorgiocarbone/complete-competitive-pokmon-datasets-may-2022 Last accessed August 21st 2025

or types. Each Pokemon was filtered to have up to 508 EV points spread across the variables, and each EV can only have up to 252 points assigned, as described in Section 2.2.

The various variables in Pokemon are explained further with examples in Table 2.

3.3 Pruning

To systematically prune the Transformer's blocks that are not necessary, a Cosine Similarity is calculated between the input and the output of a block, Figure 18, as can be seen in Figure 25. If the Cosine Similarity was between the specified threshold of -0.02 and 0.02, it meant that the input and output are similar enough that the block could be ignored for this batch due to having an insignificant impact on the output and causing the Transformer to generate homogeneous Pokemon. Cosine Similarity is calculated using this equation:

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A is the input matrix of the block and B is the output matrix of the block. n = $batchsize \times blocksize$. i is the index of each datapoint in a mini-batch.

The similarity thresholds of -0.02 and 0.02 as lower and upper bounds were chosen through trial and error, and were kept uniform throughout all of the experiments. The higher the overall threshold, the more diverse the generated datasets would be; however, it also generates nonsensical Pokemon at higher thresholds, increasing inference time and becoming a lot more random, which I wanted to avoid.

Table 2: Pokemon variables: Each variable of a Pokemon. Many of them were explored further in section 2.2. All moves follow the same rule which meant I did not specify them individually to save space.

Name	Meaning	Data type	Example
Pokemon Name	Name of the Pokemon.	String	Gengar
Type 1	This is the first typing of a Pokemon, described in Section 2.2.	String	Ghost
Type 2	This is the second typing of a Pokemon.	String	Poison
Item	This denotes what item the Pokemon uses, described in Section 2.2.	String	Choice Specs
Nature	The nature allows a Pokemon to positively impact an EV increasing it by decreasing another EV the Pokemon needs less.	String	Modest (+SpAtk, -Atk)
Health	This EV increases the health of a Pokemon, making it harder to knock out.	Integer	4
Attack	This EV increases the damage of physical moves and is used for physical attackers.	Integer	0
Defence	This EV changes the damage taken from physical attacks, the higher the defence stat, the lower the damage taken from physical attacks.	Integer	0
Special Attack	This EV increases the damage of special moves and is used for special attackers.	Integer	252
Special Defence	This EV changes the damage taken from special attacks, the higher the special defence stat, the lower the damage taken from special attacks.	Integer	0
Speed	This EV determines the speed at which the Pokemon is able to move, the faster Pokemon will use its move before the opposing Pokemon that is slower.	Integer	252
Ability	Each Pokemon has a small pool of abilities to choose from.	String	Levitate
4 Moves	Each Pokemon has a pool of moves to choose from. The position/order of the moves does not matter. See section 2.2 for more.	String	Dazzling Gleam

3.4 Pokemon Battles

To find the win rate of each generated Pokemon, 10 random teams (P) of 6 Pokemon were created. Games were played using bots that maximised damage,

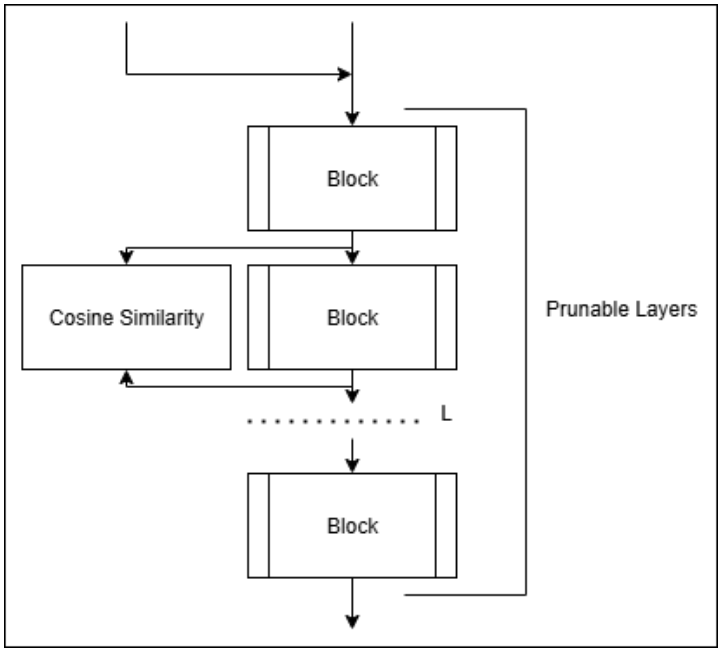

Figure 25: Pruning example: A graphical representation of the pruning procedure proposed in this thesis. At each pass of data through the network, for each block, a Cosine Similarity is calculated between the input and the output. If the similarity is below 0.02 and above -0.02, the output and input are too similar, thus generating homogeneous output. That block is not included in the pass and the input is fed into the next block where this process is repeated. The blocks are not permanently removed, but instead ignored for one pass.

so they prioritised moves with high damage output rather than defensive moves, like protect, or status effect moves, such as toxic. Each of the P teams battled against 5 other randomly chosen teams 10 times. The average win rate of each Pokemon was calculated over the course of their battles. Maximum damage agents were used for every battle to ensure integrity and to provide a system-optimised and uniform variable, that being damage. This allowed for a much smaller design space than if I used optimised agents, since all of the EVs would have to be taken into account by the system.

Pokemon battles consist of a team of 6 Pokemon, battling one at a time and having 5 actions to choose from - using one of their 4 moves or switching out to one of

the other Pokemon on their team. Moves can deal damage, affect the opposite Pokemon with status effects or EV changes, and by depleting the health of the enemy Pokemon, the Pokemon will be knocked out eventually and the Player will have to send a new Pokemon out. The loss is awarded to the bot with no Pokemon left to battle with. When the win rate is extracted, any Pokemon with a win rate between 45% and 55% is considered balanced.

3.5 Genetic Algorithms for Pokemon Selection

In this section I describe the proposed Genetic Algorithm (GA) for Pokemon selection, which was implemented in two different multi-objective optimisation versions (discussed in Section 3.5.1), and I also specify the parameter settings for the GA (in Section 3.5.2). In addition, Section 3.5.3 describes some hypotheses about the expected performance of the two multi-objective versions of the proposed GA.

Each individual, or candidate solution, consists of a vector of 200 bits, where each bit, or gene, is associated with a different Pokemon, and each gene takes the value 1 or 0 to indicate whether or not its corresponding Pokemon is selected by the algorithm. In addition, the GA enforces the constraint that, for each individual, exactly 100 bits are turned on (1), whilst the other bits are turned off (0), so that each individual always represents a selection of 100 Pokemon out of a candidate set of 200 Pokemon that were generated by the Transformer. The GA used single-point crossover and bit flip mutation as described in section 2.8

In this research, I investigated whether the type of multi-objective optimisation (MOO) fitness function used by the Genetic Algorithms impacted the results, therefore I implemented two versions of the fitness function, one using the Weighted Sum approach and the other using Pareto Dominance. In this thesis I use MOO to create a balanced dataset to use for retraining, with these objectives

to be optimised:

- Diversity, which I want to maximise over time and is calculated by the Vendi score which was precisely defined in Section 2.5.
- Balance, which I want to maximise over time and is calculated by counting the number of Pokemon with a win rate between 45% and 55%.

3.5.1 Multi-Objective Fitness Functions

The specific weights for the two objectives in the Weighted Sum approach were set to be 60:40 in favour of the balanced win rates.

Regarding the Pareto Dominance approach, selecting a solution from the Pareto front was done manually, prioritising amount of balanced win rates. However, if a solution was close to the best one by win rate, but had a substantial increase in diversity, was chosen instead.

An alternative approach would have been to calculate the hypervolume of the Pareto curve and observe the effect of taking away each solution. This would have been a more systematic approach to finding the best solution. However, due to time constraints and the scope of the project, I opted for the more simple, manual approach.

3.5.2 The GA's Hyperparameter Settings

The GA used standard single-point crossover with a probability of 1.0, standard bit-flip mutation with a probability of 0.5, and tournament selection with tournament size of 2 (a common value in the literature).

In the implementation of the genetic algorithm (GA) used in this study, both the

population size and the number of generations were set to 30. These values were selected based on a combination of empirical precedent, computational efficiency, and the principle of diminishing returns, which is commonly observed in evolutionary algorithms.

The choice of a population size of 30 reflects a balance between solution diversity and computational tractability. Larger populations generally promote greater genetic diversity, which can improve the GA's ability to explore the solution space and avoid premature convergence (Sastry et al. 2014). However, this benefit comes at the cost of increased computational overhead, especially in fitness evaluation, which is often the most resource-intensive component of the algorithm. The computational time of the genetic algorithms is outlined later in Chapter 4.

3.5.3 Hypotheses of the performance of the MOO GA approaches

My initial hypothesis is that the Weighted Sum approach will outperform the Pareto Dominance approach in maintaining balance over time.

This is due to the Weighted Sum approach having the weights favouring balance, prioritising it. The second hypothesis is that the Pareto Dominance approach will outperform the Weighted Sum approach in terms of diversity for the reason that it equally prioritises both objectives.

My final hypothesis is that Random Selection will produce unfavourable results in terms of balance, due to the fact it is not optimising it as an objective, instead it is selecting the Pokemon at random. This could however give it an edge in terms of diversity over the MOO GA-based approaches.

3.6 Significance Tests

In this section I describe how the Wilcoxon signed-rank test was used in this research. I used this test to compare the diversity and balance generated by two models. Looking to equation 2 in section 2.9, n is equal to 15 which denotes the number of iterations of retraining the models went through where the model generates Pokemon that are used for battle and later selected by the GAs to retrain the model on. The i-th data point is the individual result of both models at a specific iteration.

The Wilcoxon signed-rank test is a non-parametric statistical significance test. It is used to compare two sets of retraining at each iteration for two models. I use it to find which experiments result in statistically significant results in both diversity and balanced win rates. Typically the null hypothesis is rejected if the p-value computed by applying the test is smaller than the standard significance level (or threshold) of 0.05. However, due to the relatively large number of results (as seen in Table 4), I adjusted the significance level using the principle of multiple hypothesis testing correction (see below).

