UiA University of Agder

Master

Lost in Draft: Investigating Game Balance in Multiplayer Online Battle Arena Drafting

NGO VIEN BAO
VU HAO NHIEN

SUPERVISORS
Glimsdal Sondre
Granmo Ole Christoffer

University of Agder, 2021
Faculty of Engineering and Science
Department of Information and Communication Technology

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Vi erklærer herved at vår besvarelse er vårt eget arbeid, og at vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	**Vi erklærer videre at denne besvarelsen:** • Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • Ikke refererer til andres arbeid uten at det er oppgitt. • Ikke refererer til eget tidligere arbeid uten at det er oppgitt. • Har alle referansene oppgitt i litteraturlisten. • Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert.	Ja
5.	Vi er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	Ja

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).
Oppgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei

Acknowledgements

We would like to begin by thanking our supervisor, Sondre Glimsdal, for his invaluable guidance, support, and patience throughout this thesis.

A special thanks to Anders Ghouchbar and Valdemar Andersen, who helped proofread the thesis time and time again; special thanks to Nadine Lindvik Parra for letting us borrow her iPad to create our figures.

We would like to extend our thanks to Neuromancer and Flatline for their commitment. Without their drive to deliver results all hours of the day, this project would not have concluded.

Lastly, we would like to thank our respective workplaces for allowing us to take time off to focus on this thesis. The extra allocation of time has relieved us of immense levels of stress.

Thank you

Abstract

This thesis explores modern machine learning solutions to turn-based strategy games. In particular, we explore the possibilities of equalizing the playing field for both teams in the draft phase of Defense of the Ancients 2 (Dota 2) and League of Legends (LoL), with both games being giants in the multi-million dollar esports industry.

The thesis covers the Multiplayer Online Battle Arena video game genre and the draft phase the games use. We also discuss the technology used to address the problem, as well as the basic concepts of modern machine learning that allowed this technology to arise. We then introduce the Win Rate Predictor, which is our implementation of the reward function in the Monte Carlo Tree Search algorithm used to predict the win rate of each team given different parameters in the draft phase.

The results show clear and quantifiable differences in different parts of the draft phase. This includes reordering the pick order, the impact of including banning in the draft phase, and the balance of different draft schemes.

Specifically, first pick has a higher win rate than last pick for the majority of the draft schemes, suggesting that strong initial picks are more valuable than reactive response picks. Additionally, bans can be a way to influence the balance of a draft phase. Our simulations also suggest that the southwestern locations on the map have a higher win rate in both Dota 2 and LoL. And finally, according to our simulations, the games' respective implementation of a draft scheme is the most evenly balanced draft scheme for their game.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

What is balance, and how can we achieve it within a game? In general, game balance is adjusting the game rules to make the game fair for all players [1]. If a set of all players find the game challenging, one might change the rules to adjust the overall difficulty and satisfy these players. For example, imagine a group of friends are to play pick-up basketball. A known saying is that *"the winner stays on,"* meaning that the losing team sits out while the winning team plays a new team. First, even before they play, the group must split into teams of five players. A way to build these teams is to nominate *captains*, who alternates between picking players for their teams until the teams are full. Once the teams have been established, they can play the game. It is, in essence, a game within the game, namely *the draft phase* and *the play phase*.

In the draft phase, say all players are playing in the NBA[1], how can the captains draft to gain an advantage over the opposing team? A strategy might be to pick the player who is presumably the most skillful player left in the pool [2]. However, if those skillful players left in the pool continuously are offensive players, a team might end up with just offensive players with this strategy. Therefore, it is essential to draft a balanced team among themselves while being more robust

[1]The highest division of professional basketball in North America.

than the opposing team.

It is therefore important to draft a team which is balanced among themselves in addition to being stronger than the opposing draft.

These challenges are even more prevalent in the video game genre *multiplayer online battle arenas* (MOBA). MOBA games are split into the same two phases as the example above, illustrated in Figure 1.1. In the basketball example, there are not generally not than 20 players to pick from, but in the most notable MOBA games, there are over 100 heroes. A large number of possible combinations, synergies, and counters are a few points that make the draft phase of MOBAs difficult.

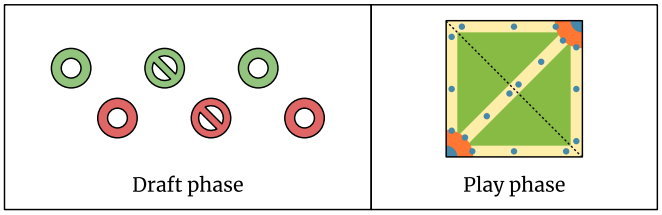

Figure 1.1: A visual representation of the two phases in MOBA game, the draft phase and the play phase.

Another important point that adds to the challenges in MOBA drafting is the two phases mentioned above. Due to human factors, it is difficult to predict the outcome of a given game with a high degree of certainty; the best one can do is to give an estimated winner. In contrast, existing works with board games such as Chess and Go have focused on the play phase as these games lack a draft phase.

Our proposed solution is to use a combination of reinforcement learning, self-play, and domain knowledge to explore known draft schemes in the draft phase of MOBA games. By considering the draft phase as a turn-based game, we can use Monte Carlo Tree Search (MCTS) as a search algorithm as it excels at these kinds of problems. It effectively narrows down the search space and prioritizes the most promising trees. More of this can be found in Section 2.4.1. Finally,

we adopt a proxy that estimates the winning team composition to deal with the difficulty of predicting the winner based on team compositions. This proxy will be further discussed in Section 3.1.

1.1 The Evolution of Computers Playing Games

Using computers to play games has been a hot topic ever since Arthur Samuel created the first computer program which was capable of playing Checkers in 1959 [3]. His pioneering work was one of the first to use heuristics search methods effectively. This program was able to assess the current board state and determine the best next move. Samuel started this project because games are less complicated than real-life problems and hoped that his studies could show how heuristic procedures and learning could be used together [4]. Little did Samuel know that his work would spiral and lead to the increasingly large interest in using computer programs to play games as we have today. Following his footsteps, others have managed to create programs that have been able to play other games such as Tic-Tac-Toe [5], Backgammon [6], Chess [7] and lastly, Go [8].

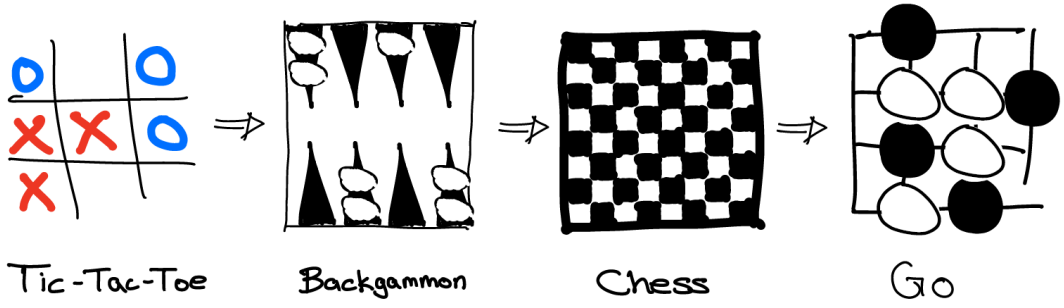

Figure 1.2: Visualization of the evolution in terms of computers playing board games.

An interesting point to note is that AlphaGo, the program playing Go, learns partly from self-play [8]. Its successor AlphaZero, learns entirely from self-play [9]. Another point, which makes AlphaZero an

important milestone, is that the game of Go is vastly more difficult to compute than other board games such as Chess. In comparison, a game of Chess has 10^{123} possible moves, whereas Go has 10^{360} possible moves [10].

Various video games of different genres have already been played at a superhuman level using computer programs, among these include, but not limited to, *StarCraft* and *StarCraft II* [11, 12] in real-time strategy (RTS), *Dota 2* [13] in MOBA, and *Doom* [14] in first-person shooter (FPS).

A common factor for many of the mentioned games is that they lack a second phase. Although research has been done in this field, it lacks in areas where there are two phases.

1.2 Motivation

The esport scene has become a multi-million business in the last few decades. Esport is a form of competing using video games and includes various game genres, one of its most significant being the MOBA genre. The MOBA genre is, in terms of monthly active players [15], one of the biggest genres in the gaming industry. Some of these events are The International (TI), an annual tournament in *Defense of the Ancients 2* (Dota 2), and Worlds, another annual tournament in *League of Legends* (LoL). In the most recent TI, 18 teams competed for a prize pool of $40 million while being watched by 2.7 million viewers live [16]. Whereas the final of Worlds 2021 peaked its concurrent viewership at 73.8 million viewers [17], a 60% increase compared to the previous Worlds. The esport scene has grown extensively over the last decade, and that the top players and teams can make a living from it. In comparison, the Super Bowl, one of the most-watched sporting events globally, peaked its viewership at 96.4 million viewers in 2021 [18]. Esports viewership numbers and large prize pool shows

how relevant it is today.

It is crucial to assess balance in all phases of a game, especially in a multi-million business as MOBA. Games such as Dota 2 and LoL, which have a large player base, big esports scene and a broad audience, must keep the game balanced to retain its competitive integrity. An unbalanced game affects the players directly, and a game can not survive without its players.

1.3 Thesis Description

Due to the comparable structure of a MOBA draft phase and a board game like Chess and Go, this thesis seeks to apply the self-play mechanism from the AlphaZero architecture to the draft phase in MOBAs. We wish to investigate how balanced existing draft schemes are and how they affect the game by utilizing this type of architecture. Our focus is on the professional scene of Dota 2 and LoL, as they are well-established in the industry, and use their drafting schemes as a baseline. By comparing other popular drafting schemes to these, we can better understand the drafting scheme's effect. Lastly, this thesis investigates whether these changed drafting schemes can be used as a tool to analyze the draft phase.

In short, this thesis aims to explore how drafting schemes affect the balance of the game in established games such as Dota 2 and LoL by using a combination of reinforcement learning, self-play, and domain knowledge.

1.4 Goals and Research Questions

The objective of this thesis is to create a model that explores alternative drafting schemes to balance the draft phase. We wish to explore the following research questions:

- **RQ 1**: Is the drafting scheme for Dota 2 and LoL balanced?
- **RQ 2**: How would modifying the draft scheme affect the win rate in Dota 2 and LoL?
- **RQ 3**: How valuable are bans in the draft phase for the two teams?
- **RQ 4**: Is there a trend in which positions get picked first or last?

1.5 Contributions

This thesis explores how the Monte Carlo Tree Search architecture can be coupled with a policy and value network to train itself by self-play, in which it is expanded to the draft phase in MOBA games. The following list summarizes the contributions:

- A sufficient MCTS model trained by self-play to simulate hero compositions for Dota 2 and LoL.
- An experimental environment for testing different MOBA draft schemes.
- Estimations of the balance of different draft schemes.
- Estimations of how different draft schemes affect game balance.

1.6 Outline

This thesis is structured into five different chapters. The chapters are as follows:

- **Chapter 2** introduces the underlying background of the MOBA genre and its draft phase. Followed by the technical theory for machine learning and its components used during this thesis.

- **Chapter 3** describes our methodology and is divided into two main sections. Section 3.1 explains our process collecting data needed to train our Win Rate Predictor (WRP). In Section 3.2 we build the Drafter which will then use the prediction from our WRP as input play the draft phase.
- **Chapter 4** presents our findings and analyzes the results in conjunction with the objectives set in Chapter 1.
- **Chapter 5** concludes our investigation and looks at future work.

Chapter 2

Background

This section provides some background on the MOBA genre and the two most popular games within this field, Dota 2 and LoL. Subsequently, we outline prior research on these two games and machine learning concepts relevant to the project.

