Detection of Team Synergies based on Individual Game Playstyle for a Game with Fantastic Elements

Bc. Adam Mitrenga

Master's thesis
2023

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Adam Mitrenga**
Osobní číslo:	**A22449**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Detekce týmových synergií na základě individuálního herního stylu pro hru s fantasy prvky**
Téma práce anglicky:	**Detection of Team Synergies Based on Individual Game Playstyle for a Game with Fantasy Elements**

Zásady pro vypracování

1. Review state-of-the-art approaches for decision making and classification of individuals on multi-modal data.
2. Based on the review, select a suitable method for team synergy detection in a game with fantastic elements.
3. Implement and test the selected model with available dataset.
4. Compare the implemented solution to state-of-the-art and analyze the result.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MŮLLER, Štěpán. Detekce nevýhodných individuálních rozhodnutí pro hru s fantastickými prvky. Praha, 2022. Diplomová práce. ČVUT v Praze. Vedoucí práce Pavel Jakubec.
2. AGGARWAL, Charu C. Data Mining: The Textbook. 2015th Edition. Imprint: Springer, 2015. ISBN 978-3319141411.
3. Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C. and Józefowicz, R., 2019. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.
4. GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. Beijing: O'Reilly, [2017], xx, 545 s. ISBN 9781491962299.
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

Vedoucí diplomové práce: **Ing. Adam Viktorin, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

I hereby declare that:

- I understand that by submitting my Master's thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Master's Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Master's Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlín.
- I am aware of the fact that my Master's Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, Tomas Bata University in Zlín has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work – Master's Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlín with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Master's Thesis include the use of software provided by Tomas Bata University in Zlín or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Master's Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Master's Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- The submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlín; dated: .....

Student's Signature

ABSTRAKT

Elektronické sporty se z příležitostné zábavy staly významnou kariérní příležitostí. Tato práce představuje model, který je přizpůsoben začínajícím hráčům, aby jim pomohl odhalit nepřesnosti ve hře a zdokonalit jejich strategie. Využitím neuronové sítě vycvičené z dat z profesionálních zápasů byla zvýšena schopnost předvídat nadcházející akce hráčů. Integrace prvku "fog of war" pomáhá usnadnit vyhodnocování nesrovnalostí mezi předpokládanými a skutečnými akcemi a upozorňuje na potenciální oblasti pro zlepšení hry.

Klíčová slova: neuronové sítě, umělá inteligence ve hrách, umělá inteligence v multiplayerových hrách, mlha války

ABSTRACT

E-sports has evolved from casual entertainment to significant career opportunities. This thesis presents a model tailored to assist novice players in pinpointing gameplay inaccuracies and refining their strategies. By harnessing a neural network trained from professional match data, the ability to predict impending player actions has been enhanced. Integrating the "fog of war" feature helps to facilitate the assessment of discrepancies between anticipated and actual actions, highlighting potential areas for gameplay improvement.

Keywords: neural networks, artificial intelligence in games, artificial intelligence in multiplayer games, fog of war

First and foremost, I would like to express my profound gratitude to my supervisors. Their invaluable insights, patient guidance, and consistent support were instrumental in addressing my queries and directing my research in the appropriate direction. Their dedication and expertise have played an essential role in shaping this thesis.

I would also like to extend my heartfelt thanks to my family, whose unwavering belief in me and continuous encouragement provided the emotional and moral support needed during the challenging phases of this endeavour.

TABLE OF CONTENTS

INTRODUCTION

E-sports games have undergone a significant evolution, transitioning from leisure activities to established career opportunities. This thesis presents a model designed to guide less experienced players in recognizing their gameplay errors and improving their strategies. The discussion begins with a theoretical foundation, retracing the origins of player vs. player games and the incorporation of Artificial Intelligence (AI). The narrative delves deep into the world of multiplayer games, especially focusing on the MOBA (Multiplayer Online Battle Arena) genre. An in-depth analysis of game mechanics follows, with a concentrated spotlight on "League of Legends". Demonstrating the diversity of e-sports, the discussion further expands to include other notable games like CS:GO, DOTA2, and the F1 2022 racing game. This theoretical foundation is essential to understand the intricate history and progression of AI applications in competitive gaming.