The Wilcoxon signed-rank test was used to compare every pair of configurations of the experiments, where a configuration consists of a choice of performing or not performing layer-pruning as well as a choice of one out of three selection methods (the two versions of the GA, Weighted Sum and Pareto Dominance, or random selection). Hence, the experiments involved 6 possible configurations, and so the test was applied 15 times (which is the number of possible pairs out of 15 configurations).

I divided the normal significance alpha level which is typically 0.05, by 15, which yielded 0.003333 - this process is called the Bonferroni correction (Japkowicz and Shah 2011) which is a conservative approach for multiple-hypothesis testing. I

rejected the null hypothesis using this new significance alpha level if the p value was below it, thus the result was considered significant. I calculated significance between the diversity and balance win rate results that each model got over time.

3.7 Methodology Summary

In this chapter I outlined the specific configurations of the Transformer used in my experiments. I described the way my dataset is structured and used as well as how the Pokemon battles work that aid in creating the retraining datasets. I also described the two methods which are the research contributions of this work, namely the layer pruning for the Transformer and a multi-objective Genetic Algorithm for Pokemon selection.

In the next chapter I describe the results that my methodology produced as well as how I evaluated them.

Chapter 4

Results and Evaluation

In this Chapter, I report computational results for the two methods proposed in Chapter 3 for improving the Pokemon metagames produced by a Transformer, namely: (a) layer-pruning and retraining; and (b) a Genetic Algorithm (GA), implemented in two different multi-objective versions, one with the Weighted Sum approach (WS-GA) for the fitness function and the other with the Pareto Dominance approach (PD-GA). The reported computational results show that, broadly speaking, these two methods achieved the overall aim of this research (specified in Section 1.2), namely improving the balance and diversity of Pokemon metagames generated by a Transformer.

This chapter is structured in terms of the two performance measures for which results are reported (balance and diversity of metagames), as follows. Section 4.1 reports and discusses the results obtained by the above two methods for the proportion of generated Pokemon which have balanced win rates. Section 4.2 reports and discusses the results for the diversity of the generated Pokemon. Section 4.3 discusses the results considering both the balance and the diversity measures as a whole, discussing the trade-off between these two measures. Section 4.4 discusses

examples of a good metagame and a bad metagame, in order to get further insights about the importance of having a balanced and diverse metagame. Finally, Section 4.5 presents a summary of the main results, contributions and limitations of this chapter.

4.1 Balanced Win Rates Results

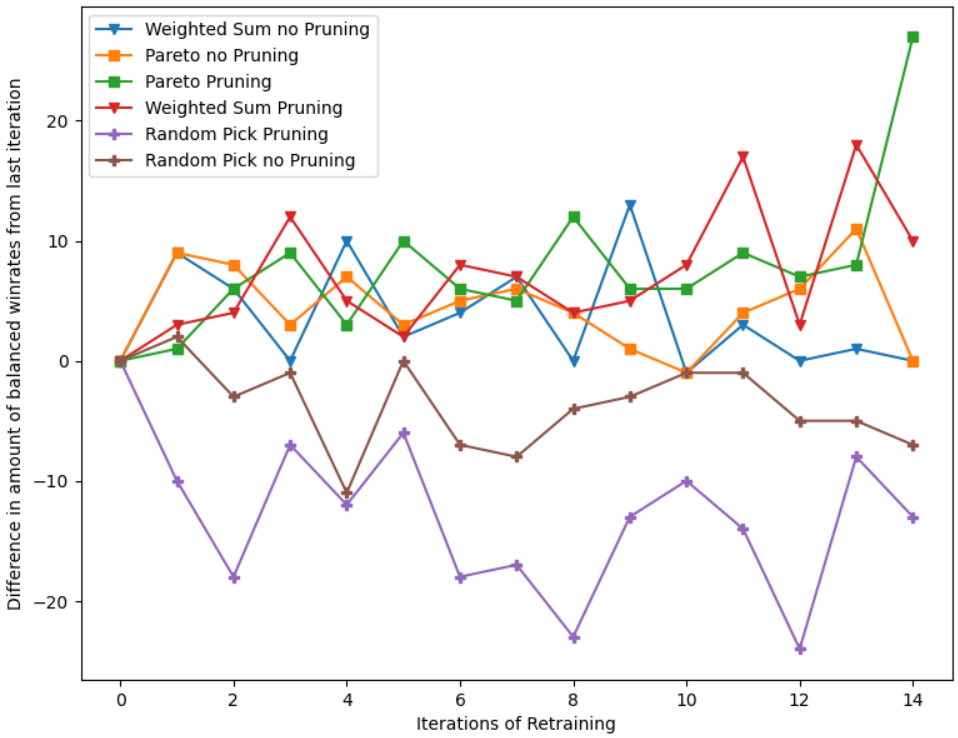

Figure 26: Figure depicting win rates over the course of retraining: Amount of balanced Win rates over time, depicting all the methods, including Pruning and no Pruning and models that used Genetic algorithms and ones that used Random Pick.

The results for balanced win rates are shown in Figure 26 and Table 3. Both this figure and this table report the results for 6 pairs of methods, where the first element of the pair consists of using or not using layer-pruning and the second

Table 3: Numerical representation of Figure 26: win rates over the course of retraining: A numerical representation of Figure 26. The table displays the differences in amount of Pokemon with a balanced win rate over the iterations of retraining compared to the original datasets amount of balanced Pokemon.

Iteration	WS-GA	PD-GA	PD-GA Pruning	WS-GA Pruning	Random Pick Pruning	Random Pick
0	0	0	0	0	0	0
1	9	9	1	3	-10	2
2	6	8	6	4	-18	-3
3	0	3	9	12	-7	-1
4	10	7	3	5	-12	-11
5	2	3	10	2	-6	0
6	4	5	6	8	-18	-7
7	7	6	5	7	-17	-8
8	0	4	12	4	-23	-4
9	13	1	6	5	-13	-3
10	-1	-1	6	8	-10	-1
11	3	4	9	17	-14	-1
12	0	6	7	3	-24	-5
13	1	11	8	18	-8	-5
14	0	0	27	10	-13	-7

element of the pair is one of three types of methods for selecting a subset of Pokemon for creating a retraining dataset (two versions of a GA and one random selection approach) – as discussed in Chapter 3.

In both Figure 26 and Table 3, the results for each pair of methods is shown across time, i.e., across the iterations of retraining of the Transformer, and the result for each iteration consists of the difference: $X_i - X_0$, where X_i is number of Pokemon with balanced win rates in the i-th iteration and X_0 is the number of Pokemon with balanced win rates at the initial time point, before retraining. Recall that a Pokemon is deemed to have a balanced win rate if its win rate (over the battles where it participated) is in the range [45%-55%].

In Figure 26 and Table 3 Random pick with pruning seems to be the most noisy,

this is due to having a random selection of Pokemon to retrain on, as well as being pruned, which further increases the diversity, to its detriment.

In addition, throughout the figures and tables in this Chapter, the acronyms WS-GA and PD-GA stand for Weighted Sum-based GA and Pareto Dominance-based GA, respective (see Chapter 3 for a detailed discussion of these two types of GAs).

Table 4: Statistical significance balance results: p-values obtained by the Wilcoxon sign-rank statistical significance test, when comparing the number of balanced win rates obtained by the pair of methods in a row against the corresponding number obtained by the pair of methods in a column, across the 15 iterations of retraining of the Transformer. The statistically significant p-values are in bold. For statistical significance the significance level (or threshold) is 0.003333. This was obtained using the Bonferroni correction to cope with multiple hypothesis testing, as described in Section 3.6.

	PD-GA	PD-GA Pruning	WS-GA	WS-GA Pruning	Random Pick	Random Pick Pruning
PD-GA	N/A	0.14	0.24	0.12	**0.0014**	**0.0009**
PD-GA Pruning		N/A	0.11	0.89	**0.0011**	**0.0009**
WS-GA			N/A	0.11	**0.0014**	**0.0009**
WS-GA Pruning				N/A	**0.0009**	**0.0009**
Random Pick					N/A	**0.0009**
Random Pick Pruning						N/A

Table 5: Table of win rates: Areas under the curve for Figure 26 to use as another evaluation of the 'best' approach. The best results are in bold.

	Area under the curve Win rate
PD-GA	486.0
PD-GA Pruning	**521.5**
WS-GA	474.0
WS-GA Pruning	**521.0**
Random Pick	369.5
Random Pick Pruning	233.5

4.1.1 Pruning vs non-Pruning Models

Looking at Figure 26 and Table 3, the models that used Pruning and Genetic Algorithms outperformed the models that did not use pruning, but did use genetic algorithms such as Pareto Dominance with layer-pruning outperforming Pareto Dominance without layer-pruning. They did not outperform them significantly, however.

Table 4 shows the statistical significance test results for the comparisons among the pairs of methods. More precisely, each cell in this table shows the p-value obtained by using the Wilcoxon-signed rank test to compare the number of Pokemon with balanced win rates obtained by the pair of methods in a row against the corresponding figure obtained by the pair of methods in a column.

When looking at Table 4, pruning had little to no impact on the balance of generated metagames as can be seen by the lack of significant differences between the MOO approaches.

Table 5 reports the area under the curve in Figure 26 for each pair of methods, where each area was calculated using trapezoids. Figure 26 shows a substantial difference between models that utilise pruning over ones that do not, however it is interesting to view how Random Pick without pruning outperformed Random Pick with Pruning. This is most likely due to pruning having no effect on balance and Random Pick without pruning seemed to have initialised with a more balanced dataset to start with.

4.1.2 Genetic Algorithms vs Random Pick

Looking at Figure 26 and Table 3, the models that utilised genetic algorithms are able to consistently increase the number of balanced Pokemon with the retraining of the Transformer, by comparison with the initial number of balanced Pokemon

before retraining. The contrasts with the two models which utilised Random Pick, which look noisy and were unable to maintain or increase the amount of balanced Pokemon generated.