2.1 Multiplayer Online Battle Arena (MOBA)

MOBA is a subgenre of strategy video games in which two teams battle against each other to destroy their opponents base. The first team to do so is considered the winner. The match is played on a predefined map with two sides. In Dota 2, the sides, named *Radiant* and *Dire*, are located in the southwest corner and northeast corner, respectively. In LoL they are named *Blue* and *Red* with the same positions as in DotA 2 on a similar map. In most MOBAs, each team consists of five players, where each player controls their own hero with their unique sets of abilities. Each player chooses their hero from a pool of heroes.

2.1.1 Play Phase

The generic MOBA map shown in Figure 2.1 is a square consisting of three lanes top, middle, and bottom lane. Each lane has three turrets

guarding the lane, starting from the base and extending outwards to the middle of the map. Due to its squared shape figure, the middle lane is naturally shorter than the other two lanes.

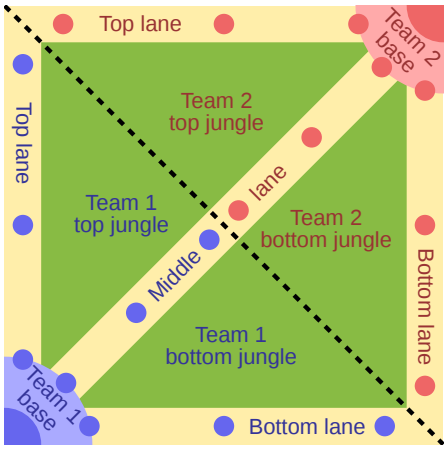

Figure 2.1: A generic map of the play phase in a MOBA game.

There spawns several creeps from the base in set intervals and move out through the lanes. A player can kill the opponent's creeps to gain gold and experience. A player does not gain gold or experience by killing their own creeps; some MOBA games do not even allow inflicting damage to their own creeps. In between the lanes is the jungle. The jungle is the home of neutral creeps, neutral because both teams can kill them in to gain gold and experience. This opens up room for you or the opponent to invade and steal gold and experience. The game's objective is to destroy the opponent's base structure, and the first team to do so is declared the winner. These are the base guidelines for most MOBA games. Each MOBA game has its style, including rules and features such as the map, hero pool, items, visuals, and other mechanics.

2.1.2 Draft Phase

In both Dota 2 and LoL, there is a pool of over 100 heroes for each player to pick from. These heroes have unique abilities, and they are also categorized within different roles giving them various strengths

and weaknesses. It is therefore essential to compose a well-rounded team.

Before a game can start, each player has to pick their heroes from this pool. To balance the game, the developers introduced bans to this phase. This phase will be called the *draft phase*. During this phase, the two teams alternate between picking and banning heroes. An example of how a draft phase can be performed is illustrated in the figure below, Figure 2.2, also known as a *draft scheme*.

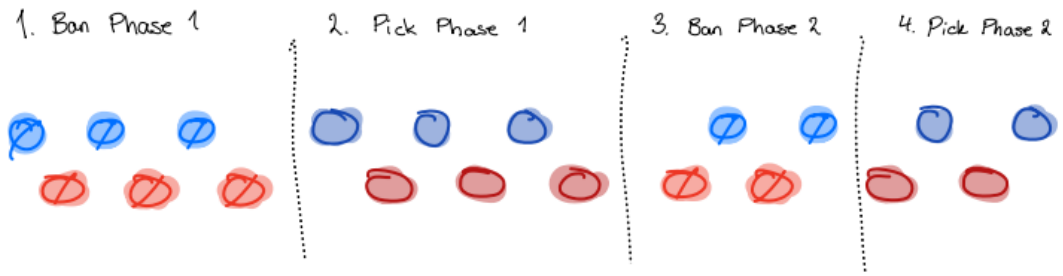

Figure 2.2: Example of a draft scheme. The teams are distinguished by the colors and the picks and bans are separated by the hue of the color. All draft scheme of this type starts from the left side at Ban Phase 1 with blue banning first, alternating between the teams towards the right side to Pick Phase 2. This type of draft will ban 10 heroes. B=Ban, P=Pick

The vast pool of totally different heroes makes it a complex combinatorial problem. It is also imperative to choose a strong combination of heroes from these combinations to beat the other team. Once the team has been drafted, the teams can play the game.

Each game has its own draft scheme, and some of the ones we will take a deeper look at are the tournament draft scheme for Dota 2 and LoL. These draft schemes will be further explained in the next section.

Draft Schemes

There are different draft schemes in MOBA. Ranked matchmaking games in Dota 2 use Ranked All Pick, while professional competitive

games primarily use Captain's Mode drafting. The selected schemes are further described in this section. We use Captain's Mode and Split Draft as a baseline ansd experiment on other schemes in our implementation.

Schoolyard

We define a custom draft rule with no bans, and each player alternate between picking. This scheme will provide insight into whether banning heroes has any effect.

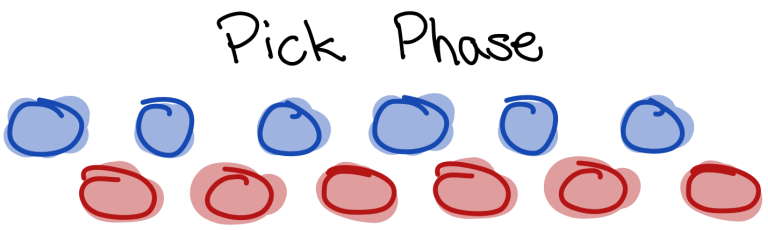

Figure 2.3: Schoolyard drafting, assuming that Player 1, colored in blue, is the starting team.

Captain's Mode

The draft rule used in Dota 2 professional games is the Captain's Mode drafting, shown in Figure 2.4. The two players alternate between banning in a total of four heroes. *P1* then gets to pick the first hero, and to compensate for the potential valuable first pick, *p2* gets to pick the next two heroes in succession. *P1* finishes the first pick phase with the fourth pick. The drafting again switches into a six hero ban phase. Next in the draft sequence is a pick phase, followed by a ban and final picks. Note that a team gets both the first and the last pick in two of these three phases.

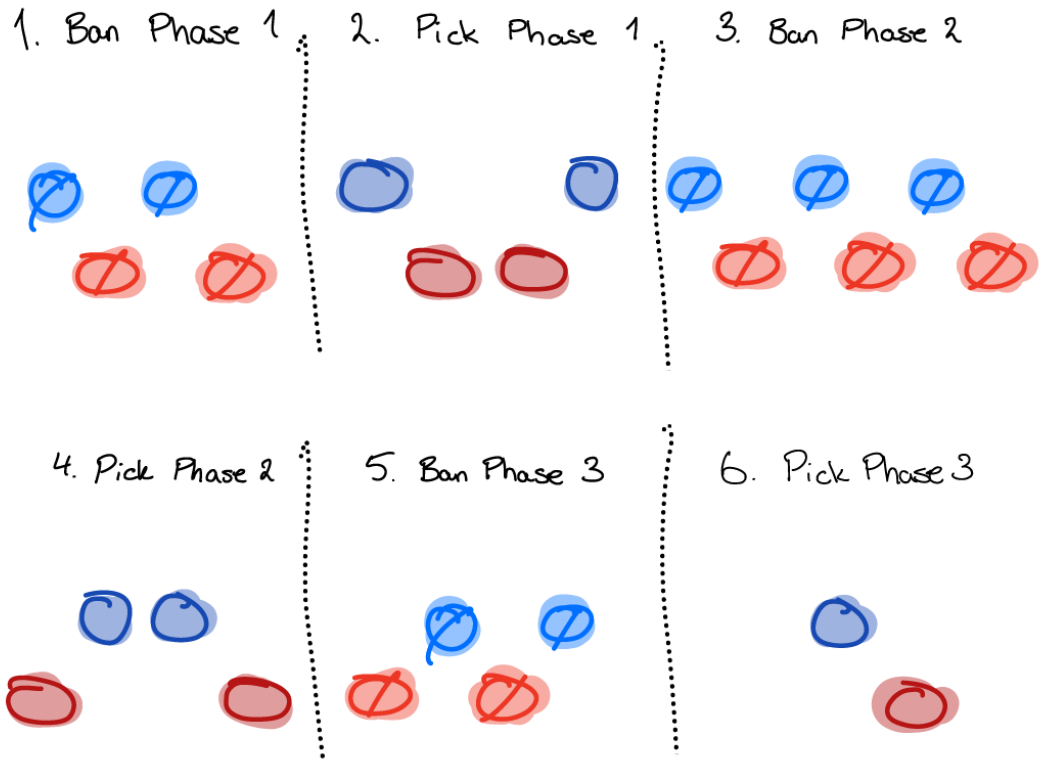

Figure 2.4: Captain's Mode drafting, assuming that Player 1, colored in blue, is the starting team. This draft bans a total of 14 heroes.

Split Draft

Split Draft is the draft used in professional LoL games, visualized in Figure 2.5. This scheme has only four phases, whereas Dota's Captain's Mode has six. In addition to fewer phases, this scheme has fewer overall bans, totaling ten bans. Note that this scheme does not let the same player pick first twice. This scheme has an equal amount of first picks and first bans split between the two teams.

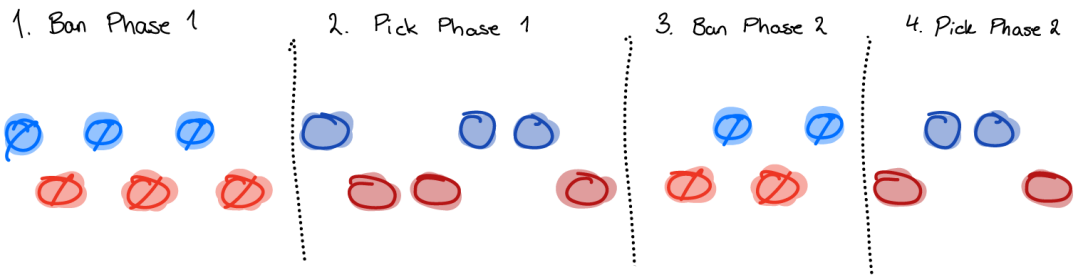

Figure 2.5: Split Draft. This draft bans a total of 10 heroes.

HoN Draft

The HoN Draft is the draft scheme used in tournament games in Heroes of Newerth (HoN). HoN is another popular MOBA game released in 2010 but will be discontinued on June 20, 2022. It is estimated to have between 10k-12k active concurrent players during the last few years. This draft scheme is identical to Schoolyard in terms of picking; the difference is that HoN Draft includes two ban phases, shown in Figure 2.6.

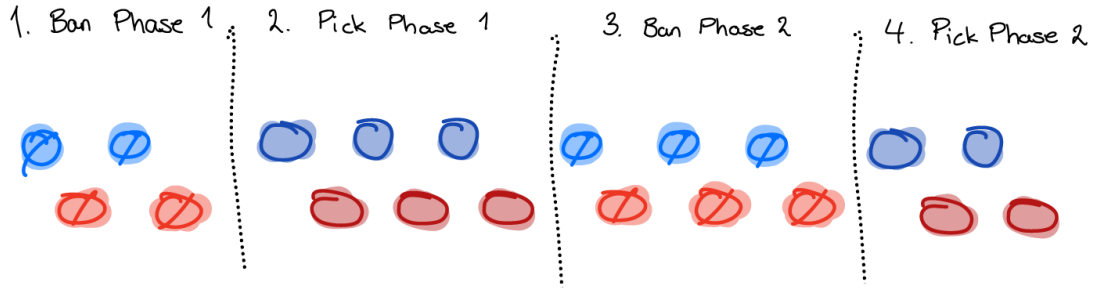

Figure 2.6: HoN Draft. This draft bans a total of 10 heroes, alternating each time between the two teams.

An interesting aspect about this draft scheme is that *P1* bans and picks first in all four phases of the draft.