The subsequent practical section offers a more granular examination of the research components. It presents a comprehensive overview of the datasets, emphasizing the strategies employed in their refinement and management. Special attention is directed towards the innovative approach adopted to incorporate the Fog of War feature. The study's primary model is meticulously outlined, setting the stage for a comparative analysis of its iterations, both with and without the Fog of War. In the end, expert opinions on these findings have been gathered from professional player.

I. THEORETHICAL PART

1 INTRODUCTION TO GAMES

This section will discuss the differences between games where players compete against one another or a computer versus games where multiple players have to cooperate and compete with each other.

1.1 Player vs. Player (1v1) games

AI (Artificial Intelligence) used to find it difficult to play games player vs. player like Chess and Go due to the complex rules, strategies, and decision-making involved. Playing these games can be considered a complicated planning problem, requiring skills like pattern recognition, predicting opponent moves, and making decisions under uncertainty. But with the development of machine learning and deep neural networks, AI systems such as AlphaZero [1] or Stockfish [2] have made significant progress in playing these games and even outperforming humans consistently on the highest level [3].

1.1.1 Chess

For a considerable time, humans have been trying to develop AI that can surpass them in the game of chess. In 1985, a chess computer called "Deep Thought" was developed by IBM. It competed at the world championship level and became the first computer to defeat a grandmaster in a tournament game [4]. However, it was defeated by world chess champion Garry Kasparov in a match in 1989. Continuing research led to the design of a new supercomputer, "Deep Blue", which competed against Kasparov in 1996. Kasparov won the match, but Deep Blue became the first computer to win a game against a reigning world champion in a competition. In a rematch in 1997, Deep Blue defeated Kasparov in a six-game match, becoming the first computer to defeat a world champion in a game under tournament conditions [5]. After Deep Blue's success, chess engines' development continued, and one of the most notable ones is Stockfish. As an open-source engine, it is currently developed by the whole community of open-source developers. It is highly customizable and has gained widespread popularity among chess players. It has competed successfully in many tournaments and has been used to analyze numerous games by chess players and analysts. A conventional alpha-beta tree search and a neural net evaluation are combined in Stockfish NNUE [2].

In recent years, AlphaZero has become one of the most significant breakthroughs in

the development of AI for playing chess. AlphaZero uses a neural network to learn from its own moves and become better at games. Using a unique approach to machine learning called reinforcement learning. It generates a list of moves from its neural network and then uses the Monte Carlo Tree Search algorithm to explore the best ones for each game. AlphaZero defeated the world champion chess engine Stockfish in a highly publicized match [1].

1.1.2 Go

Go is a board game that originated in China over 2,500 years ago. It is believed to be the oldest board game that is still played today. The game is played by two players who take turns placing black and white stones on a wooden board with a 19 by 19 grid of lines, thus having even more possible combinations than chess. One of many challenges in creating AI for go was the large number of possible moves, which made it extremely difficult to evaluate the board position and choose the current best move, yet another challenge a top of that compared to, for example, chess, was the way the game is conceptualized where in chess player wins only by check-mating the opponents' king. However, in GO, participants win by controlling more territory than the opponent, making it hard to create an evaluation function that accurately reflects the game's current state.