Table 4 illustrates clearly that using an appropriate multi-objective selection algorithm to increase balance is statistically better than using random selection. Every model that used genetic algorithms to select the Pokemon significantly outperforms the Random Pick models. However, there are no significant differences between the two GA versions (with the Weighted Sum and Pareto Dominance approaches for multi-objective optimisation), as shown in Table 4 when comparing PD-GA vs. WS-GA (p-value = 0.24) and when comparing PD-GA-Pruning vs WS-GA-Pruning (p-value = 0.89). Pruning also seems to have little to no impact on the balance of the generated metagame.

Looking at Table 5, there is a minor difference between the two approaches which both used Pruning and Genetic Algorithms, however there is a substantial difference between using Genetic Algorithms and Random Pick. To conclude, an appropriate selection algorithm for the retraining datasets is paramount to guide the Transformer towards the objective of maximising the number of Pokemon with balanced win rate in the generated metagame.

Looking at the table it becomes clear that the most consequential factor affecting amount of balanced win rates is the use of an appropriate selection algorithm for retraining. In this instance randomly picking the Pokemon gets outperformed by using genetic algorithms, however there is no significant benefit to either of the genetic algorithms based approaches when looking at this table alone.

When looking at these results, the genetic algorithm as a selection algorithm greatly outperforms the models that used Random Pick. This table solidifies the advantage of having an appropriate selection algorithm.

4.2 Diversity Results

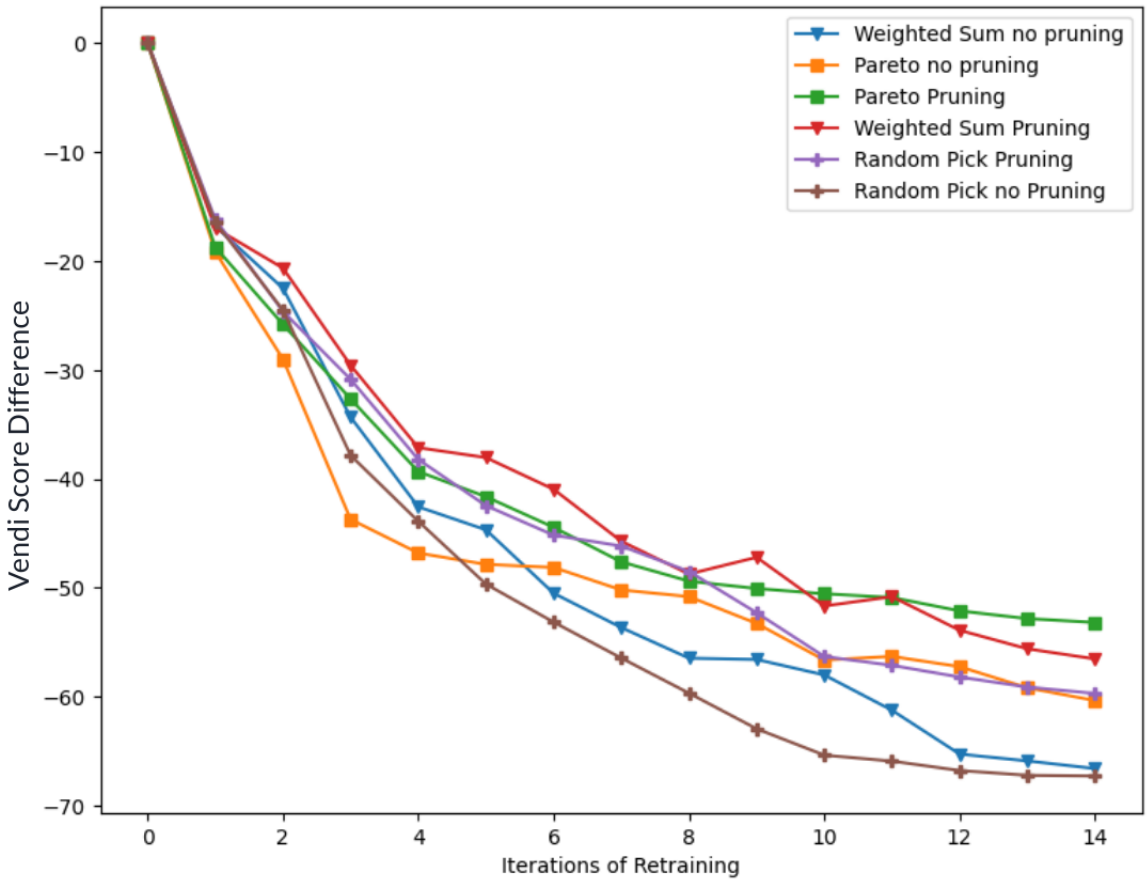

Figure 27: Figure depicting diversity over the course of retraining: The diversity trends over time, measured by the differences in Vendi Scores between the starting point and current iteration, illustrating every method, including models that used Pruning and ones that did not as well as models that used Genetic Algorithms and ones that used Random Pick.

Figure 27 illustrates the diversity trends of the Pokemon in the metagames generated by the Transformer across the iterations of its retraining, for each of the 6 pairs of methods (combining pruning/no pruning with a Pokemon selection method) used in the experiments. Table 6 provides the corresponding numerical values for a clearer quantitative comparison. The diversity in each case is measured using the Vendi Score, a reliable metric for capturing the richness of generated

Table 6: Numerical representation of Figure 27: diversity over the course of retraining.: A numerical representation of Figure 27. Differences in the Vendi Diversity Score between the starting point and the current iteration for each method. The differences were used to display the performance of each model, since each Transformer would initialise differently, thus giving different answers even when trained on the same data. To ensure a lack of bias to models that initialised more strongly, I used the differences. This made sure the performance over time independent of initialisation was evaluated.

Iteration	WS-GA	PD-GA	PD-GA Pruning	WS-GA Pruning	Random Pick Pruning	Random Pick
0	0.0	0.0	0.0	0.0	0.0	0.0
1	-16.81	-19.16	-18.74	-16.99	-16.19	-16.46
2	-22.51	-29.01	-25.82	-20.66	-24.63	-24.51
3	-34.34	-43.72	-32.67	-29.58	-30.87	-37.85
4	-42.57	-46.81	-39.33	-37.14	-38.22	-43.89
5	-44.68	-47.85	-41.64	-38.05	-42.46	-49.66
6	-50.5	-48.13	-44.45	-40.96	-45.17	-53.15
7	-53.65	-50.2	-47.62	-45.73	-46.17	-56.45
8	-56.46	-50.81	-49.39	-48.74	-48.5	-59.67
9	-56.58	-53.27	-50.07	-47.2	-52.29	-62.94
10	-58.0	-56.63	-50.55	-51.67	-56.31	-65.36
11	-61.24	-56.3	-50.89	-50.82	-57.12	-65.91
12	-65.26	-57.24	-52.12	-53.9	-58.19	-66.77
13	-65.89	-59.18	-52.83	-55.6	-59.11	-67.21
14	-66.59	-60.35	-53.18	-56.53	-59.7	-67.27

data. By closely examining both Figure 27 and Table 6, it becomes evident that incorporating pruning into the training process significantly mitigates the loss of diversity over time. This trend is particularly noticeable in the later iterations of training, where the differences between approaches become more pronounced.

Table 7 shows that using pruning has a significant benefit. When comparing models that did use pruning, such as PD-GA Pruning, and ones that did not, such as PD-GA, a significant result can be seen and confirmed when looking at Figure 27 where Pareto Pruning is consistently outperforming Pareto no Pruning. The models that did not use pruning, such as PD-GA have a significant result,

Table 7: Statistical significance test diversity results: p-values obtained by the Wilcoxon sign-rank statistical significance test, when comparing the diversity (Vendi score) obtained by the pair of methods in a row against the corresponding score obtained by the pair of methods in a column, across the 15 iterations of retraining of the Transformer. The statistically significant p-values are in bold.

	PD-GA	PD-GA Pruning	WS-GA	WS-GA Pruning	Random Pick	Random Pick Pruning
PD-GA	N/A	**0.0009**	0.36	**0.0009**	0.018	0.006
PD-GA Pruning		N/A	0.0042	0.10	**0.0018**	0.24
WS-GA			N/A	**0.0012**	**0.0012**	**0.0018**
WS-GA Pruning				N/A	**0.0012**	**0.002**
Random Pick					N/A	**0.0012**
Random Pick Pruning						N/A

only compared to PD-GA pruning and WS-GA pruning, meaning using Pareto Dominance to select Pokemon for the next dataset did not outperform randomly selecting the Pokemon. This is most likely due to randomness being inherently diverse, as presented in Table 7. Interestingly when looking at the Weighted Sum with pruning approach, it significantly outperforms all other models except Pareto Dominance with Pruning, as shown in Table 7 and Figure 27. This contrasts Pareto Dominance Pruning, which was unable to outperform Random Pick Pruning, showing that the Weighted Sum approach was better at maintaining diversity. These results clearly show that using pruning is paramount in maintaining diversity over time, especially long term.

Table 8: Table of diversity: Areas under the curve for Figure 27 to use as another evaluation of the 'best' approach. The best results are in bold.

	Area under the curve Diversity
PD-GA	751.51
PD-GA Pruning	817.29
WS-GA	738.21
WS-GA Pruning	**834.69**
Random Pick	696.53
Random Pick Pruning	794.92

4.2.1 Pruning vs non-Pruning Models

In the final iterations of retraining of the Transformer, the divergence between pruning and non-pruning methods is especially evident in Figure 27 and Table 6. All of the pairs of methods that implemented pruning maintained a higher level of diversity compared to those that did not. This consistent pattern strongly suggests that pruning plays a key role in preserving diversity, which is crucial for the creation and maintenance of a balanced and diverse metagame.

It is worth noting that each model was initialised independently, a necessary consideration given the inherent variability introduced by probabilistic model training. Despite this variability, every pair of methods using layer-pruning obtained superior diversity scores compared to all pairs of methods not using pruning. This result is a compelling indication of the effectiveness and potential advantages of pruning.