2.2 Following the Meta

The meta is, generally speaking, knowledge of the latest strategic trends and its usage to create an optimal strategy to win. The meta game starts already in the draft phase as there are many variables to consider when a player picks their hero. Variables such as strengths, weaknesses, synergies, counters, and many others, are changed throughout the lifespan of a video game, thus changing the meta continuously.

The previously mentioned games, Dota 2 and LoL, regularly release new patches where they adjust these variables. They occasionally release new heroes to keep their respective game fresh and fun

to play. All of these changes are contributing to changing the meta. For example, a patch could introduce a new hero with strong synergy towards another existing hero. This synergy could potentially lead to an advantage. As people begin to the synergy and how to use it to gain an advantage, more people will follow and adapt to the new meta. The opposing team might then pick certain heroes to counter this synergy. For example, patches might be released that adjust the synergy if it is too strong. This cycle will continue until the game is balanced and the whole player base is happy (which will likely never happen).

The meta game can be so specific that it makes or breaks whether a team has a specific hero or not. Fortunately, this is not the case for many games. Both Dota 2 and LoL are in a healthy state with a healthy meta where each player has their role in the team. How each player contributes to their role is entirely up to them and their team.

2.3 Player Positions and Roles

The game publishers did not define any specific strategies for their respective games; instead, the games were published, and the players established and defined a meta. Once players gained experienced and became better at the game, some made videos and guides, tutoring other players and helping define the meta. Metas and strategies have shifted drastically throughout the years, but some things remain the same; the ideology of where a player plays on the map and the role affiliated with that position.

There are primarily five positions each player can choose to play. These positions go by different names depending on the game you play, but they fulfill the same duties. The roles are:

Carries; these players usually play a hero whose weak early but gets stronger the longer the game is. Their primary job in the

Lane	Dota 2	Role	LoL	Role
Bottom lane	Position 1	Hard-carry	Bot	Hard-carry
Mid lane	Position 2	Semi-carry	Mid	Semi-carry
Top lane	Position 3	Offlaner	Top	Offlaner
Jungle	Position 4	Soft-support	Jungle	Soft-support/jungler
Bottom lane	Position 5	Hard-support	Support	Hard-support

Table 2.1: Player positions and roles.

game is to deal damage. Damage is categorized into physical, magical and pure/true damage. Commonly, the carries' primary source of damage opposes the other. Carries are typically leading these charts after a game. These carries often need time to gather gold and experience in order to scale into late. Because of their weak early game, they are usually assisted by a support so that they can farm gold and experience easier. The *hard-carry* is almost always paired with a support in one of the side lanes while the semi-carry plays in the middle lane. The *semi-carry* can be left alone in the middle lane because the lane is shorter, and they tend to have a stronger early game or some safety measure.

Hard-supports; also known as just the *support*. The role of hard-supports is to keep their team alive and give them opportunities to gain extra gold and experience. The hard-supports generally have abilities to engage/disengage depending on the situation. Some also have strong healing or shielding spells, making them the perfect partner for the hard-carry. They normally wander around in the jungle in order to gain map control and keep track of the enemy soft-supports (jungler).

Soft-supports/jungler; while the other roles can gain gold and experience through clearing lane creeps, the soft-support clears neutral camps in the jungle for gold and experience (which is why they are commonly named the *jungler*). Their role is to assist the team by ganking a lane. Ganking a lane refers to

the strategy where one hero leaves their lane/jungle to create a numbers advantage in another lane which hopefully leads to an advantage later on. Soft-supports can roam together with the hard-supports to gain map control or even gank the soft-support while securing a neutral creep.

Offlaner; these players play in the last unoccupied side lane. This lane is just as long as the lane where the carry and hard-support plays, but unlike the carries, the offlaner does not have a support-player to assist them. Therefore, they must play a beefy hero who survives well on their own or an agile or mobile hero who can quickly get to a safe position and avoid ganks. This role is very versatile as many types of heroes fit this description. Three of the most common ones are;

1. *pushers* heroes who specialize in pushing down their lane to create an advantage. They bring down towers quickly and acquire map control, usually alone while the team focuses on other objectives.
2. *initiators* heroes who got abilities to lock down an enemy hero. They can usually start a team fight from a relatively safe position. Their strong lockdown abilities can consistently lock down a hero long enough for the team to eliminate that hero, causing a number advantage.
3. *durables* heroes who are capable of taking a heavy amount of damage. They tend to have a large health pool and have strong resistances. Important abilities with a long cooldown timer that normally deal a lot of damage to carries do typically not deal much damage to durable heroes. Therefore, their job is to be the meatshield and eat up these abilities for the team.

It is normal that a player primarily commits their time playing one position, maybe two. This way, they can fully immerse themselves into the role and master the game quicker. It is the same way in traditional sports; you do not usually train to play defense and offense. Instead, you specialize in one role.

2.4 Machine Learning Concepts

This section will describe the different machine learning concepts relevant to this thesis.

2.4.1 Monte Carlo Tree Search (MCTS)

MCTS is a heuristic search algorithm that is frequently used to play board games. The algorithm is useful in turn-based games such as Chess, Go, and Tic Tac Toe. Rather than searching through the whole tree as other traditional tree search algorithms, MCTS simulates the search and takes an action based on its accumulated reward. MCTS' main idea is to effectively search and prioritize the search trees with the most promising actions. The algorithm consists of four steps, illustrated in Figure 2.7 which are repeated: **selection, expansion, simulation** and **backpropagation**. In theory, MCTS wants to iterate until time or memory resource allocation is depleted.

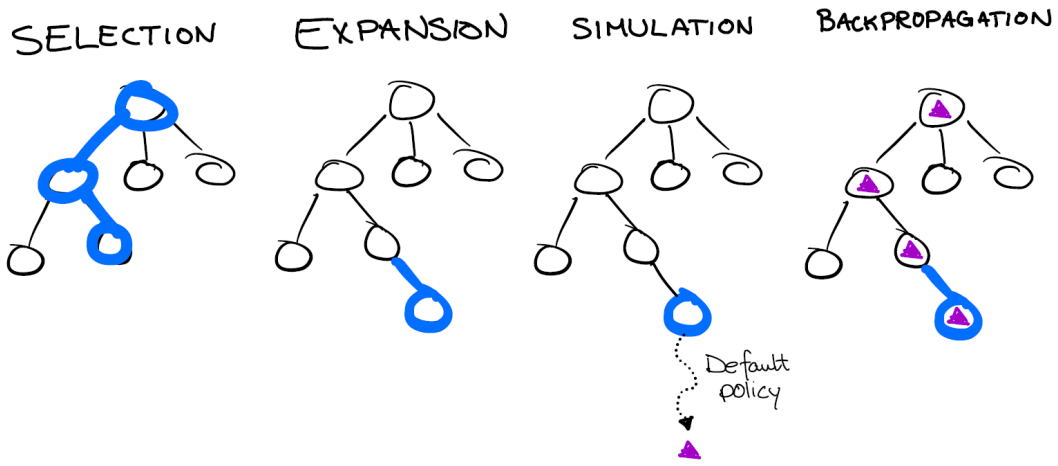

Figure 2.7: A visualization of the steps of MCTS adopted from [19].

1. **Selection**

   In the first step, the algorithm starts at the root node and selects a node to traverse. It traverses down until it reaches the terminal state, or it reaches an *expandable node*. A node is expandable if it is not terminal or if the node has unvisited child nodes.

2. **Expansion**

   If the traversion ended at a node which is not terminal, the algorithm will expand the node.

3. **Simulation**

   When the algorithm has expanded a node, it performs a rollout until a terminal state is found. While the algorithm does the rollout, it perform actions based on a default policy. The default policy could be random sampling. When reaching a terminal state, a reward is calculated and backpropagated. Note that during the rollout, all nodes that have been passed are not considered as visited.

4. **Backpropagation**

   The algorithm updates the value of all the nodes from the expanded node to the root node with the calculated reward. The

visit count of the said nodes are also incremented.

2.4.2 Upper Confidence Bound applied to Trees (UCT)

The most prominent challenging step of MCTS is its ability to select a child node in the simulation. What is difficult about this is the ability to maintain the balance between *exploration* and *exploitation*. MCTS should have a good balance of nodes with known rewards and unvisited nodes.

An example of the exploration/exploitation dilemma is the Multi-Armed Bandit problem. An agent has to choose actions (arms) to maximize its cumulative reward in the long term. For instance, a player faces three slot machines, each with a different reward distribution. The player wants to find the machine with the best payout without spending too much money. If the player plays all the machines once, then plays on the only machine that gave a payout, the choice could be sub-optimal; instead, the player should gather more information on the other machines by playing them more often. As a result, the player has to find a balance between exploitation and exploration to find optimal actions.

Upper Confidence Bound applied to Trees (UCT) is a proposed solution that addressed the balancing issue between exploitation and exploration and was introduced by Kocsis and Szepesvári in 2006 [20]. The formula is defined below in Equation (2.1):

$$UCT_j = X_j + C \times \sqrt{\frac{ln(n)}{n_j}} \tag{2.1}$$

where:

- X_j is the win ratio of the child.
- n is the number of times the parent has been visited.
- n_j is the number of times the child has been visited.

- C is an adjustable exploration-constant. Theoretically equal to $\sqrt{2}$.

2.4.3 Predictor + UCT (PUCT)

PUCT is a modified version of the aforementioned UCT algorithm. It was introduced by Rosin in 2011 [21], and later refined by Silver et. al in 2017 with AlphaGo [22]. PUCT estimates the policy $\vec{p}_\theta(s)$ given a state s. The upper bound confidence is selected by the calculations of:

$$U(s,a) = Q(s,a) + c_{puct} \times P(s,a) \times \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)} \tag{2.2}$$

where:

- $Q(s,a)$ is the expected reward for taking action a from state s.
- $N(s,a)$ is the number of times action a has been taken from state s.
- $P(s,\cdot) = \vec{p}_\theta(s)$ is the initial estimation taken from the policy which is returned by a NN.
- c_{puct} is the adjustable parameter that controls the degree of exploration.

2.4.4 Policy and Value Networks

An artificial neural network is a network connected with nodes, similar to the neurons in the brain. The most basic form of an ANN consists of three layers of interconnected nodes: input, output and a hidden layer between the two. In a simple ANN the hidden layer only consists of one layer.

Policy and Value Networks are NN and were first introduced as a new search algorithm in MCTS simulations by AlphaGo [8]. In

short, a *value network* evaluates the board state and the *policy network* selects a move. The general idea is that the network will learn what board states could lead to wins (or losses). The successor AlphaZero combine the two network into one to allow it to be trained and evaluated more effiently [9].

2.5 Related Work

In 2018, Chen, Z. et al. published a paper [23] proposing a hero recommendation system that suggests a hero which maximizes the team's prospect for victory. The proposed model uses MCTS for estimating the values of hero combinations and a classification model as their reward function. They experimented with three different classification models, Gradient Boosted Decision Tree (GBDT), Neural Network (NN) and a generalized linear model, LR, where their NN scored highest with a prediction accuracy of 0.65345. Continuing with NN as their reward function they are able to confirm that MCTS-based recommendations leads to stronger hero compositions compared to other baselines. Some of its limitations include not considering the hero position, but rather the complete draft.

DeepMind created a model named OpenAI Five [13] in 2019, which is a system that has its primary focus on the play phase of Dota 2. OpenAI Five is trained to play the best game possible given a random selection of heroes. It narrows down the pool of over 100 heroes to 17 and uses a Minimax algorithm to select the best hero given the board. This algorithm plays out the draft phase randomly without much thought for robustness in evaluations against human players. Its hero pool size of 17 makes it a limitation given the otherwise large pool size.

Another paper that used MCTS was published by Chen, S. et al. [24] two years later, in 2020. This proposed model, named JueWu-

Draft, applies MCTS and neural network for finding the optimal hero draft. JueWuDraft utilizes PUCT as the search algorithm as opposed to Chen, Z. et al. who used the default policy, which is default set to random sampling. JueWuDraft did not take the ban phase into consideration, which is a key element in MOBA games today.