Before deep neural networks and reinforcement learning techniques emerged, early AI programs designed for playing Go relied on different approaches to simulate human expertise. One such system involved incorporating human-made rules or patterns to guide the AI's decision-making process. For instance, Arthur Samuel's program, developed in 1959, used a scoring function considering liberties, territory, and influence features. However, these early programs often performed badly. They made errors easily avoidable by human players, primarily due to their inability to capture the intricacy and complexity inherent in the game of Go. An alternative approach to developing AI for playing Go was to leverage pattern recognition techniques to identify recurrent shapes and situations on the board. The Many Faces of Go, created by David Fotland in 1988, was one of the first programs to adopt this method. Utilizing a vast database of game records, the program identified common patterns and incorporated them into its decision-making process. Nonetheless, like other rule-based techniques, The Many Faces of Go had certain limitations in comprehending the nuances and intricacies inherent in the game of Go.

From the rule-based approaches, Monte Carlo techniques were developed to simulate

numerous potential games from a particular position and approximate the likelihood of victory for each move. In 2003, Bruno Bouzy pioneered the application of Monte Carlo methods to Go, with Rémi Coulom subsequently improving them [6]. These Monte Carlo techniques did not necessitate explicit evaluation functions or domain knowledge, rendering them more malleable and versatile. While these programs surpassed previous rule-based or pattern-based programs, they were still limited in terms of accuracy and efficiency [7].

The breakthrough in the field of AI for Go came in 2016 when Google's DeepMind team unveiled AlphaGo, a deep neural network-based program that combined supervised and reinforcement learning. AlphaGo used a combination of Monte Carlo Tree Search and neural networks to evaluate and choose moves, ultimately defeating Lee Sedol, one of the world's top Go players[8].

1.2 Multiplayer games

Multiplayer games have presented a significant challenge for AI to master due to the dynamic and unpredictable nature of interactions between players. Compared to single-player games, multiplayer games require AI to be flexible and responsive to a constantly changing environment where decisions are based on incomplete information. Thus, participating in multiplayer games poses a highly intricate planning challenge for AI. It demands competencies in pattern recognition, decision-making under uncertainty, and forecasting the movements and actions of opponents and teammates.

Thanks to the advancement of ML and deep neural networks, significant progress has been made in developing AI systems that are capable of grasping multiplayer games. This progress has been displayed by AI systems such as OpenAI's DOTA 2 bot and DeepMind's AlphaStar, which have achieved exceptional success in beating professional players in high-stakes matches [9].

Methods used for achieving this were a mix of supervised and reinforcement learning techniques to learn the game's mechanics, strategies, and decision-making processes by playing against other AI or human players. They also used advanced techniques such as computer vision so the AI can understand what is happening at any given time and natural language processing to understand the game state and communicate with teammates or opponents. This demonstrated the capability of AI to learn and adapt to complex environments and situations, highlighting the power and potential of AI as a tool that could be translated for solving real-world problems and very complex

games [9].

1.2.1 League of legends

League of Legends (LoL) is a popular multiplayer online battle arena (MOBA) video game developed and published by Riot Games. The game is played in real-time. The game's complexity comes from the numerous strategies and combinations of champions, items, and objectives that can be used to gain an advantage over the enemy team. Players must also take into account the positioning and movements of their opponents to make informed decisions on when to engage in fights or objectives.

AI systems that aim to play League of Legends would have to recognise the different champions and their abilities, as well as understand the game's various objectives and strategies. Additionally, the AI would have to be able to predict and respond to the actions of other players in real-time.

The game offers several modes, including ARAM (All Random All Mid) and One for All. However, Summoners Rift stands out as the most favored mode. It is on this mode that the focus will be directed, where players begin by choosing their champion from the available pool. As of 23 March 2023, there are currently 163 released champions. Riot games are adding 4-8 new champions per year[10].

Summoners Rift overview It is a five versus-five multiplayer mode where players combat each other on a square map that is diagonally mirrored. The standard assignment is one player in the jungle, one in the top lane, one in the mid lane, and two in the bottom lane, with designated roles as marksmen and support. This can be seen in Figure 1.1. The jungle player roams the map, taking down neutral monsters to gain gold and experience and assisting their teammates in lanes when needed.