Looking at Table 7 it can be seen that Pruning models significantly outperform non-pruning models with two exceptions: Pareto Dominance compared to Random Pick with Pruning, showing that randomly selecting the Pokemon for the next dataset does not maximise diversity; and Pareto Dominance with Pruning compared to Weighted Sum without Pruning, which indicates that Weighted Sum was able to, to an extent, maintain the diversity without pruning.

Table 8 reports the area under the curve in Figure 27 and Table 26 for each pair of methods, where each area was calculated using trapezoids. This allowed me to find the highest performing model in terms of diversity over the span of all of the iterations of retraining. That showed Weighted Sum with Pruning as the best model in terms of preserving diversity over time. Weighted Sum with Pruning and Pareto Dominance with Pruning are by far the best performing models, with Random Pick Pruning being somewhat close. This indicates that pruning

outperforms non-pruning models in terms of diversity using every metric and in most the difference is statistically significant.

Every pruning model outperformed non-pruning models in diversity. This was what I hypothesised. Pareto Dominance approach looks to drop too quickly and level off, maintaining the diversity over time. Weighted Sum approach looks more spiky and unpredictable. Pareto no Pruning approach drops quickly and levels off, at iteration 11 it begins to outperform Random Pick Pruning, but at iteration 13 and after it goes down again. This is quite unexpected as pruning models should be outperforming the non-Pruning models significantly. This shows Random Pick to be detrimental to both balance and diversity.

4.2.2 Genetic Algorithms vs Random Pick

Looking at Figure 27 and Table 6, Random Pick with Pruning performs very closely or even outperforms the non-pruning genetic algorithm approaches, indicating that using genetic algorithms has a much smaller impact on preserving diversity.

Turning to Table 7, Pareto Dominance significantly outperformed Weighted Sum and Random Pick without Pruning, although it did not significantly outperform Random Pick. This further illustrates the lack of impact an appropriate selection algorithm has on diversity of the metagame.

Finally looking at Table 8, Random Pick without Pruning is by far the worst performer which is in line with my hypothesis. The models which used genetic algorithms without pruning are far below pruning models, however they still outperform Random Pick.

To conclude, multi-objective selection algorithms have little to no impact on diversity of the system, but as I will illustrate in the next section, they play a crucial

role in increasing balance.

Regarding computational time, a full run of the genetic algorithm took around 30 minutes. This was reasonable as 15 full runs take 7.5 hours, meaning each model's retraining time was increased by 7.5 hours on average. Without genetic algorithms, 15 full runs took around 40 hours, meaning the added time from the use of GAs was negligible. The hardware used for these experiments is an Nvidia 3090 GPU and an i9 CPU.

4.3 Combining Diversity and Balance Results

In this Section I discuss the trade-off between diversity and balance results associated with using different selection approaches (the two versions of the GA and Random Pick) as well as using or not using layer pruning.

Looking at Figure 27, the diversity decreases over time as the model continues to be retrained, whilst as seen in Figure 26, this is traded for an increased balance over time. However, this trade-off between diversity and balance is less clear in the results involving different selection approaches, particularly different versions of the GA (Weighted Sum and Pareto Dominance). That is because the results are not significantly different and have look to be very close on the graphs in their performance.

When looking at these results; the approach of combining the Weighted Sum-based GA for Pokemon selection with layer-pruning seems to be the most optimal choice as it outperforms every other approach in terms of maintaining diversity.

4.3.1 Pareto Dominance vs Weighted Sum Diversity

Looking to Figure 27 and Table 6, the Weighted Sum approach consistently outperforms the Pareto Dominance approach in terms of diversity (Vendi score) until the last few iterations where it seems to drop off.

As seen in Figure 27, the Weighted Sum approach is much more noisy and spiky than the Pareto Dominance approach, whose diversity score continued to drop until levelling off with minimal spikes. It could be argued here that Pareto Dominance is more consistent as an approach, however it still performed worse when looking at Table 7, where it is shown that PD-GA Pruning did not significantly outperform WS-GA or Random Pick Pruning.

When looking at the non-pruning models for both approaches, the Pareto Dominance approach is close to parity with Random Pick Pruning and is not significantly outperformed by any model that does not use a genetic algorithm-based Pokemon selection method as well as pruning. This is unlike the non-pruning Weighted Sum approach which was significantly outperformed by WS-GA, Random Pick and Random Pick Pruning. As a last comparison, looking at Table 8, WS-GA Pruning is also far above PD-GA Pruning.

To conclude, the Pareto Dominance approach was outperformed by the Weighted Sum approach regarding the diversity (Vendi score) results - this was an unexpected result as the weights used for the Weighted Sum approach were skewed towards balance, making diversity less of a priority. However, it is worth noting that Pareto Dominance has a much more consistent decline in diversity, which could mean that with more iterations of retraining it would outperform the Weighted Sum approach if the trend seen in the last 3 iterations of Transformer retraining continues.

4.3.2 Pareto Dominance vs Weighted Sum Balance

Looking at Figure 26 and Table 3, the same issue of noise in the Weighted Sum with layer-pruning approach is present, whereas Pareto Dominance with Layer-Pruning is quite clean. A noteworthy outlier is the final iteration of the Transformer's retraining using Pareto Dominance and layer-pruning. Due to the probabilistic nature of Transformers as well as the semi-random agents used for battle this outlier is not surprising, however it is noteworthy. When looking at the non-pruning models that utilised genetic algorithms, the trend of the Weighted Sum approach being noisy continues.

Table 4, illustrates that there are no significant differences between the genetic algorithm approaches, in both scenarios of using or not using layer-pruning. When looking at Table 5, this idea is solidified as there is very little difference between the approaches. This should not be the case, as Weighted Sum prioritises balance while Pareto Dominance balanced both of the objectives simultaneously and evenly.

4.3.3 Combining Diversity and Balance Results Conclusion

Weighted Sum is the less consistent approach, with very noisy results that could get worse with more iterations of retraining, as can be seen in Figure 27 where the Weighted Sum lines are more spiky and seem to have more variance between iterations. However, it is clear that, in these experiments and this time frame, Weighted Sum is on par with Pareto Dominance in terms of preserving and increasing balance and greatly outperforms it in terms of maintaining diversity across all pairs of a Pokemon selection methods and a choice of using or not using layer-pruning.

It is worth noting that the Weighted Sum approach requires the user to assign arbitrary weights to the objectives, which can skew the data greatly and introduces an element of chance in that. Pareto Dominance does not struggle with this and instead is a very consistent and more objective approach.

4.4 Discussion Contrasting Good and Bad Metagames

In this section I describe the results in a more qualitative manner, rather than describing numerical results as in the previous section.

My hypothesis was that using multi-objective optimisation to select Pokemon for the next dataset (for retraining the Transformer) would yield better results than randomly picking Pokemon, since the dataset is systematically optimised. As seen in this Chapter, this hypothesis proved correct.

In the next two subsections I will describe two metagames created by the Transformer: one as an example of a good metagame, created with the use of layer pruning and a genetic algorithm for Pokemon selection; and another as an example of a bad metagame, created without using layer pruning and randomly picking Pokemon. Examining the differences between those two examples of metagames will be useful to show how a metagame can be identified as being balanced or unbalanced, as well as giving a more in-depth understanding of the qualities of metagames.

4.4.1 Example of a Good Metagame

An example of a good metagame can be found in Figure 28.