The same year, 2020, Tomašev et al. published a paper [25] about exploring alternative rule sets in Chess with AlphaZero. In essence, the paper is about exploring and defining new rules for Chess and lets AlphaZero train and learn with new rule sets in order to assess the balance in Chess. Small changes in the rule sets allows AlphaZero to develop new strategies and tactics while maintaining the integrity of Chess.

Chapter 3

Methodology

This chapter explains our methodology concerning the objectives set in Section 1.4. The chapter will be divided into two sections. The first section dives into what the Win Rate Predictor (WRP) is, how we use it as a reward function, and why it is fundamental for our thesis. Finally, the last section explains how our MCTS Drafter works and the steps implementing it.

3.1 Win Rate Predictor as a Reward Function

Once the MCTS algorithm reaches its terminal state, it outputs a complete draft, it uses a reward function to determine a certain winner, but since the play phase has not been played, it can not do so. Thus, we adopt this proxy for the reward function, namely the WRP. In short, the WRP estimates which team has a higher probability of winning the game based on the draft phase. A classifier returns the estimation and is trained individually for Dota 2 and LoL with their respective match datasets. The match dataset contains information about the team composition and which team won the game. The WRP trains exclusively on the team compositions with positions to predict a winner. The order of picks and bans is only relevant for the Drafter, not the WRP, which will be further discussed in Section 3.2.

The processes of training the WRP for Dota 2 and LoL are different. Thus the following sections are split respectively. The sections will provide the necessary information about the differences and what obstacles emerged.

3.1.1 Dota 2 Overview

Which positions each hero plays is important for this project, and the raw match dataset does not contain this type of information. To get this information, we need a lane dataset to train a position classifier, which enriches the match dataset with positions. Lastly, we use the enriched match dataset to train the WRP. Figure 3.1 show an overview of the process of training a Dota 2 WRP. We have divided the process into the following three phases:

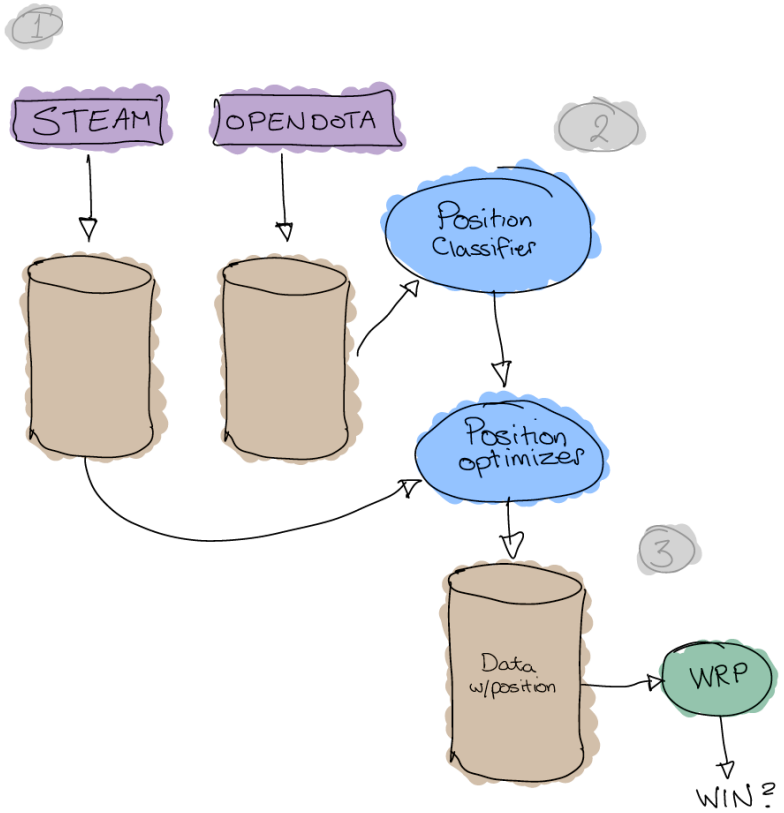

Figure 3.1: Process of training the Dota 2 WRP.

1. Gather data from two APIs to create two datasets, one for train-

ing the position classifier and one for training the Dota 2 WRP, explained in Section 3.1.1.

2. Train a position classifier that predict the player positions, further explained in Section 3.1.1.
3. Finally train the WRP with the enriched dataset.

Data Collection

We propose two datasets for Dota 2, a lane dataset for training the position classifier and a match dataset for training the WRP.

Match Dataset

We used the Steam WebAPI to gather 25 million matches played between 18 August and 28 October 2021. The matches are based patch *7.30* up to patch *7.30d*. We did not collect matches from patch *7.30e* and after as they introduced adjustments that could change the meta and new heroes. Each match in the dataset contains the heroes in the match, as well as the attributes mentioned in Table 3.1. The matches fulfill the following conditions:

- The game mode must be either All Pick or Ranked Matchmaking
- The match duration is longer than 20 minutes and less than 60 minutes.
- The match has no leavers. All players must be present during the whole match with a total of 10 players.
- The player skill level is "high skill" or "very high skill".

We used the position classifier to enrich the original match dataset with player positions to create the final match dataset.

Lane Dataset

OpenDota is an open-source platform that provides Dota 2 data and analytics. The platform collects data from the Steam WebAPI and also extracts more detailed data from match replay files [26]. Each match in the dataset contains the same information as the dataset mentioned above, but with additional information such as which lane a hero played during the game. We collected 50k matches from the OpenDota API.

Following the meta, we assume that every player in a team has designated positions, mentioned in Section 2.3. We did feature engineering on the original lane dataset as it only contains lane information and not player positions. Equation (3.1) show an example of lane information in the data obtained from OpenDota.

$$team_n = \begin{cases} hero_1, hero_5 & 1\text{: safelane} \\ hero_2 & 2\text{: midlane} \\ hero_3, hero_4 & 3\text{: toplane} \end{cases} \tag{3.1}$$

Within a team, we rank every hero from 1 to 5, where 1 being the highest rank and 5 the lowest, in the following attributes; *gold per minute (GPM)*, *experience per minute (XPM)*, *kills*, *deaths*, *assists*, *wards placed*, *last hits*, *hero damage* and *tower damage*, which are described in Table 3.1.

We assume that heroes played in position 5 is the highest *wards placed* rank and one of the lowest *GPM* rank. We also assume that the hero with the highest *GPM* rank in a team play as position 1. The following steps describe how we label the heroes with positions:

1. If lane 1 has more than one hero, the hero with the highest *GPM* rank will be labeled as position 1, leaving the other heroes as position 5 candidates.

Attribute	Description
GPM	Gold per minute obtained by a player
XPM	Experience Per Minute obtained by a player
Kills	Number of kills
Deaths	Number of deaths
Assists	Number of assists
Wards placed	Sum of all
Last hits	Number of last hits
Hero damage	Total damage dealt on the opponent heroes
Tower damage	Total tower damage done by the player

Table 3.1: Selected attributes used for ranking

2. If lane 2 has more than one hero, the hero with highest *GPM* rank will be labeled as position 2, and the other heroes will be marked as position 4 candidates.

3. If lane 3 has more than one hero, the hero with highest *GPM* rank will be labeled as position 3, and the other heroes will be marked as position 4 candidates.

After going through the three lanes, we are left with position 4 and position 5 candidates. Among the position 5 candidates, the candidate with the highest rank in *wards placed* is labeled as position 5. The same method is also done to label the position 4.

We processed all the matches in the original dataset and created one dataset for every hero, which is 121. Each data point is a feature matrix where its size is a multiplication of the total number of features by the five available ranks. The target label is the assigned position. The final dataset was used to train the position classifier.

Finding the Player Positions

The position classifier is an ensemble of 121 logistic regression classifiers trained on the lane dataset mentioned in Section 3.1.1. Each classifier predicts the log probability for each position for a hero, given their ranking within their team. For example, within a team, the hero

Phantom Assassin ranks number 1 in GPM, XPM, and kills, and rank number 5 in wards placed. The classifier for Phantom Assassin will most likely return a high probability that the hero plays as position.

We created an algorithm we call Team Position Optimizer (TPO) to enrich the match dataset. We explain how TPO works with the following example. Given the following team; Treeant Protector (TP), Batrider, Tidehunter, Lion, Morphling, we calculate the ranks on the attributes mentioned in Table 3.1. We acquire the position probabilities for all the heroes with the ranks by using the position classifier. At this point, we sum the probabilities for every permutation of the team. A permutation of the team could for instance be TP (position 5 p=*0.7*), Batrider (position 2 p=*0.8*), Tidehunter (position 3 p=*0.6*), Lion (position 4 p=*0.8*) and Morphling (position 5 p=*0.9*). Assuming that the permutation had the highest sum, we finally assigned the heroes' given positions.

3.1.2 LoL Overview

The general flow of how WRP was trained for LoL is visualized below as Figure 3.2. The process are divided into three steps.

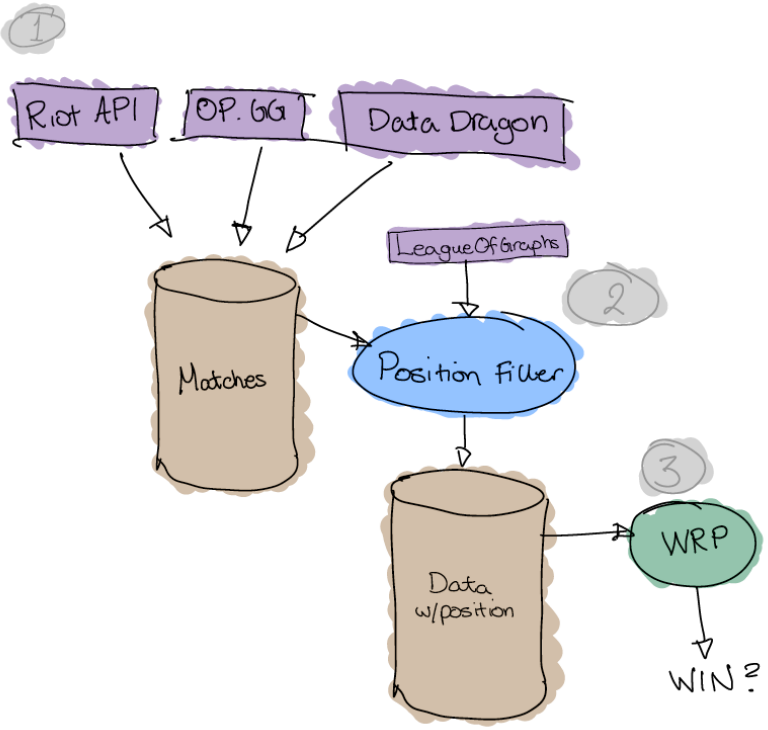

Figure 3.2: Process of training the LoL WRP.

1. Request and get the data needed to train the WRP for LoL from the developers API. However, additional data was needed as input to request the match data. These additional data were gathered from *OP.GG* [27]. Once the data has been collected, it was validated against the hero data from *Data Dragon* (ddragon) [28].
2. The validated data is then run through a function that fills in missing player positions to build a new desired feature vector.
3. Finally, the feature vector is used to train the WRP classifier.

Data Collection

One way to scrape LoL matches is through the game developers, Riot Games' API. A development user was created in order to gain access to their API. It is not possible to get the complete match data by calling a single API endpoint. In order to retrieve the match info, an API call was made to three different endpoints with appropriate

parameters. These three endpoints are seen in Table 3.2.