While this is the traditional setup, there have been instances where players have deviated from it and experimented with different strategies. Successful strategies have been developed where players played differently. However, the game's developer, Riot Games, has quickly addressed these strategies and often nerfs them in subsequent patches to keep the standard lane assignments.

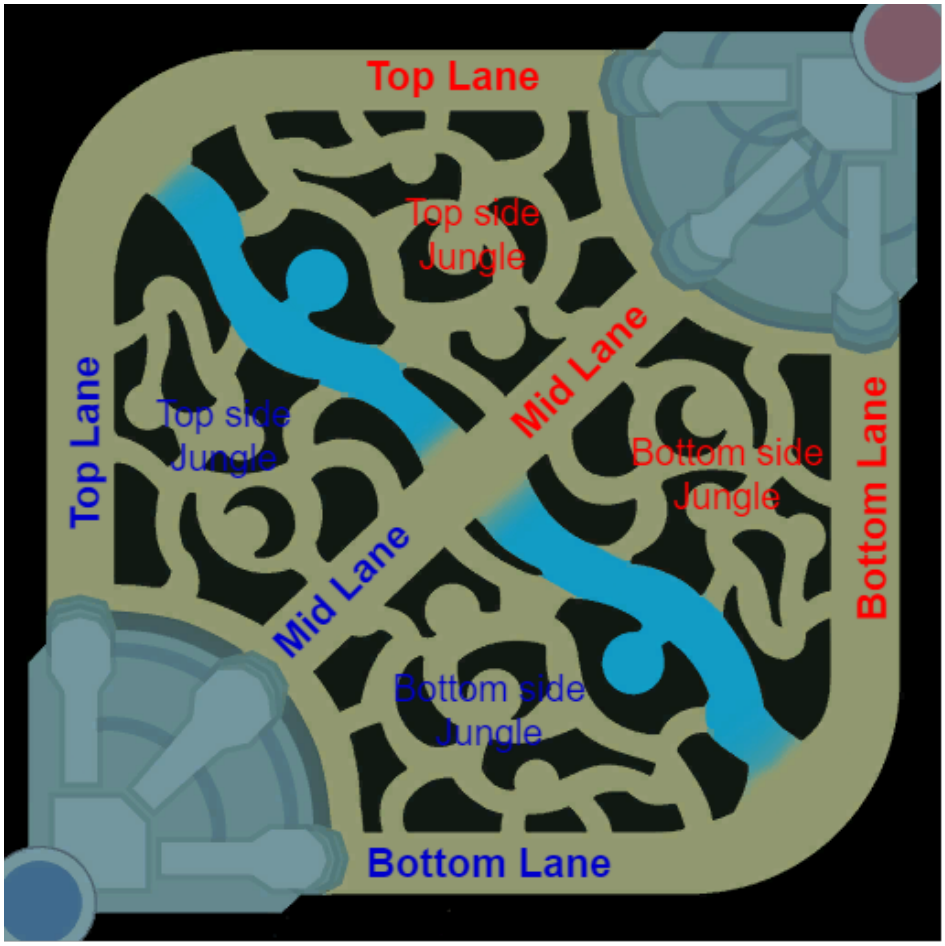

Figure 1.1 Diagram of Summoners rift map with depicted lanes

Despite this, there is still some flexibility in champion selection for each role. Players can choose unique champions to play in each role that were not specifically designed for that position. However, the number of players in each lane and the jungle remains fixed in the latest patches.

Pick-ban phase Before players can begin to play the game, they must go through a pick-ban phase. In this phase, each player selects their champion for their assigned role, and five champions are banned from being played in the game.

In the banning phase, each team bans five champions from being played in the game. The champions that are banned are the ones that are considered overpowered or that the opposing team has a particular weakness against. Once the banning phase is complete, players move on to the picking phase. In professional conditions, a strategy is known as target banning, where players ban champions, they know their known enemy excels at, even if the champion is not regarded as strong in the current meta.

Once the banning phase is complete, players move on to the picking phase.