Looking at Figure 28, a large variety of Pokemon have been generated, meaning the model did not converge on specific Pokemon to maximise balance. Additionally many different moves were used which is in contrast with the bad metagame

```
Roselia,Dark,None,Assault Vest,Adamant,252,0,0,0,0,0,Clear Body,Sucker Punch,Flamethrower,Dragon Darts,Phantom Force,
Corphish,Electric,None,Life Orb,Adamant,252,0,0,0,0,0,Clear Body,Superpower,Earthquake,Coaching,Behemoth Bash,
Ferrothorn,Grass,None,Leftovers,Bold,252,0,0,0,0,0,Overcoat,Protect,Iron Defense,Breaking Swipe,Clanging Scales,
Aerodactyl,Flying,None,Focus Sash,Adamant,0,0,252,0,0,0,Aroma Veil,Trick Room,Moonblast,Dazzling Gleam,Heal Pulse,
Gigalith,Rock,None,Weakness Policy,Adamant,252,0,0,0,0,0,Emergency Exit,Poison Jab,Liquidation,Drill Run,X Scissor,
Seadra,Electric,None,Focus Sash,Adamant,252,0,0,0,0,0,Aroma Veil,Trick Room,Moonblast,Dazzling Gleam,Heal Pulse,
Bastiodon,Electric,None,Focus Sash,Adamant,252,0,0,0,0,0,Intimidate,Snarl,Thunder Wave,Thunder Wave,Thunder Wave,
Thievul,Bug,None,Psychic Seed,Adamant,252,0,0,0,0,0,Clear Body,Superpower,Earthquake,Coaching,Behemoth Bash,
Jirachi,Electric,None,Air Balloon,Mild,4,0,0,0,0,0,Intimidate,Earthquake,Hurricane,Dragon Claw,Shadow Claw,
Persian,Electric,None,Assault Vest,Adamant,252,0,0,0,0,0,Regenerator,Rage Powder,Sleep Powder,Sludge Bomb,Leech Seed,
Claydol,Ground,None,Assault Vest,Modest,252,0,0,0,0,0,Intimidate,Earthquake,Hurricane,Dragon Claw,Shadow Claw,
Lumineon,Electric,None,Life Orb,Adamant,0,252,0,0,0,252,Cursed Body,Fake Tears,Will O Wisp,Icy Wind,Ally Switch,
Gloom,Electric,None,Focus Sash,Adamant,252,0,0,0,0,0,Clear Body,Sucker Punch,Flamethrower,Dragon Darts,Phantom Force,
Combee,Electric,None,Air Balloon,Mild,4,0,0,0,0,0,Lightning Rod,Snarl,Thunder,Thunder Wave,Thunder Wave,
Drampa,Electric,None,Assault Vest,Modest,252,0,0,0,0,0,Intimidate,Earthquake,Hurricane,Dragon Claw,Shadow Claw,
Cherrim,Grass,None,Focus Sash,Adamant,252,0,0,0,0,0,Cursed Body,Fake Tears,Will O Wisp,Icy Wind,Hurricane,
Dracozolt,Dragon,Fire,Focus Sash,Modest,4,0,0,0,0,0,Teravolt,Protect,Iron Defense,Breaking Swipe,Clanging Scales,
Hitmonlee,Fighting,None,Psychic Seed,Adamant,252,0,0,0,0,0,Clear Body,Sucker Punch,Flamethrower,Dragon Darts,Phantom Force,
Medicham,Electric,None,Focus Sash,Modest,252,0,0,0,0,0,Intimidate,Fake Out,Iron Head,Drill Run,X Scissor,
Roserade,Steel,Grass,Focus Sash,Modest,0,0,0,252,0,0,Download,Trick Room,Eerie Impulse,Shadow Ball,Recover,
Electivire,Electric,None,Assault Vest,Adamant,252,0,0,0,0,0,Clear Body,Superpower,Earthquake,Coaching,Behemoth Bash,
Swellow,Electric,None,Focus Sash,Adamant,252,0,0,0,0,0,Clear Body,Sucker Punch,Flamethrower,Dragon Darts,Phantom Force,
Glaceon,Ice,None,Focus Sash,Modest,252,0,0,0,0,0,Aroma Veil,Trick Room,Moonblast,Dazzling Gleam,Heal Pulse,
Weezing,Poison,None,Focus Sash,Adamant,252,0,0,0,0,0,Aroma Veil,Trick Room,Moonblast,Dazzling Gleam,Heal Pulse,
Tepig,Dragon,Fire,Focus Sash,Modest,4,0,0,0,0,0,Beast Boost,Rising Voltage,Hyper Beam,Energy Ball,Dazzling Gleam,
Pichu,Electric,None,Eviolite,Hardy,252,0,0,0,0,0,Regenerator,Rage Powder,Sleep Powder,Sludge Bomb,Leech Seed,
Sealeo,Electric,None,Focus Sash,Adamant,252,0,0,0,0,0,Overcoat,Protect,Iron Defense,Breaking Swipe,Clanging Scales,
Haxorus,Dragon,None,Wacan Berry,Adamant,252,0,0,0,0,0,Clear Body,Sucker Punch,Flamethrower,Dragon Darts,Phantom Force,
Charmander,Electric,None,Focus Sash,Adamant,252,0,0,0,0,0,Clear Body,Sucker Punch,Flamethrower,Dragon Darts,Phantom Force,
Scizor,Bug,None,Choice Band,Adamant,252,0,0,0,0,0,Clear Body,Superpower,Earthquake,Coaching,Behemoth Bash,
Misdreavus,Electric,None,Life Orb,Adamant,252,0,0,0,0,252,Inner Focus,Icy Wind,Surf,Sunny Day,Psychic,
Kecleon,Normal,None,Life Orb,Adamant,252,0,0,0,0,0,Intimidate,Earthquake,Hurricane,Dragon Claw,Shadow Claw,
Typhlosion,Water,None,Life Orb,Adamant,252,0,0,0,0,0,Clear Body,Superpower,Earthquake,Coaching,Behemoth Bash,
Emboar,Ground,None,Leftovers,Bold,252,0,0,0,0,0,Inner Focus,Icy Wind,Surf,Sunny Day,Triple Axel,
Litwick,Normal,None,Eviolite,Hardy,252,0,0,0,0,0,Regenerator,Rage Powder,Sleep Powder,Sludge Bomb,Leech Seed,
```

Figure 28: Example of a good metagame: A part of the last metagame created using Pareto Dominance with Pruning. It depicts a rather diverse metagame where many different moves are used, as well as many different EVs are being maximised. It also shows many abilities, items and natures being used.

example seen in Figure 30 where the model converged on using very few different moves. The metagame in Figure 28 contains the largest amount of balanced Pokemon in it, standing at 84 out of 200. It also achieved a 25.52 Vendi Score which was the highest diversity for any model's final iteration of retraining. The metagame, although diverse, has some outliers that are favoured over other choices. For example, the network converged on mostly using 252 and 0's for the EVs, this is most likely due to competitive Pokemon found in the initial set doing so as well.

Another example of this can be seen when looking at the number of fairy and poison types used, both standing at 1. This is most likely due to fairy types being very strong in Pokemon, having only 2 weaknesses (steel and poison), meaning the system must not have been retrained on many fairy types as they would exceed the win rate threshold. This also explains why poison types are under-represented - poison types are only strong against grass and fairy types. This could mean fairy types need balancing - adding a new weakness such as fire type weakness could decrease the power of the type and make it more balanced.

The most represented item in this metagame is 'focus sash' with 78 Pokemon using it. The nature of using agents that prioritise high-damage moves means surviving high damage hits is important. The item 'focus sash' allows the Pokemon to survive being knocked out in one attack from maximum health.

These findings show the system to be evolving and adapting to the environment created. Some Pokemon have picked the highest damaging moves, like Earthquake and Hyper Beam, and this is highly beneficial. On the other side of this you have Pokemon with focus sash and moves like 'Trick Room', which inverts the speed of Pokemon on the field, giving slower Pokemon a chance at winning. Moves like 'Trick Room' do not deal damage, meaning you need to pair it with 'focus sash' to survive at least one turn to set 'Trick Room' up.

This emergent behaviour could prove very interesting to study in other games, to see how this autonomous system converges on solutions to problems the constraints of the metagame produce.
Figure 29 shows a distribution of tokens used by the model to present diversity in a more visual way.

From this Figure I removed Pokemon names due to the large number of different names making the graph much harder to interpret. I also removed 0's as the

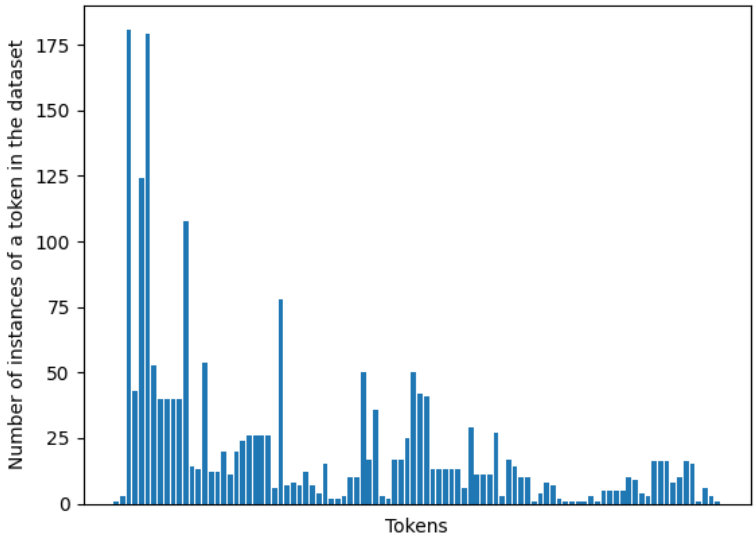

Figure 29: Token distribution of the good metagame: A graphical representation of the distribution of Tokens in the Balanced Dataset 28 without Pokemon names or 0's in EV's. A visual method to evaluate diversity of a system. This Figure depicts a wide variety of different tokens being used, which suggests the metagame that uses this dataset is diverse.

average Pokemon will have five 0's in their EV's and it dampens every number making the graph harder to interpret.

Looking at Figure 29, two centralising tokens can be seen, those being: 'None' with 181 instances, meaning the system prioritises mono-typal Pokemon, most likely because dual-type Pokemon are much harder to balance as they have more weaknesses and more resistances, meaning more dual type Pokemon on the high and low end of win rates in the metagame. The second token is '252' with 179 instances, which is the highest possible an EV can go up to, most Pokemon have a maximises EV and some have two maximised as seen in line 31 of Figure 28.

Looking at Figure 28 a lot of Pokemon can be seen to have a maximised health EV, most likely to combat the agents picking highest damaging moves and attempting to survive these high damage attacks. The final outlier to explore is 'Adamant' with 124 instances, which is the nature that increases attack while decreasing

special attack. The high instances of this token is most likely due to the highest damaging moves in the game being physical moves rather than special ones.

Looking at the whole Figure, it is clear that many different tokens are represented, which reinforces the idea that pruning layers of the Transformer increases diversity. Some of the Pokemon generated are also very interesting. For example the last Pokemon in line 35 of Figure 28, Litwick, uses Eviolite which is an item that boosts the Pokemon's Defences and Special Defences, but only if the Pokemon is unevolved. The rest of this set contains:

- Regenerator, an ability that heals the Pokemon when it switches out; Rage Powder, which taunts the opposing Pokemon to attack Litwick.
- Leech Seed, which slowly drains the opposing Pokemons health while healing Litwick.
- Sleep Powder, which applies a sleep status condition and prevents the opposing Pokemon from moving.
- Sludge Bomb, which is the only damaging move and has a slight chance of poisoning the opponent.

All of these put together mimic a very strong Pokemon, Amoonguss, but while used with Litwick and Eviolite, it actually could outperform the frequently used Amoonguss.