Endpoint	**Returns**
/summoner/v4/summoners/by-name/*{summonerName}*	The users PUUID
/match/v5/matches/by-puuid/*{puuid}*/ids	n Match IDs
/match/v5/matches/*{matchId}*	Match info

Table 3.2: Table of Riot API endpoints

Before making requests to the mentioned endpoints, we needed to collect summoner names to send along the requests. The Riot API does not allow to get large numbers of summoner names efficiently, so it was scraped off a third-party website named OP.GG [27]. OP.GG is a website that provides insights to gamers for games such as LoL, PlayerUnknown's Battlegrounds, and Overwatch [27]. Riot has eleven different platforms scattered globally where a player can own an account. It was necessary to scrape all platforms to get more accurate data. Therefore, we iterated through all platforms and scraped OP.GG's first 20 pages of the leaderboard for that platform. Each page consists of 100 players, which gives 4400 players globally, 400 players for each platform. This list of player, or summoner names in this context, will be the first building block needed for data collection.

The Riot API uses three different IDs for players; summoner IDs, account IDs, and PUUIDs. Since Riot has several servers worldwide in different regions, we had to retrieve the globally unique ID, the PUUID. By sending along the summoner name, its correspondent platform, and n number, where n is the desired number of matches for that player, initially set to 20, we get a list of n match IDs. These match IDs are then sent along with the platform to retrieve the complete match info. This process is repeated until all match IDs are iterated through for all players on all platforms.

This process was very long and tedious due to Riot API's rate limit of 20 requests every 1 second and 100 requests every 2 minutes

[29]. We encountered our first problem due to this rate limiter. The first 20 pages of the leaderboard that was scraped earlier represent approximately 0.21% of the total player base for that platform [30]. There are remarkably few players at this high rank, which effectively means that they are playing many games against the same players. Thus, some of the first 20 match IDs that were scraped for one player may match one or several other players' match IDs. This means that we are using the rate requests to collect many of the same matches. One could, in theory, ignore this problem and filter out the duplicated matches at a later stage, but considering how slow the process was due to the rate limiter, the problem was addressed. A function was made to compare the list of n match IDs against each other and remove the duplicates, thus only calling the API with unique matches IDs.

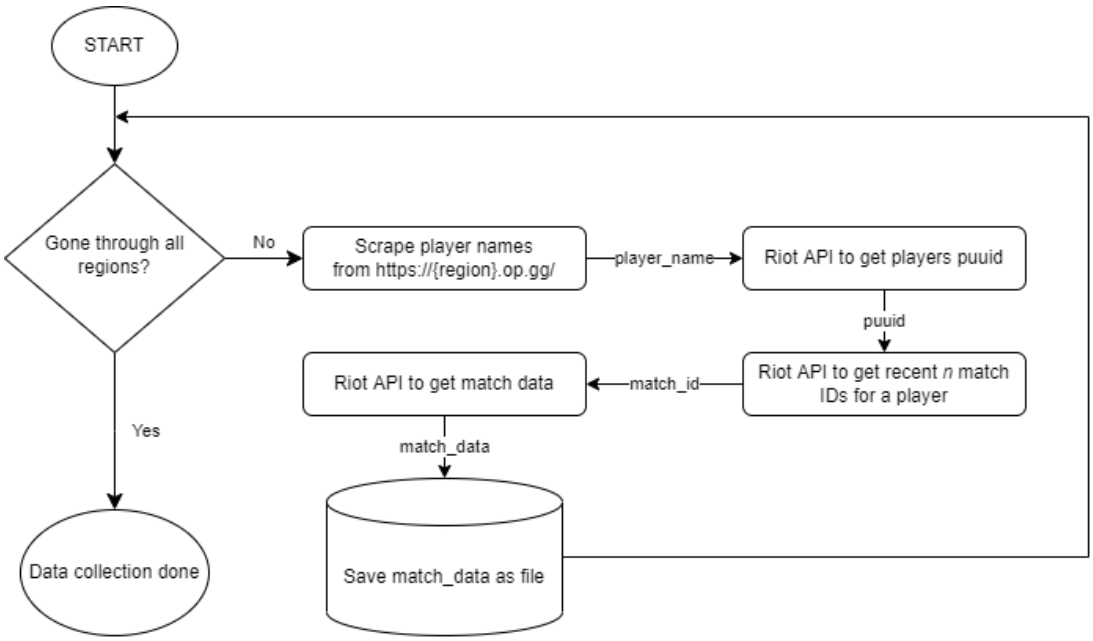

Figure 3.3: Flowchart for data collection for LoL.

Once all the duplicate match IDs were removed, they were looped through as input to an API call with the appropriate platform as an additional argument. The match data contain valuable information,

such as game mode, game type, participants, which heroes were picked by who and where those heroes played on the map, and most importantly, which team won the game.

LoL has multiple game modes and types, and we are only interested in games played with the original MOBA game mode. Another condition we want is only to have high-ranked matches, but this condition was fulfilled when we scraped the top of the leaderboard and their matches. The list of conditions for a valid LoL match are as follows:

- The patch the game is played on must be no newer than 11.18.1, which is the latest patch before a new hero was released.
- The game mode must be "CLASSIC", meaning the normal 5v5 MOBA game mode.
- The game type must be "MATCHED_GAME", which is with a ranked system for players. This mode does not allow players to play the same heroes as the opponent.

Lastly, the dataset was checked against the hero information and its IDs from ddragon to validate it. The validated and filtered dataset totaled around 110k matches.

As mentioned earlier, the Riot API provides information about the player position. Unfortunately, there are matches where this information is lacking or invalid. Looking through the data, we can see that there is six valid input variable as positions:

1. BOTTOM
2. MIDDLE
3. TOP
4. JUNGLE
5. UTILITY

6. INVALID

From Section 2.3, we can recognize that the first five input variables resemble each of the positions a player can play on the map. The variables are computed as the best guess for which position the player played by Riot [29], but they sometimes return *INVALID*. A function was created to catch matches in which a position or positions are *INVALID*. If there is only one invalid position, it is changed to fill the missing position. If there are two or more invalid positions, a lengthier procedure is initiated.

First, the position distribution of all heroes is scraped from another third party website named League of Graphs [31], which is a website that tracks millions of LoL matches played every day to gather champion stats, popularity, among other things. A list of available heroes and their ID, gathered from ddragon [28], was used together with this information to calculate the probability of that hero playing in that invalid position. This algorithm selects the positions greedily based on which hero has the highest probability. The process is repeated for the remaining heroes and the invalid positions until all positions are occupied and valid for all matches.

3.1.3 Feature Vector

Once data has been collected and missing positions have been filled, we can start building the structure for the data. There are many different features available from the two datasets retrieved earlier. We selected a few features which adhere to the thesis' objectives, and more can be added at a later stage if needed. The following features that were selected as a foundation were:

- Hero ID
- Which team *hero* are playing for (team 1 or 2)

- Role position for said hero
- Which team won (team 1 or 2)

The next step is to encode this information in a meaningful manner. Ideally, we wish to reduce the size to increase efficiency while keeping necessary information about the features. Another important aspect is that the dataset should differentiate which positions a hero is playing, or else our Drafter might draft a team composition with several heroes with the same roles. Our proposed data structure is made by one-hot encoding the heroes with their respective positions in a large list depending on which team they are on and appending which team won as the last element. We multiply the total number of heroes with the five roles available for each hero, and lastly, append which team won.

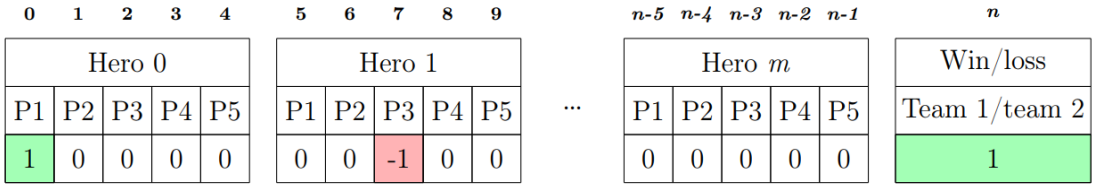

Figure 3.4: Visualization of encoded data.

The visualization in Figure 3.4 shows that hero with ID 0 is playing for team 1 at position 1, whereas hero with ID 1 is playing for team 2 at position 3, and also the fact that team 1 won. Each encoded sample is $m + 1$ where m is the number of heroes available, which differs whether the data comes from Dota 2 or LoL.

3.1.4 Training the WRP

The next step is to train the classifier that our MCTS implementation uses as a reward function. The WRP uses LR to classify the draft and returns 0 and 1. In a MOBA match, the play phase determines the match outcome. However, the draft phase can only give an advantage

or disadvantage heading to the play phase. Two identical drafts can play out the match completely differently and, therefore, get different results, making the WRP an estimation rather than a definite prediction. The accuracy that the WRP got from the datasets is presented in Table 3.3.

	Dota 2		LoL	
Model	Train	Test	Train	Test
LR	0.589	0.5786	0.5636	0.5420

Table 3.3: Results of WRP for Dota 2 and LoL. cv=5

3.1.5 Evaluation

To evaluate the WRP, we took hero compositions from high-skilled games for Dota 2 and LoL, used it as input in our WRP, and compared the results.

The match seen in Figure 3.5 is a Dota 2 match between high-ranked players. In this particular match, the dire team won. The left-side team (radiant) is *Team 1* and right-side (dire) is *Team 2*. The two team compositions are used as input in our WRP, and the results can be seen in Table 3.4.

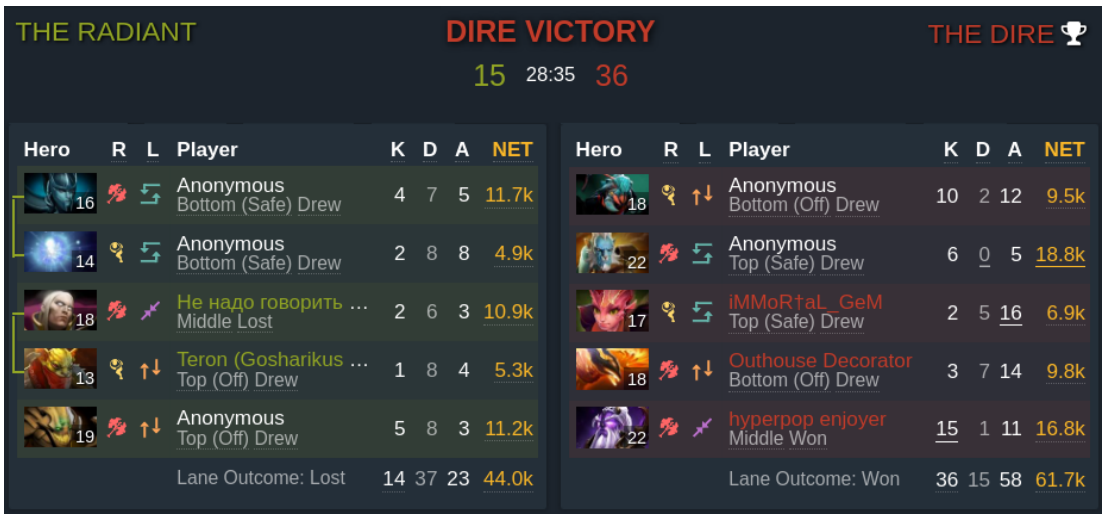

Figure 3.5: An example of a Dota 2 match result obtained from [26]. The Dire side won this particular match.

	Safe	Mid	Off	Soft-support	Hard-Support	WRP	Win Rate
Team 1	Phantom Assassin	Invoker	Sand King	Bounty Hunter	Io	Loss	0.4446
Team 2	Phantom Lancer	Void Spirit	Phoenix	Weaver	Dark Willow	Win	0.5554

Table 3.4: The outcome and win rate predicted by Dota WRP, with the teams in Figure 3.5 as input. The WRP predicted Dire win with a win rate of 0.5554.

The match from Figure 3.6 is from a real high-skilled game, with a distinctive winning team. The left-side team is *Team 1* and right-side is *Team 2*. The two team compositions are used as input in our WRP, and the results can be seen in Table 3.5.