During the picking phase, players choose their champions, starting with the team with the first pick. After that order of picks and bans alternates between the two teams. If one team gets the first pick, the other gets the last two, which can be advantageous in certain situations.

In this phase, players select their champions based on their assigned roles, such as top lane, mid lane, or support. Each player can only select one champion, and once all players have chosen their champions, the game begins.

If a certain champion is desirable, players don't have to pick for themselves but can pick for one of their teammates and swap champions at the end of the pick-ban phase without consequences.

Early game After the pick-ban phase in LoL, players move on to the game itself. The game starts with a brief period where players can buy items from the shop using gold whilst not being able to leave the respawn zone. Each player starts with a small amount of gold. Additional gold can be earned by killing enemy minions, neutral monsters, enemy champions or killing enemy buildings. Furthermore, after minions start spawning (1:30), a small amount of gold is earned passively for each passing second.

Items are equipment that players can purchase from the in-game shop to improve their champion's stats and abilities. Items can only be purchased in respawn zone or when the champion is respawning (dead). Many different types of items are available in the game, each with its unique set of stats and effects. For example, some items might provide increased health or mana, while others might increase the damage dealt by a champion's abilities. Other items might increase movement speed or provide crowd control effects such as slowing down or stunning enemy champions.

After the short buying period, the game begins, and players can leave their team's respawn zone and move out onto the map. The minions start to spawn. Minions are small computer-controlled units that spawn in static intervals of 30 seconds [11] and automatically moves down each lane towards the enemy base. Each team has its own minions, and they play an essential role in pushing towards the enemy base. Additionally, champions get gold for landing the killing blow on an enemy minion or monster to gain gold and experience. This technique is known as the last hitting.

Champions also get experience just for being nearby when enemy minions die. Minions come in three types: melee, caster, and siege. Melee minions are strong but must get close to enemy champions to attack them. Caster minions have ranged attacks, making them useful for poking at enemy champions from a distance. Siege minions are more durable than the other types, deal significantly more damage to turrets and inhibitors and take less damage from these structures.
Each player moves to their assigned lane and the **early game** begins. The early game is generally considered to be the first 10-15 minutes of the game. During this phase, players typically try to establish a strong presence in the lane, focusing on gaining experience and gold by killing minions, neutral monsters, buffs and early tower takedowns. These objectives can be seen in Figure 1.3. As players progress through the game, they earn experience points and level up, gaining new abilities and increasing their champion's stats. Each and every champion is unique, with their own skills and different playstyles. Some champions are stronger during the early phases of the game thus, trying to play more aggressively and roaming on the map can be a good strategy for players, but usually, players stay bound to their assigned lanes as seen in Figure 1.1 due to the possible loss of experience and gold. During the early game, champions gain between 125-147 golds [11]. So any potential roam from laners is usually a trade-off from experience and gold loss if it doesn't lead to a successful kill or objective takedown.

Mid game The game transitions into the **mid game** as the players go through time. This phase typically occurs around 15-30 minutes. During the mid-game, players roam more freely around the map, grouping with teammates to secure objectives and engage in team fights. Objectives such as turrets, dragons, and Rift Herald/Baron Nashor become more contested as they provide significant advantages to the team that secures them. Turrets provide gold, map control, and vision, while dragons grant various team-wide buffs depending on the dragon type.

Vision control becomes increasingly more important during the mid game, as securing objectives often requires proper vision setup and denial. Players can use wards (trinkets or items) to gain vision over specific areas of the map and can use a sweeper trinket or control wards to deny enemy vision. This allows teams to spot enemy movements, avoid ambushes, and create opportunities for surprise attacks or objective steals. Team fights become more frequent during the mid game, as teams often group to contest objectives or push lanes. In a team fight, proper positioning, target prioritization, and crowd control abilities can be crucial in determining the outcome.