This shows the system is able to stitch together Pokemon in interesting ways, creating new and diverse Pokemon when both layer pruning and GA-based Pokemon selection are used.

```
Sandshrew,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Gligar,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Nidorino,Water,None,Assault Vest,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Whismur,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Cosmog,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Shiftry,Grass,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Unfezant,Normal,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Slaking,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Riolu,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Toxicroak,Normal,None,Assault Vest,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Beldum,Normal,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Eevee,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Shinx,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Durant,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Yanma,Normal,None,Assault Vest,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Carvanha,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Poochyena,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Rowlet,Water,None,Weakness Policy,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Gastrodon,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Corviknight,Normal,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Teddiursa,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Grookey,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Qwilfish,Normal,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Sudowoodo,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Tyrogue,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Misty Surge,Moonblast,Hydro Pump,Muddy Water,Surf,
Poliwhirl,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Bouffalant,Normal,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Pheromosa,Water,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Munna,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Fearow,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Seedot,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
Fearow,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Thievul,Water,None,Wacan Berry,Quiet,252,0,0,0,0,0,Liquid Voice,Hyper Beam,Moonblast,Perish Song,Flip Turn,
Chandelure,Fire,None,Life Orb,Serious,252,0,0,0,0,0,Lightning Rod,Protect,Drill Run,Megahorn,Horn Drill,
Geodude,Water,None,Life Orb,Serious,252,0,0,0,0,0,Drizzle,Protect,Water Spout,Yawn,Ice Beam,
```

Figure 30: Example of a bad metagame: A snippet of the last metagame created using no Genetic Algorithms and no Pruning. This is an example of a very homogeneous metagame. Every Pokemon is mono typal and every Pokemon has one maximised EV which is health. There is a lack of variety in the moves used. Looking at the abilities, there are only three used frequently and one that is used once. This is a very unbalanced and uniform metagame.

4.4.2 Example of a Bad Metagame

An example of an unbalanced metagame can be found in Figure 30. Many obvious issues can be seen here, including the use of only 252 and 0 for the EV's. There are no variations of any EV's unlike with the previous dataset presented, in which certain Pokemon varied their EVs.

In the metagame presented in Figure 30, each Pokemon maximised their health EV and nothing else. Every Pokemon converged to be mono-typal and 159 of them are water types. This is most likely due to water having very few weaknesses [1]. Additionally, lightning rod is the most represented ability, which allows water types to remove one of their worst match-ups - electric type attacks. With no fighting types, normal type Pokemon are the second most represented due to having a same-type-attack-boost (when a Pokemon uses a move that shares a type with the Pokemon, the damage of the move is boosted by 150%) with Hyper Beam, the highest damaging move in the game, due to the move being a normal type.

The biggest issue which is consistent throughout the dataset is that the majority of the Pokemon are using a combination of the same moves - moves that are either very strong, such as Hyper Beam or Megahorn, or are defensive measures against these strong moves, such as protect. The metagame is centralised on very few tactics with minimal variance, and I believe that with more iterations of retraining a rock, paper, scissors dynamic would emerge, where you have 3 archetypes of Pokemon that all beat one another. This is a common problem many unbalanced games face - an aggressive archetype, such as the Pheromosa on line 28, will lose to a control archetype, such as the Sandshrew seen on line 1, which will lose to a value-based archetype, such as Litwick as seen in Figure 28 on line 35. If you only have one strong representative of each archetype, you create an environment in which only those 3 entities will be used, as it would be detrimental to use a suboptimal version of an archetype when a better version exists and offers a much higher chance of winning.

Figure 30 clearly illustrates why balance is so important, the fun and diversity of the game stems from balance, seeing the same entities in every game gets tiring

[1] https://pokemondb.net/type Last accessed 11th September 2025

and boring very quickly which means balance is paramount.

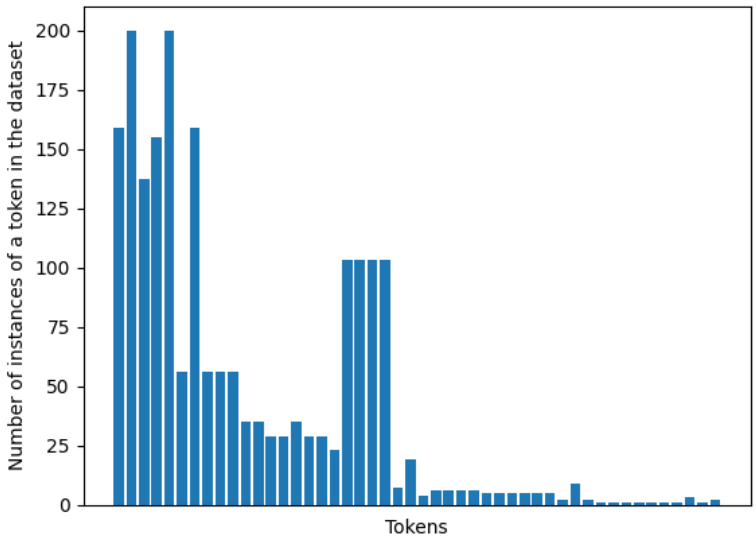

Figure 31: Token distribution of the bad metagame: A graphical representation of the distribution of Tokens in the Unbalanced Dataset 30 without Pokemon names or 0. This graph depicts a very small diversity in tokens showing that the metagame which this graph represents is homogeneous.

Figure 31 shows a distribution of tokens used by the model to present diversity in a more visual way.

For this Figure I also removed Pokemon names and 0's for the same reason as Figure 29. Much like the good metagame example, the two most represented tokens are 'None' with 200 instances and '252' with 200 instances. This tells us that every single Pokemon in the dataset is a mono-typal Pokemon with a single maximised EV, which is health as seen in Figure 30.

The last outlier to explore here is 'Serious' with 155 instances. Unlike in the good metagame example, this nature does not change any EVs and is used very infrequently in any competitive environment, because it gives no benefits. The system most likely converged on it because it is easier to balance around a nature that does not have an effect.

'Protect', a move that cancels damage to the Pokemon rather than attacking, is the most represented move in this metagame, and the most represented type is water as mentioned above. Water types are known as the most defensive-leaning types in the game. This suggests that, to balance the game, the system has promoted a defensive play style even when the agents would rarely pick 'Protect' due to their focus on damaging moves.

4.5 Summary of Results, Contributions and Limitations

4.5.1 Summary of Results and Contributions

The two objectives of this thesis were to perform layer-pruning in a Transformer model and to develop a multi-objective genetic algorithm for Pokemon selection, in order to achieve the overall aim of improving the win-rate balance and the diversity of a set of Pokemon in a metagame. The results reported in this chapter show that overall these two types of proposed methods have successfully improved the balance and diversity of Pokemon metagames. This can be noted e.g. by analysing the results showing an increasing balance in Figure 26 and a maintenance of diversity in Figure 27, over time (i.e. over the iterations of retraining of the Transformer). In addition, the two types of proposed methods have significantly outperformed the approaches of not performing layer pruning and of randomly picking Pokemon, as shown by the results in Tables 4 and 7.

There is relatively little difference of performance between the two versions of the genetic algorithm for Pokemon selection, namely the versions implementing the

Weighted Sum and the Pareto Dominance approach for multi-objective optimisation (MOO).

The results also show that pruning inconsequential layers of the Transformer during training may not only be useful in decreasing computational time, but it can also be used to combat the decrease in diversity of a Transformer.

The initialisation of the Transformer, the use of layer pruning, and the use of a MOO-based genetic algorithm for Pokemon selection greatly impacts the metagame. The emergent behaviour displayed in both Figure 28 and Figure 30 shows autonomous systems can create very interesting and diverse metagames without human interference. These figures also illustrate that diversity can be easily interpreted even when looking at small sections of the whole metagame, much like looking at a few recent tournaments to determine the health of a particular game. It may not show the entire story, but will outline problems.

By guiding the Transformer towards generating Pokemon with 50% win rates, the metagame exhibited many different playstyles, ways of using different Pokemon, on different teams, for the goal of winning. An example of very different playstyles generated can be seen in Figure 28 with Drampa that uses intimidate and four attacking moves alongside assault vest, this combination creates a very powerful attacker that also has a way of preserving itself on the battlefield with the choice of an attack reducing ability and defence increasing item. In contrast to this the outlined Litwick is the perfect stall Pokemon, very defensive with a healing ability and a defensive item alongside 3 non-damaging moves. By having a larger pool of Pokemon to choose from, players are able to win using a strategy they are most drawn to, rather than having to pick from a select few strategies considered good at a given point in time, giving the player ability to express their skill and themselves as a player.

4.5.2 Limitations of the proposed approach

In this thesis I manually chose solutions from the Pareto front produced by the genetic algorithm based primarily on balance although if two solutions were close in that metric, I chose one with higher diversity. This approach is less than optimal. Ideally, I would have calculated the hypervolume of the graph below the Pareto front, when an optimal solution is removed, the hypervolume changes. Finding the solution that maximises the hypervolume finds the most optimal solution. This is a more systematic method of finding the optimal solution, however it was not possible in this project due to time constraints and limited scope, but this can be explored in future research.

Chapter 5

Conclusion and Future Work

This thesis offers two main contributions, involving two proposed methods for improving the Pokemon metagame generated by a Transformer. The first contribution is the approach of pruning layers of a Transformer model, whilst the second contribution is the development of a genetic algorithm that was implemented in two different versions, with two different multi-objective optimisation approaches – namely the Weighted Sum and the Pareto Dominance approaches (described in detail in Chapter 3). In general, the results of experiments with a Pokemon dataset have shown that the proposed methods effectively achieved the overall aim of this research (specified in Section 1.2), namely to increase the balance and diversity of the set of Pokemon in a metagame generated by a Transformer.

5.1 Sumary of Contributions

Based on the experiments performed in this research, the overall best approach for retraining the Transformer is to use a Genetic Algorithm (GA) with Weighted Sum fitness function combined with pruning (WS-GA Pruning), prioritising balance over diversity in the GA's fitness function, which achieved overall the best results in terms of maximising diversity and increasing the number of balanced

win rates over time. The GA with Pareto Dominance combined with pruning (PD-GA Pruning) had a performance close to the performance of WS-GA Pruning, and it could have been closer if the best solution from the Pareto front was chosen systematically by using the hypervolume measure.

More broadly, the experimental results have shown that layer-pruning is paramount in preserving diversity over time – every Transformer model produced without using layer-pruning was significantly outperformed by models generated using pruning. This is shown in Tables 7 and 8, where the noteworthy and significant results stem from the comparison between Transformer models produced by method combinations like WS-GA Pruning and models produced by method combinations that did not use pruning. The only model that outperformed WS-GA Pruning regarding diversity was PD-GA Pruning, which used both layer pruning and an appropriate multi-objective optimisation GA for Pokemon selection. In addition, looking at the results of models that used pruning and GA-based Pokemon selection in Figure 26, it can be noted that pruning is able to help maintain diversity. This is also shown in Figure 27, as both PD-GA Pruning and WS-GA Pruning greatly outperform other models.