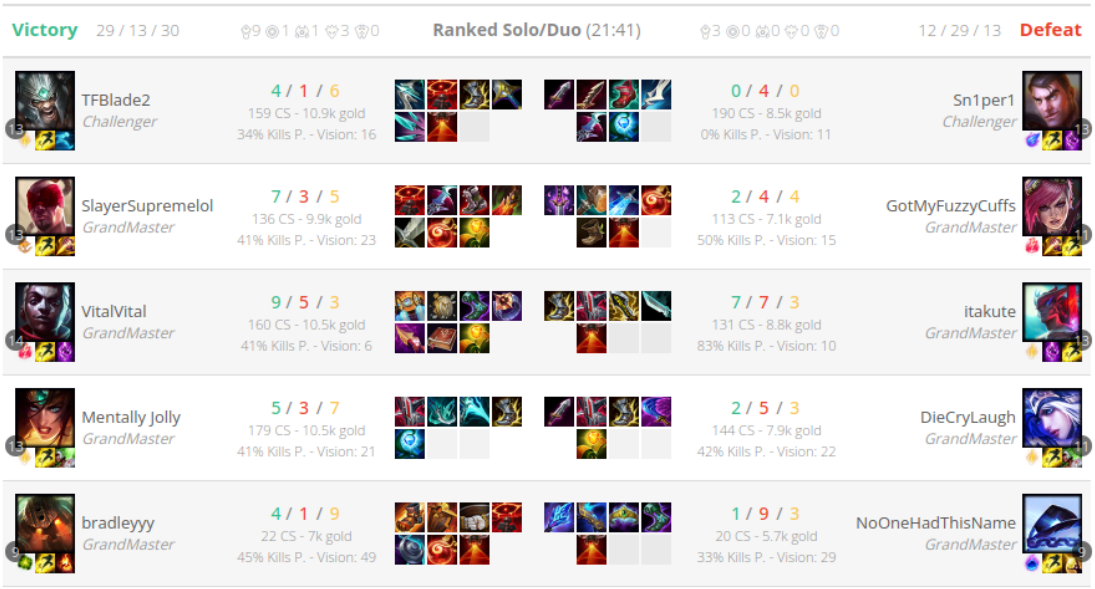

Figure 3.6: An example of a LoL match result obtained from [32]. The blue side (left) won this particular match.

	Offlane	Jungle	Soft-Carry	Hard-Carry	Hard-Support	WRP	Win Rate
Team 1	Tryndamere	Lee Sin	Ekko	Sivir	Nautilus	Win	0.5227
Team 2	Jayce	Vi	Yone	Ashe	Xerath	Loss	0.4773

Table 3.5: The outcome and win rate predicted by LoL WRP, with the teams in fig. 3.6 as input. The WRP predicted Team 1 to win with a win rate of 0.5227.

3.2 Learning to Draft by Using MCTS and NN

This chapter details how we applied the draft phase into an AlphaZero inspired implementation we call *Drafter*. Further in this chapter, we

describe how we used the *Drafter* to train MCTS models for Dota 2 and LoL for use in our experiments.

3.2.1 Drafter Setup

We implemented a MCTS algorithm with a NN as a policy and value network. The network is a fully connected ReLU network with three hidden layers which take in a board state. We have two heads on our network, a policy head and a value head. The policy head is a fully connected layer with a softmax function, which outputs the probabilities for the actions. The value head is a fully connected layer with a tanh function that outputs the value of the current board state.

In draft schemes used in tournaments, two teams alternate between picking and banning, but sometimes a team can pick or ban in succession. Therefore, we reconstructed the board state to recreate the draft phase in the Drafter. The board state contains information of the picks and bans in which is visualized in Figure 3.7. The values of the board are seen from the perspective of the current player. For instance, the board state $[0, 1, 0]$ is seen from the perspective of player 1. The board state for player 2 is then $[0, -1, 0]$.

The *pick array* is a 1-dimensional array with 121 indices. Each index represents a hero. The pick array is initially filled with zeroes, indicating that all heroes are available for picking. The *p1* and *p2* pick action are denoted with 1 and -1 respectively.

The ban array is a 1-dimensional array with the same shape as the *pick array*. The initial state of the ban array is 121 zeroes. A ban is denoted with 1 when a player bans a hero. We multiply the *pick array* with *ban array* to mask out the banned heroes from *pick array*. The game is terminal when both players have selected five heroes.

In order to incorporate the draft phase into the MCTS, we persist the next action in the board state. The next action is a number

ranging from 0 to the number of actions in a draft scheme. Every time a player selects an action, the number increment by 1.

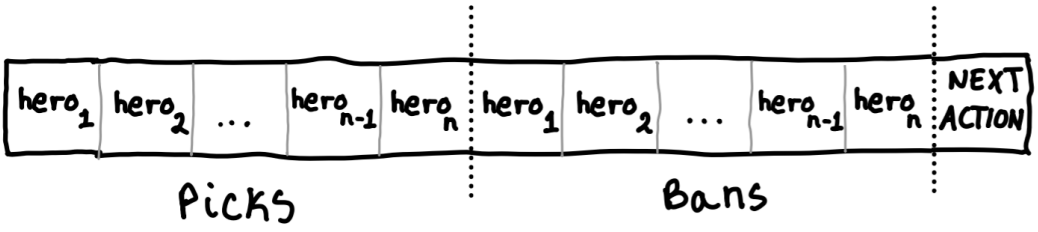

Figure 3.7: The board state. The state is seperated in three sections; picks, bans, and next action. n is the number of heroes available.

3.2.2 The Steps of the Drafter

Considering one simulation, we break down how the Drafter works. We divide the steps into the following three phases shown in fig. 3.8: selection, evaluation and expansion, and backpropagation.

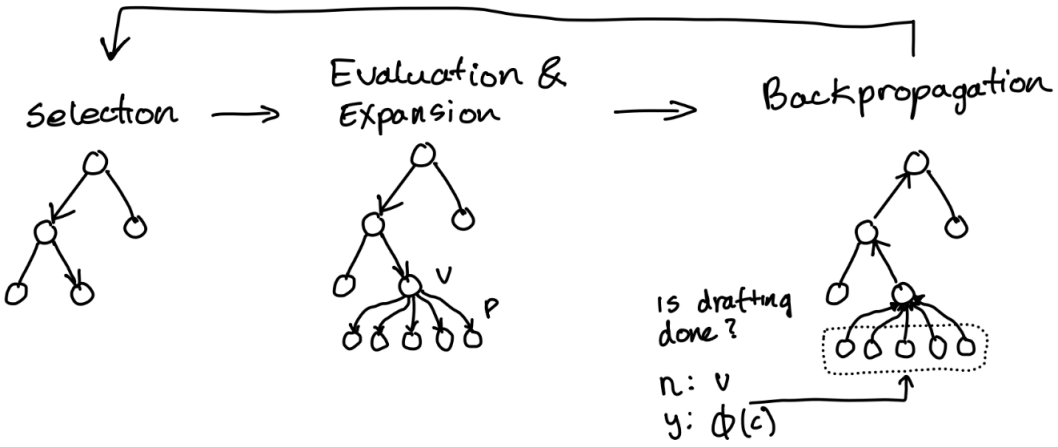

Figure 3.8: An illustration on how the MCTS use the policy and value network.

Selection

In this phase, the algorithm starts the simulation from the root node for a current board state. It traverses down the tree by selecting nodes according to the visit counts and value in the tree by using the PUCT algorithm mentioned in Section 2.4.3.

Evaluation and Expansion

The algorithm asks the policy and value network for the prior probabilities for every possible legal move it can choose, denoted as p and v. All the prior probabilities are then added as child nodes to the root node. The child nodes have not been explored yet by the algorithm. If the board state is terminal, the value from the NN is replaced with the predicted winner by the WRP.

Backpropagation

After expanding the node, the tree will update the statistics on visit counts and action-values for all the visited nodes.

Play

Lastly, the algorithm selects a move to play from the root position when the search has ended.

3.2.3 Training the MCTS Models with Self-Play

Figure 3.9 shows the self-play process. Starting from the top left corner, the *Drafter* performs tree search until the search reaches a terminal state, explained in Section 3.2.2. All board states leading up to terminal state during self-play, including the predicted winner that we get from the WRP, are used as train data. We create mini-batches of the train data to train the policy and value network. The process is then repeated.

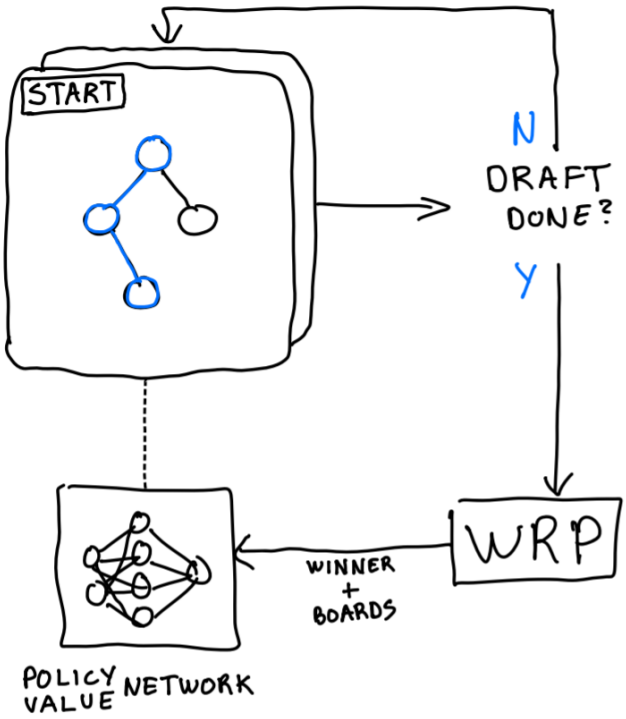

Figure 3.9: A general overview of how the drafter play itself to become better at drafting.

We trained two models, a Dota 2 model trained on Captain's Mode and a LoL model trained on Split Draft. Both models were trained over five days on a single NVIDIA 2080TI GPU. Each model generated around 1.5 million board states of self-play with 300 simulations per MCTS. Each simulation took 4 seconds. Figure 3.10 shows the policy and value loss for the Dota 2 model trained on 400 mini-batches of 4096 board states. We made a Random Model that outputs random prior probabilities and a random value to evaluate the trained models.

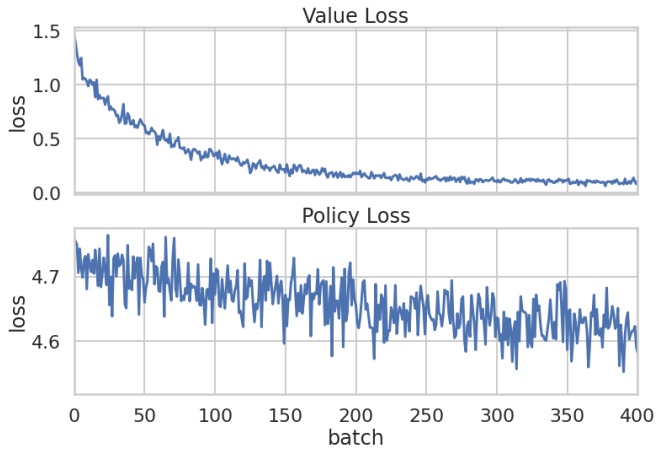

Figure 3.10: Dota 2 model's value and policy loss curves per batch. Batch size was set to 4096.

At this point we have three different models, all of which have been trained in different ways. The three models are as follows:

Dota 2 model (m_{dota})	Trained model which has been trained on Dota 2's drafting scheme, Captain's Mode.
LoL model (m_{lol})	Trained model which has been trained on LoL's drafting scheme, Split Draft.
Random model (m_{random})	A model where the policy network returns random probabilities for which action to take, and the value network values the current state randomly.