The experimental results also have shown that using an appropriate selection algorithm, one that uses genetic algorithms for multi-objective-optimisation, is paramount in increasing the balance of metagames generated. This is shown in Tables 4 and 5, where the significant results stem from the comparison of Transformers models produced by combining layer-pruning with genetic algorithms such as WS-GA and PD-GA Pruning.

However there seems to not be a significant difference in which MOO approach you use as both Weighted Sum and Pareto Dominance seem to perform closely to each other with no significant difference between the two.

5.2 Future Work

There are several research directions for future work, as follows. First, it would be useful to perform experiments with a larger dataset, in order to investigate if this would improve the quality of a metagame as measured in terms of diversity and balance of Pokemon.

Second, as mentioned earlier, one limitation of the current experiments with the PD-GA is that I have manually chosen the best solution among all non-dominated solutions returned by the algorithm. Hence, a natural research direction would be to replace that manual selection by a more systematic and automated solution-selection criterion, based e.g. on the concept of hypervolume (as discussed in Figure 23).

Third, the development of another type of multi-objective optimisation (MOO) GA could be investigated. More precisely, this thesis focused on two types of MOO GAs, based on the Weighted Sum and the Pareto Dominance approaches, which are the two most popular approaches for MOO GAs; but it would be interesting to develop a MOO GA based on lexicographic optimisation for Pokemon selection. The basic principle of lexicographic optimisation is to optimise objectives in decreasing order of priority, which, in the context of this thesis, could be used to specify a type of fitness function where balance has higher priority than diversity, without requiring that users specify ad-hoc, arbitrary values for these objectives' weights (as in the Weighted Sum approach).

Beyond metagame generation, these findings can be applied to other areas, such as story-generating Transformers designed to maintain creativity while being guided towards a specific concept. For example, adaptive story telling in games could use this system as a way of adapting the game with the players' choices. This system

can also be used for any games where users want to include procedurally generated entities. They can take player feedback and statistics to adapt the generative system to keep the game fresh and exciting while also maintaining balance. This could result in a unique experience for every player, by adapting the game to their choices during the game.

Bibliography

Aha, David W, Héctor Muñoz-Avila, and Michael van Lent (2005). "Proceedings of the 2005 IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games". en. In.

Andrade, Gustavo, Geber Ramalho, Alex Gomes, and Vincent Corruble (2006). "Dynamic Game Balancing: an Evaluation of User Satisfaction". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 2.1, pp. 3–8. ISSN: 2334-0924. DOI: 10.1609/aiide.v2i1.18739. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/18739 (visited on 11/03/2024).

Babin, Mathias and Michael Katchabaw (May 2025). "Game Balance Through Procedural Content Generation". In: *Proceedings of the 20th International Conference on the Foundations of Digital Games*. FDG '25. New York, NY, USA: Association for Computing Machinery, pp. 1–4. ISBN: 979-8-4007-1856-4. DOI: 10.1145/3723498.3723748. URL: https://dl.acm.org/doi/10.1145/3723498.3723748 (visited on 05/05/2026).

Becker, Alexander and Daniel Görlich (Apr. 2020). "What is Game Balancing? - An Examination of Concepts". en. In: *ParadigmPlus* 1.1, pp. 22–41. ISSN: 2711-4627. DOI: 10.55969/paradigmplus.v1n1a2. URL: https://journals.itiud.org/index.php/paradigmplus/article/view/7 (visited on 07/13/2025).

Bengio, Y., P. Simard, and P. Frasconi (Mar. 1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural*

Networks 5.2, pp. 157–166. ISSN: 1941-0093. DOI: 10.1109/72.279181. URL: https://ieeexplore.ieee.org/document/279181/ (visited on 07/31/2025).

Cox, D R (Sept. 1982). "Statistical significance tests." In: *British Journal of Clinical Pharmacology* 14.3, pp. 325–331. ISSN: 0306-5251. DOI: 10.1111/j.1365-2125.1982.tb01987.x. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1427620/ (visited on 08/18/2025).

David Stephens, Conor (June 2024). "Balancing multiplayer games design using deep learning techniques". en. thesis. University of Limerick. DOI: 10.34961/researchrepository-ul.26509015.v1. URL: https://researchrepository.ul.ie/articles/thesis/Balancing_multiplayer_games_design_using_deep_learning_techniques/26509015/1 (visited on 05/21/2025).

DeLaurentis, Daniel A., Jitesh H. Panchal, Ali K. Raz, Prajwal Balasubramani, Apoorv Maheshwari, Adam Dachowicz, and Kshitij Mall (Aug. 2021). "Toward Automated Game Balance: A Systematic Engineering Design Approach". In: *2021 IEEE Conference on Games (CoG)*, pp. 1–8. DOI: 10.1109/CoG52621.2021.9619032. URL: https://ieeexplore.ieee.org/document/9619032/ (visited on 08/11/2025).

Fonseca, Carlos M. and Peter J. Fleming (Mar. 1995). "An Overview of Evolutionary Algorithms in Multiobjective Optimization". en. In: *Evolutionary Computation* 3.1, pp. 1–16. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/evco.1995.3.1.1. URL: https://direct.mit.edu/evco/article/3/1/1-16/733 (visited on 08/18/2025).

Freiknecht, Jonas and Wolfgang Effelsberg (Sept. 2020). "Procedural Generation of Interactive Stories using Language Models". In: *Proceedings of the 15th International Conference on the Foundations of Digital Games*. FDG '20. New York, NY, USA: Association for Computing Machinery, pp. 1–8. ISBN: 978-1-4503-8807-8. DOI: 10.1145/3402942.3409599. URL: https://dl.acm.org/doi/10.1145/3402942.3409599 (visited on 09/24/2024).

Friedman, Dan and Adji Bousso Dieng (July 2023). *The Vendi Score: A Diversity Evaluation Metric for Machine Learning*. URL: http://arxiv.org/abs/2210.02410 (visited on 11/19/2024).

Gow, Jeremy and Joseph Corneli (2015). "Towards Generating Novel Games Using Conceptual Blending". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 11.3, pp. 15–21. ISSN: 2334-0924. DOI: 10.1609/aiide.v11i3.12824. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/12824 (visited on 09/19/2025).

Han, Ruizi and Jinglei Tang (2025). "Straightforward Layer-Wise Pruning for More Efficient Visual Adaptation". en. In: *Computer Vision – ECCV 2024*. Ed. by Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol. Cham: Springer Nature Switzerland, pp. 236–252. ISBN: 978-3-031-73220-1. DOI: 10.1007/978-3-031-73220-1_14.

He, Shwai, Guoheng Sun, Zheyu Shen, and Ang Li (Oct. 2024). *What Matters in Transformers? Not All Attention is Needed*. URL: http://arxiv.org/abs/2406.15786 (visited on 10/28/2024).

Hendrikx, Mark, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup (Feb. 2013). "Procedural content generation for games: A survey". en. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 9.1, pp. 1–22. ISSN: 1551-6857, 1551-6865. DOI: 10.1145/2422956.2422957. URL: https://dl.acm.org/doi/10.1145/2422956.2422957 (visited on 07/31/2025).

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov (July 2012). *Improving neural networks by preventing co-adaptation of feature detectors*. DOI: 10.48550/arXiv.1207.0580. URL: http://arxiv.org/abs/1207.0580 (visited on 08/18/2025).

Huang, Hanjuan, Hao-Jia Song, and Hsing-Kuo Pao (May 2024). *Large Language Model Pruning*. DOI: 10.48550/arXiv.2406.00030. URL: http://arxiv.org/abs/2406.00030 (visited on 12/10/2024).

Japkowicz, Nathalie and Mohak Shah (2011). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge: Cambridge University Press. ISBN: 978-0-521-19600-0. DOI: 10.1017/CBO9780511921803. URL: https://www.cambridge.org/core/books/evaluating-learning-algorithms/3CB22D16AB609D1770C24C (visited on 09/19/2025).

Karita, Shigeki, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang (Dec. 2019). "A Comparative Study on Transformer vs RNN in Speech Applications". In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 449–456. DOI: 10.1109/ASRU46091.2019.9003750. URL: https://ieeexplore.ieee.org/abstract/document/9003750 (visited on 07/09/2025).

Kim, Sehoon, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer (Aug. 2022). "Learned Token Pruning for Transformers". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Washington DC USA: ACM, pp. 784–794. DOI: 10.1145/3534678.3539260. URL: https://dl.acm.org/doi/10.1145/3534678.3539260 (visited on 05/01/2025).

Kwon, Woosuk, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami (Nov. 2022). "A fast post-training pruning framework for transformers". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., pp. 24101–24116. ISBN: 978-1-7138-7108-8. (Visited on 05/21/2026).

Lakew, Surafel M., Mauro Cettolo, and Marcello Federico (June 2018). *A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation*. DOI: 10.48550/arXiv.1806.06957. URL: http://arxiv.org/abs/1806.06957 (visited on 07/09/2025).

Lara-Cabrera, Raúl, Carlos Cotta, and Antonio J. Fernández-Leiva (June 2014). "On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms". en. In: *Natural Computing* 13.2, pp. 157–168. ISSN: 1572-9796. DOI: 10.1007/s11047-014-9418-9. URL: https://doi.org/10.1007/s11047-014-9418-9 (visited on 08/11/2025).

Lu, Yao, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu (Nov. 2024). *Reassessing Layer Pruning in LLMs: New Insights and Methods*. DOI: 10.48550/arXiv.2411.15558. URL: http://arxiv.org/abs/2411.15558 (visited on 12/10/2024).

Mahlmann, Tobias, Julian Togelius, and Georgios N. Yannakakis (June 2012). "Evolving card sets towards balancing dominion". In: *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8. DOI: 10.1109/CEC.2012.6256441. URL: https://ieeexplore.ieee.org/document/6256441/ (visited on 09/09/2025).