Chapter 4

Experiments, Results, and Discussion

The objective of this thesis is to analyze how different draft schemes affect the game. From this, we can derive and answer different questions, like how influential a draft scheme is to the game. This section will present the results of several experiments to evaluate the balance in different draft schemes. To this end, we will utilize our trained MCTS models and let them play themselves.

We pitted the trained models (mentioned in Section 3.2.3) against each other for 100k games where each player had up to 20 simulations or a time constraint of 30 seconds before selecting an action. The number of simulations and time constraints were set low to match the time constraint given in the draft phase of the different games[1]. We performed all the experiments on a single NVIDIA RTX 2080 Ti. In some of our experiments we pitted one of our trained models, m_{dota} or m_{lol}, against m_{random} on four different draft schemes. The four draft schemes, explained in Section 2.1.2, are the following:

Captain's Mode	Professional draft scheme for Dota 2.
Split Draft	Professional draft scheme for LoL.

[1]A player in the draft phase does not have infinite time to think about what to pick or ban. 30s for Dota 2 and 27s for LoL.

Schoolyard	Alternating pick order, no ban phase.
HoN Draft	Alternating pick and ban.

By setting all three models to play against each other on four draft schemes, we can better understand how each draft scheme affects the game. The following chapter will provide results from the experiments and a brief discussion.

4.1 Results for Dota 2

The data used to train our WRP shows that the Radiant side has an overall win rate of 52.28% over the Dire side. With this in mind, we know that the Radiant has an advantage. The conducted experiments aim to explore if different draft schemes affect this ratio.

We had the trained m_{dota} play against itself. After 100k games, player 1, denoted by *first pick*, had a mean win rate of 50.5%, shown in Table 4.1, which corresponds to historical statistics about Radiant having a slight win advantage.

Draft Scheme	**First Pick**	**Last Pick**
Captain's Mode	0.505	0.495
Split Draft	0.412	0.588
Schoolyard	0.509	0.491
Alternating Draft	0.546	0.453

Table 4.1: m_{dota} vs. m_{dota}. The same model playing against each other on the four different schemes, and the respective results for first and last pick.

The table further shows that when m_{dota} drafted with Split Draft, which is designed for a different game, it scored better when having the last pick. The cause might be the different total numbers of bans between Captain's Mode and Split Draft. The drafter might not ban heroes accordingly because it effectively has four bans less, which makes the last pick more important.

The next experiment for Dota 2 consists of setting itself against m_{random} for the draft schemes, both as *first pick* and *last pick*, seen in Table 4.2.

	m_{dota}	m_{random}	m_{random}	m_{dota}
Draft Scheme	**First Pick**	**Last Pick**	**First Pick**	**Last Pick**
Captain's Mode	0.802	0.198	0.147	0.853
Split Draft	0.806	0.194	0.127	0.873
Schoolyard	0.863	0.137	0.026	0.974
HoN Draft	0.733	0.267	0.147	0.853

Table 4.2: m_{dota} vs m_{random} in four different draft schemes. m_{dota} wins the draft phase dominantly over m_{random} on all four draft schemes both as first and last pick.

As expected, the trained model performed better than the random model, winning the draft phase both as first and last pick for all draft schemes.

4.2 Results for LoL

The data for LoL displays much of the same results as Dota 2. From the dataset collected in section 3.1.1, we see that the Blue side in LoL has a slight advantage over the Red side with a win rate of 50.62%, which also corresponds to the historical statistics on Blue vs. Red. In LoL, the team with first pick is set to be the Blue team, whereas, in Dota 2, the team that does not get first pick gets to choose the side.

Draft Scheme	**First Pick**	**Last Pick**
Captain's Mode	0.545	0.455
Split Draft	0.504	0.496
School Yard	0.522	0.478
HoN Draft	0.573	0.427

Table 4.3: m_{lol} vs. m_{lol}. The same model playing against each other on the four different schemes, and the respective results for first and last pick.

	m_{lol}	m_{random}	m_{random}	m_{lol}
Draft Scheme	**First Pick**	**Last Pick**	**First Pick**	**Last Pick**
Captain's Mode	0.662	0.338	0.219	0.781
Split Draft	0.887	0.113	0.173	0.827
Schoolyard	0.726	0.274	0.167	0.733
HoN Draft	0.712	0.288	0.104	0.896

Table 4.4: m_{lol} vs m_{random} in four different draft schemes. m_{lol} wins the draft phase dominantly over m_{random} on all four draft schemes both as first and last pick.

4.3 Modified Draft Scheme

In this experiment, we introduce two new draft schemes, namely *General's Mode* and *Chop Draft* which is a copy of Dota 2's and Lol's primary draft schemes, but with the last two picks flipped between the two teams. This effectively means that the team with the first pick also has the last pick, as shown in the two figures below, Figure 4.1 and Figure 4.2.

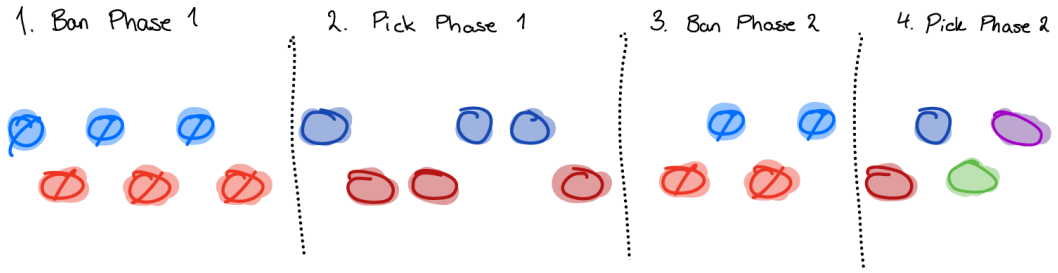

Figure 4.1: Illustration of Chop Draft. A copy of Split Draft in LoL where the 2 last picks are flipped, colored green and purple.

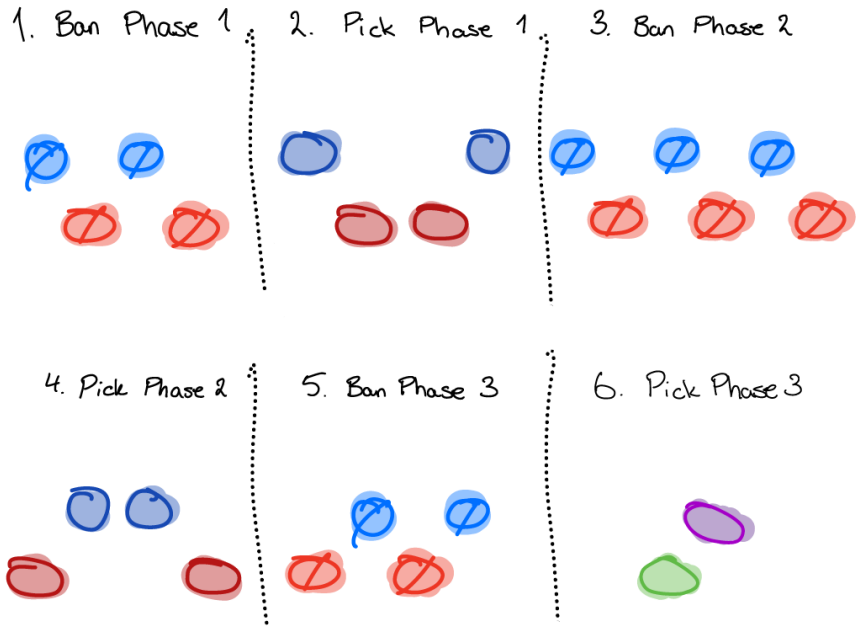

Figure 4.2: Illustration of General's Mode. A copy of Captain's Mode in Dota 2 where the 2 last picks are flipped, colored green and purple.

For this experiment, the team with both first and last pick will be denoted Team 1, and the other team will be denoted Team 2.

	Draft Scheme	**Team 1**	**Team 2**
m_{dota}	Captain's Mode	0.505	0.495
	General's Mode	0.408	0.592
m_{lol}	Split Draft	0.545	0.455
	Chop Draft	0.386	0.614

Table 4.5: Results from m_{dota} and m_{lol} in which they both played against themselves with flipped last picks.

The results from Table 4.5 show that although Team 1 has both the first and last pick, Team 2 gets a considerable advantage. The models tend to constrict themselves to a few hero compositions that it likes to draft, much like normal players. It is reasonable to believe that once the model knows the first pick, it can start building its composition around a counter for that hero. Due to the flipped picks, Team 2 gets the chance to finish this composition a step earlier while also potentially denying Team 1s last pick.

4.4 Ban Impact

The following table, Table 4.6, displays what difference the ban phase has in two otherwise identical draft schemes, Schoolyard and HoN Draft. The HoN Draft is a modification of the Schoolyard draft scheme. First, it has a ban phase at the start, then the pick phase is split into two phases and another ban phase between the pick phases, illustrated in Figure 4.3.

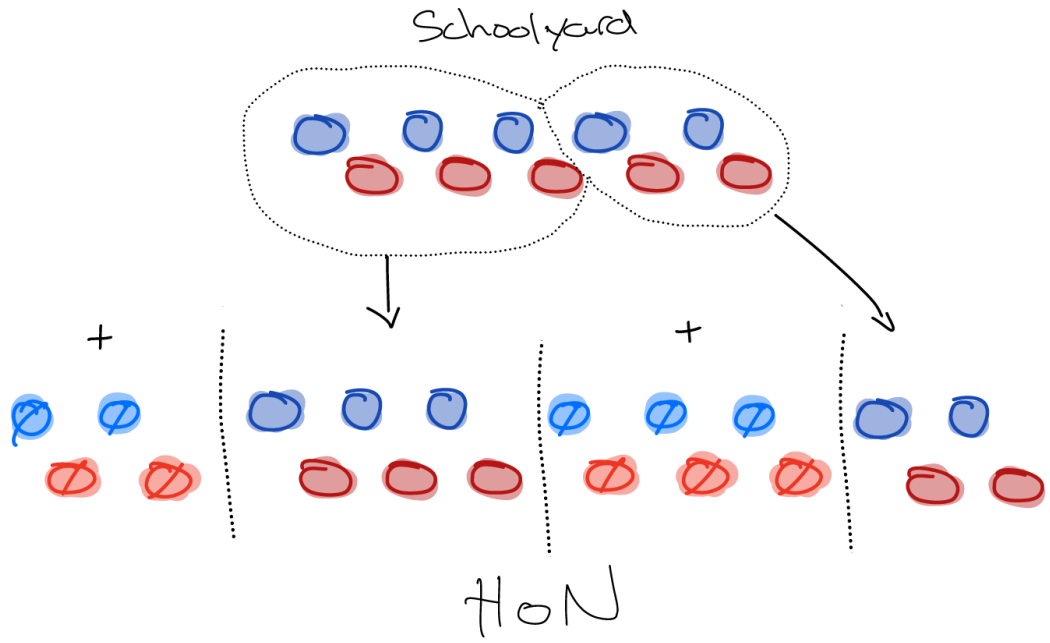

Figure 4.3: Illustration of how adding ban phases to Schoolyard to make it a HoN draft.

When adding the two ban phases we observed a 3-5% increase in win rate for both m_{dota} and m_{lol}, shown in Table 4.6.

	Draft Scheme	**First Pick**	**Last Pick**
m_{dota}	Schoolyard	0.509	0.491
	HoN Draft	0.546	0.454
m_{lol}	Schoolyard	0.522	0.478
	HoN Draft	0.573	0.427

Table 4.6: The impact two ban phases has in for both m_{lol} and m_{dota}.

The HoN draft scheme alternates between the teams, beginning with the same team in each phase. We note from previous results that the team with first pick has the advantage in almost all cases. It is, therefore, reasonable to argue that the team with first pick has the advantage coming into each phase.