Mason, Stacey, Ceri Stagg, and Noah Wardrip-Fruin (Aug. 2019). "Lume: a system for procedural story generation". In: *Proceedings of the 14th International Conference on the Foundations of Digital Games*. FDG '19. New York, NY, USA: Association for Computing Machinery, pp. 1–9. ISBN: 978-1-4503-7217-6. DOI: 10.1145/3337722.3337759. URL: https://dl.acm.org/doi/10.1145/3337722.3337759 (visited on 05/21/2026).

McCandlish, Sam, Jared Kaplan, Dario Amodei, and OpenAI Dota Team (Dec. 2018). *An Empirical Model of Large-Batch Training*. DOI: 10.48550/arXiv.1812.06162. URL: http://arxiv.org/abs/1812.06162 (visited on 08/18/2025).

Mohaghegh, Sajad, Mohammad Amin Ramezan Dehnavi, Golnoosh Abdollahinejad, and Matin Hashemi (Oct. 2023). *PCGPT: Procedural Content Generation via Transformers*. DOI: 10.48550/arXiv.2310.02405. URL: http://arxiv.org/abs/2310.02405 (visited on 09/24/2024).

Nafday, Avinash M. (Oct. 2009). "Strategies for Managing the Consequences of Black Swan Events". EN. In: *Leadership and Management in Engineering* 9.4, pp. 191–197. ISSN: 1532-6748. DOI: 10.1061/(ASCE)LM.1943-5630.0000036. URL: https://ascelibrary.org/doi/10.1061/%28ASCE%29LM.1943-5630.0000036 (visited on 08/12/2025).

Osborn, Joseph C., Melanie Dickinson, Barrett Anderson, Adam Summerville, Jill Denner, David Torres, Noah Wardrip-Fruin, and Michael Mateas (Oct. 2019). "Is Your Game Generator Working? Evaluating Gemini, an Intentional Generator". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 15.1, pp. 59–65. ISSN: 2334-0924. DOI: 10.1609/aiide.v15i1.5225. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/5225 (visited on 07/31/2025).

Peebles, William and Saining Xie (Mar. 2023). *Scalable Diffusion Models with Transformers*. DOI: 10.48550/arXiv.2212.09748. URL: http://arxiv.org/abs/2212.09748 (visited on 08/08/2025).

Peeperkorn, Max, Tom Kouwenhoven, Dan Brown, and Anna Jordanous (May 2024). *Is Temperature the Creativity Parameter of Large Language Models?* DOI: 10.48550/arXiv.2405.00492. URL: http://arxiv.org/abs/2405.00492 (visited on 11/25/2024).

Peer, David, Sebastian Stabinger, Stefan Engl, and Antonio Rodríguez-Sánchez (May 2022). "Greedy-layer pruning: Speeding up transformer models for natural language processing". In: *Pattern Recognition Letters* 157, pp. 76–82. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2022.03.023. URL: https://www.

sciencedirect.com/science/article/pii/S0167865522000885 (visited on 05/01/2025).

Perez-Nieves, Nicolas, Yaodong Yang, Oliver Slumbers, David H. Mguni, Ying Wen, and Jun Wang (July 2021). "Modelling Behavioural Diversity for Learning in Open-Ended Games". en. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, pp. 8514–8524. URL: https://proceedings.mlr.press/v139/perez-nieves21a.html (visited on 08/18/2025).

Pfau, Johannes and Magy Seif El-Nasr (Aug. 2024). "On Video Game Balancing: Joining Player- and Data-Driven Analytics". In: *ACM Games* 2.3, 27:1–27:30. DOI: 10.1145/3675807. URL: https://doi.org/10.1145/3675807 (visited on 05/28/2025).

Pleines, Marco, Daniel Addis, David Rubinstein, Frank Zimmer, Mike Preuss, and Peter Whidden (Mar. 2025). *Pokemon Red via Reinforcement Learning*. DOI: 10.48550/arXiv.2502.19920. URL: http://arxiv.org/abs/2502.19920 (visited on 05/21/2025).

Politowski, Cristiano, Fabio Petrillo, Ghizlane ElBoussaidi, Gabriel C. Ullmann, and Yann-Gaël Guéhéneuc (Apr. 2023). *Assessing Video Game Balance using Autonomous Agents*. DOI: 10.48550/arXiv.2304.08699. URL: http://arxiv.org/abs/2304.08699 (visited on 08/11/2025).

Pugh, Justin K., Lisa B. Soros, and Kenneth O. Stanley (July 2016). "Quality Diversity: A New Frontier for Evolutionary Computation". English. In: *Frontiers in Robotics and AI* 3. ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00040. URL: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2016.00040/full (visited on 07/31/2025).

Reis, Simão, Rita Novais, Luís Paulo Reis, and Nuno Lau (June 2023). "An Adversarial Approach for Automated Pokémon Team Building and Metagame

Balance". In: *IEEE Transactions on Games* 16.2, pp. 365–375. ISSN: 2475-1510. DOI: 10.1109/TG.2023.3273157. URL: https://ieeexplore.ieee.org/document/10115492/ (visited on 05/28/2025).

Reis, Simão, Luís Paulo Reis, and Nuno Lau (Aug. 2021). "VGC AI Competition - A New Model of Meta-Game Balance AI Competition". In: *2021 IEEE Conference on Games (CoG)*, pp. 01–08. DOI: 10.1109/CoG52621.2021.9618985. URL: https://ieeexplore.ieee.org/document/9618985/ (visited on 05/21/2025).

Rupp, Florian, Alessandro Puddu, Christian Becker-Asano, and Kai Eckert (Aug. 2024). "It might be balanced, but is it actually good? An Empirical Evaluation of Game Level Balancing". In: *2024 IEEE Conference on Games (CoG)*, pp. 1–4. DOI: 10.1109/CoG60054.2024.10645642. URL: https://ieeexplore.ieee.org/document/10645642/ (visited on 08/11/2025).

Sastry, Kumara, David E. Goldberg, and Graham Kendall (2014). "Genetic Algorithms". en. In: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Ed. by Edmund K. Burke and Graham Kendall. Boston, MA: Springer US, pp. 93–117. ISBN: 978-1-4614-6940-7. DOI: 10.1007/978-1-4614-6940-7_4. URL: https://doi.org/10.1007/978-1-4614-6940-7_4 (visited on 08/22/2025).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162/ (visited on 07/20/2025).

Shim, Kyuhong, Iksoo Choi, Wonyong Sung, and Jungwook Choi (Oct. 2021). "Layer-wise Pruning of Transformer Attention Heads for Efficient Language

Modeling". In: *2021 18th International SoC Design Conference (ISOCC)*, pp. 357–358. DOI: 10.1109/ISOCC53507.2021.9613933. URL: https://ieeexplore.ieee.org/document/9613933/?arnumber=9613933 (visited on 10/31/2024).

Togelius, Julian, Emil Kastbjerg, David Schedl, and Georgios N. Yannakakis (June 2011). "What is procedural content generation?: Mario on the borderline". en. In: *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. Bordeaux France: ACM, pp. 1–6. ISBN: 978-1-4503-0872-4. DOI: 10.1145/2000919.2000922. URL: https://dl.acm.org/doi/10.1145/2000919.2000922 (visited on 10/23/2024).

Upadhyay, Uddeshya, Nikunj Shah, Sucheta Ravikanti, and Mayanka Medhe (Dec. 2019). *Transformer Based Reinforcement Learning For Games*. URL: http://arxiv.org/abs/1912.03918 (visited on 10/23/2024).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (Aug. 2023). *Attention Is All You Need*. URL: http://arxiv.org/abs/1706.03762 (visited on 03/10/2024).

Woolson, R. F. (2008). "Wilcoxon Signed-Rank Test". en. In: *Wiley Encyclopedia of Clinical Trials*. John Wiley & Sons, Ltd, pp. 1–3. ISBN: 978-0-471-46242-2. DOI: 10.1002/9780471462422.eoct979. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780471462422.eoct979 (visited on 08/18/2025).

Wu, Shengqiong, Hao Fei, Liangming Pan, William Yang Wang, Shuicheng Yan, and Tat-Seng Chua (Apr. 2025). "Combating Multimodal LLM Hallucination via Bottom-Up Holistic Reasoning". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 39.8, pp. 8460–8468. ISSN: 2374-3468. DOI: 10.1609/aaai.v39i8.32913. URL: https://ojs.aaai.org/index.php/AAAI/article/view/32913 (visited on 07/09/2025).

Yang, Yifei, Zouying Cao, and Hai Zhao (Nov. 2024). "LaCo: Large Language Model Pruning via Layer Collapse". In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Yaser Al-Onaizan, Mohit Bansal,

and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, pp. 6401–6417. DOI: 10.18653/v1/2024.findings-emnlp.372. URL: https://aclanthology.org/2024.findings-emnlp.372 (visited on 12/10/2024).

Zaidan, Tiago and W. Fabrício Luís Góes (2016). "Evolvestone: An evolutionary generator of balanced digital collectible card games". In: URL: https://www.semanticscholar.org/paper/Evolvestone%3A-An-evolutionary-generator-of-balanced-Zaidan-Fabr%C2%B4%C4%B1cio/ed3e64efef23f38c4d224f3d66d3a (visited on 05/05/2026).

Zeyer, Albert, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney (Dec. 2019). "A Comparison of Transformer and LSTM Encoder Decoder Models for ASR". In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 8–15. DOI: 10.1109/ASRU46091.2019.9004025. URL: https://ieeexplore.ieee.org/abstract/document/9004025 (visited on 07/09/2025).

Zhang, Yuji, Sha Li, Cheng Qian, Jiateng Liu, Pengfei Yu, Chi Han, Yi R. Fung, Kathleen McKeown, Chengxiang Zhai, Manling Li, and Heng Ji (Feb. 2025). *The Law of Knowledge Overshadowing: Towards Understanding, Predicting, and Preventing LLM Hallucination*. DOI: 10.48550/arXiv.2502.16143. URL: http://arxiv.org/abs/2502.16143 (visited on 07/09/2025).