From the tables presented in Section 4.5, we can argue that the respective models might use the banning phase to ban heroes that might counter their last two picks. It is also reasonable to further argue that the models prefer picking heroes that complete hero composition with heroes that synergizes well and strengthen their team over sub-optimal heroes to counter the opposing team - this gives a further advantage to Team 1 with the first pick if both teams are competing for the same heroes.

4.5 Positions by Pick Order

From the conducted results, we derived a table showing the distribution of hero positions by pick order in winning team compositions for both Dota 2, Figure 4.4 and LoL, Figure 4.5. Both games are played with their respective draft scheme.

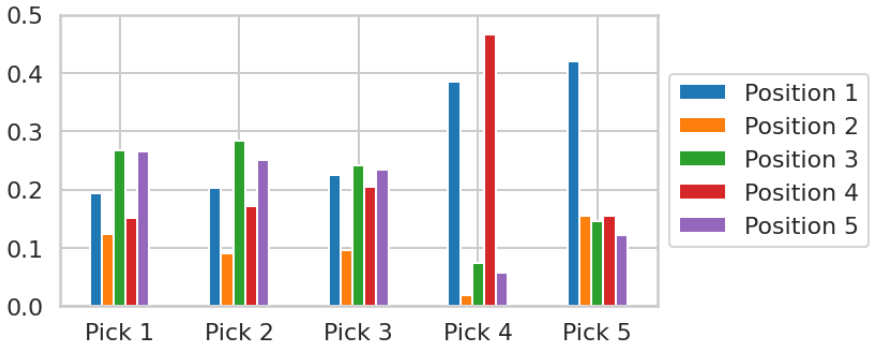

Figure 4.4: Distribution of roles over pick order for winning team composition in Dota 2 with Captain's Mode.

For Dota 2 we notice an abnormality in the second pick phase where m_{dota} favored picking heroes in *Position 1* and *Position 4*.

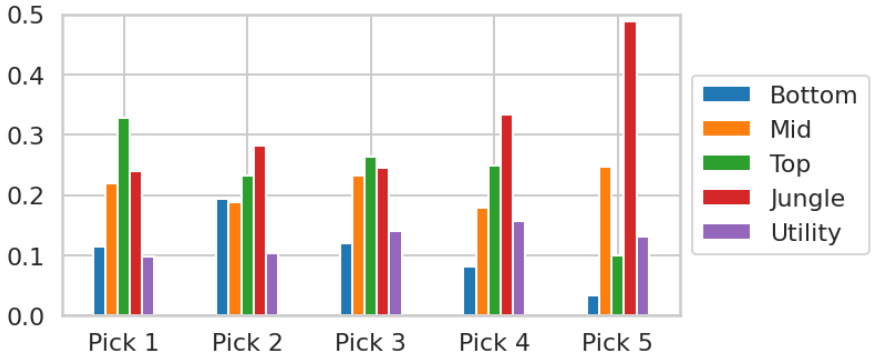

Figure 4.5: Distribution of roles over pick order for winning team composition in LoL with Split Draft.

A similar abnormality is seen for LoL where the m_{lol} favored picking heroes playing *Jungle*. Almost 50% of the last pick for m_{lol} was a hero playing in the *Jungle*. Another abnormality for m_{lol} is the first pick, in which m_{lol} favored picking a hero for *Top*. Most likely due to this position being the most flexible position in the game.

4.6 Revisiting Research Questions

In Section 1.4 we presented a set of research questions. This section will review these research questions in relation to our results.

RQ 1: Is the draft scheme for Dota 2 and LoL balanced?

Our findings presented in Section 4.1 and Section 4.2 shows a 0.5% and 0.4% advantage towards the team located in the south west corner for Dota 2 and LoL, respectively. Both of these draft schemes are tournament-only for both games, which implies that the games are using a different draft scheme as default.

Historical statistics for both games show that there is $> 1\%$ advantage towards the team located in the south west corner with the games' default draft scheme. Their tournament-only draft schemes, Captain's Mode and Split Draft, decrease this advantage to $< 1\%$.

RQ 2: How would modifying the draft scheme affect the win rate in Dota 2 and LoL?

We modified a draft scheme by flipping the two last picks such that a team had both the first and last pick. We discovered from these results that the team who had neither increased their win rate by over 10% for both games.

RQ 3: How valuable are bans in the draft phase for the two teams?

Our experiment shows that adding ban phases to a draft scheme actively increases the team's win rate with first pick by 3-5%. The win rate might change according to how these ban phases are added, to what draft schemes, and in what kind of sequence, but bans are valuable to one team or another.

RQ 4: Is there a trend in which positions get picked first or last?

Our models tend to pick certain heroes at a specific turn in the draft phase. Notably m_{dota} favored picking heroes to fill *Position 1* and *4* at pick 4 and 5, whereas m_{lol} favored picking a hero for the *Top* position at pick 1 and a hero for the *Jungle* position at either pick 4 or 5.

This confirms with the current meta for the respective games. In Dota 2 heroes playing in *Position 1* have a greater impact in the play phase compared against other positions, as for LoL it is the position *Jungle*.

Our models show a trend towards picking heroes to fill positions with the least impact first, effectively saving impact-picks for the last phase.

4.7 Limitations

As mentioned in Section 3.2.3, we limited the total simulations for the models to 300. However, we could not train the models for the desired time due to hardware limitations and time constraints.

In terms of the WRP datasets, we decided not to collect professional matches for both Dota and LoL as the sample size would be too small for use. It is worth noting that the collected matches in the datasets, mentioned in Section 3.1.1 and Section 3.1.2, use a different draft scheme, which has not been explored.

Our project was only based on Dota 2 and LoL, two well-established games in the MOBA industry, as there were difficult to gather enough data in other small MOBA games.

Chapter 5

Conclusion and Future Work

The primary objective of this thesis was to assess game balance within MOBA games, more specifically, through the draft phase. Throughout this thesis, we have studied the effects of different draft schemes applied on the two most prominent MOBA games, Dota 2 and LoL. We trained a WRP classifier as a proxy for the reward function in our MCTS models with an enriched dataset including hero positions. We set up experiments to assess game balance once our MCTS models have learned to a satisfactory level through self-play.

Even though our models never trained on an actual game with a tournament-only draft, they managed to simulate its data and learn through self-play to show that these draft schemes are more balanced than the default draft schemes both Dota 2 and LoL uses. Our results provide valuable insight into how much impact the draft phase has and how a small change to the draft scheme has on the overall game balance. More specifically, in most of the draft schemes, it is shown that the first pick is more valuable than the last pick for both games, thus also conveying the importance of draft order across various MOBA games. Additionally, the same results show a trend of which type of heroes would be picked first. Lastly, one can see how a few changes to the draft scheme can drastically change the prediction outcome. We conclude that it is possible to assess game balance by

observing only the draft phase in conjunction with these results.

5.1 Future Work

As the objective and results for this thesis was narrowed to a simple game balance assessment, we suggest the following further work:

Improve Drafter

While we got reasonable results with our models, there is always room for improvement. It would be interesting to see if adding to the complexity in the feature vector improves the model. In theory, a model trained on complex data would be able to draw complex solutions.

Improve the WRP

We think that improving the WRP will set a better foundation for the Drafter. To start off, we suggest adding bans to the dataset which the WRP will be trained on.

Scale up

We used a single thread implementation of the MCTS algorithm. We believe that by implementing a parallel MCTS algorithm would lower the simulation time substantially. Additionally, having more computational resources available would definitely speed up the overall process.

Bibliography

1. Andrade, G., Ramalho, G. L., Gomes, A. S. & Corruble, V. Dynamic Game Balancing: An Evaluation of User Satisfaction. *AIIDE* **6,** 3–8 (2006).
2. Zou, S. & Scott, D. Constraints to pickup basketball participation among Chinese American women. *Leisure Sciences* **40,** 307–325 (2018).
3. Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development* **3,** 210–229 (1959).
4. *Samuel's Checkers Player* http://www.incompleteideas.net/book/ebook/node109.html. (accessed: 12.12.2021).
5. Michie, D. & Chambers, R. A. BOXES: An experiment in adaptive control. *Machine intelligence* **2,** 137–152 (1968).
6. Tesauro, G. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation* **6,** 215–219 (1994).
7. Campbell, M., Hoane Jr, A. J. & Hsu, F.-h. Deep blue. *Artificial intelligence* **134,** 57–83 (2002).
8. DeepMind. *AlphaGo is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion, and is arguably the strongest Go player in history.* https://deepmind.com/research/case-studies/alphago-the-story-so-far. (accessed: 26.12.2021).
9. Silver, D. *et al.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362,** 1140–1144 (2018).
10. Koch, C. *How the Computer Beat the Go Master* https://www.scientificamerican.com/article/how-the-computer-beat-the-go-master/. (accessed: 12.12.2021).
11. Vinyals, O. *et al.* Starcraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).
12. Vinyals, O. *et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575,** 350–354 (2019).

13. Berner, C. *et al.* Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).

14. Kempka, M., Wydmuch, M., Runc, G., Toczek, J. & Jaśkowski, W. *Vizdoom: A doom-based ai research platform for visual reinforcement learning* in *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (2016), 1–8.

15. Allio, M. *Top 5 Most Played Gaming Genres, Ranked By Active Players* https://twinfinite.net/2018/09/most-played-gaming-genres-active-players-ranked/4/. (accessed: 14.10.2021).

16. Hassall, M. *TI10 viewership breaks records with 2.7 million peak viewers* https://esports.gg/news/dota-2/ti10-viewership-breaks-records-averages-and-peak-viewers/. (accessed: 29.11.2021).

17. Fudge, J. *Riot Games reveals Worlds 2021 Finals viewership numbers* https://www.sportsbusinessjournal.com/Esports/Sections/Media/2021/11/Worlds-2021-Finals-AMA. (accessed: 29.11.2021).

18. Baker, A. *The Most Watched Sporting Events in The World* https://www.roadtrips.com/blog/the-most-watched-sporting-events-in-the-world/. (accessed: 29.11.2021).

19. Browne, C. B. *et al.* A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* **4,** 1–43 (2012).

20. Kocsis, L. & Szepesvári, C. *Bandit based monte-carlo planning* in *European conference on machine learning* (2006), 282–293.

21. Rosin, C. D. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence* **61,** 203–230 (2011).

22. Silver, D. *et al.* Mastering the game of go without human knowledge. *nature* **550,** 354–359 (2017).

23. Chen, Z. *et al. The art of drafting: a team-oriented hero recommendation system for multiplayer online battle arena games* in *Proceedings of the 12th ACM Conference on Recommender Systems* (2018), 200–208.

24. Chen, S. *et al.* Which Heroes to Pick? Learning to Draft in MOBA Games with Neural Networks and Tree Search. *arXiv preprint arXiv:2012.10171* (2020).

25. Tomašev, N., Paquet, U., Hassabis, D. & Kramnik, V. Assessing game balance with AlphaZero: Exploring alternative rule sets in chess. *arXiv preprint arXiv:2009.04374* (2020).

26. OpenDota. *The OpenDota Blog | About* https://blog.opendota.com/2014/08/01/faq/. (accessed: 13.12.2021).

27. *OP.GG* https://op.gg/. (accessed: 24.09.2021).

28. Games, R. *Data Dragon* https://riot-api-libraries.readthedocs.io/en/latest/ddragon.html. (accessed: 24.09.2021).

29. *Riot Developer Portal* https://developer.riotgames.com/docs/portal. (accessed: 24.09.2021).

30. LeagueOfGraphs.com. *Rank distribution* https://www.leagueofgraphs.com/rankings/rank-distribution. (accessed: 15.10.2021).

31. *Champion Stats* https://www.leagueofgraphs.com/champions/stats. (accessed: 15.10.2021).

32. LeagueOfGraphs.com. *League of Graphs* https://www.leagueofgraphs.com/. (accessed: 15.10.2021).