Late game As the game moves into the **late game**, typically around 30 minutes, champions reach their full potential with a full or nearly full set of items and abilities. The stakes are higher, as death timers are longer, and any mistake can significantly impact the game's outcome. During the late game, the main focus is securing key objectives, such as Baron Nashor, Elder Dragon, and inhibitors. Same As in mid-game, vision plays a key role in the late game as having information about the enemy's position or the lack of it can determine how well each team can make the decision. The game ends when one team destroys the enemy Nexus, a large structure located at the centre of each team's base. The Nexus is protected by two Nexus turrets, which must be destroyed before the Nexus can be damaged. Destroying the Nexus results in victory for the attacking team and ends the game.

Buffs Players can enhance their champions through various means, like purchasing items with gold, gaining experience, and acquiring both temporary and permanent buffs from neutral objectives. As mentioned earlier, four primary buffs can be obtained from neutral objectives: Blue buff, Red buff, Baron buff, and Dragon buff.

- **Blue buff (Crest of Insight):** This buff is acquired by defeating the Blue Sentinel, a neutral monster in the jungles' blue and red sides. The Blue buff grants the champion increased mana or energy regeneration and cooldown reduction for 120 seconds or until the death of the champion, whichever comes first. This buff is particularly valuable for champions that rely heavily on their abilities and have high mana costs [12].

  This buff is typically intended for the jungler to take in the early game. However, as the mid-game approaches, it is often passed on to the mid laner. This is because the enhanced spell regeneration the buff provides allows the mid laner to control the centre of the map better.

- **Red buff (Crest of Cinders)**: This buff is obtained by defeating the Red Brambleback, a neutral monster found again on both teams' sides of the jungle. The Red buff grants the champion a slow effect on their basic attacks and bonus true damage over time for 120 seconds or until death, and it also provides health regeneration [13].

  Similar to blue buff, it is typically intended for the jungler to take in the early game since the extra damage and sustain will help him go through early fights in the jungle, but in mid game usually marksman or another long-range champion that can utilise the on-hit effect of this buff will be the receiver of this buff

- **Rift Herald**: Rift Herald is a neutral monster that spawns on the top side of the river, in the Baron Nashor pit, at the 8-minute. And despawns at 19:45 (or 19:55 if in combat) to make way for Baron Nashor, described later. When defeated, it drops the "Eye of the Herald," an item that replaces the trinket for the champion that picks it up [14].

  The Eye of the Herald can be used within 4 minutes after picking it up to summon the Rift Herald on the player's side. When summoned, Rift Herald pushes the nearest lane, charging and dealing substantial damage to enemy objectives. The Rift Herald primarily aims to help the team take down turrets and gain map control. It can be instrumental in accelerating the game's pace, breaking open lanes, or creating pressure on the map while contesting other objectives on the opposite side of the map.

- **Baron buff (Hand of Baron)**: This arguably the most buff is obtained by defeating Baron Nashor, a neutral monster that spawns at 20 minutes. The Baron buff grants increased attack damage, ability power, and an empowered recall for the entire team. Additionally, when players are nearby friendly minions with Hands of Baron, it makes them more durable and empowers their stats. The buff lasts for 180 seconds or until death. This is one of the most powerful buffs in the game, and when one team takes him, it is coined as Baron Power play in the professional scene [15].

- **Dragon Buffs** The game has six Elemental Drakes, each providing different permanent buffs for teams. Drakes first appear in the Dragon Pit at the 5-minute mark. The initial dragon type is selected randomly from a pool of six. The second dragon is chosen from the remaining five types, ensuring that the same type won't spawn consecutively.

  Once the second dragon is defeated, the third dragon is chosen in a similar way. However, from this point forward, the dragon type remains the same for the rest of the game, and the map changes accordingly to the dragon type.

  When a team successfully defeats four dragons, an Elder Dragon will spawn

  1. **Chemtech Drake** When a team slays a Chemtech Drake(CD), it grants each member a stacking buff called "Chemtech Might." This buff increases Ability Power (AP) and Attack Damage (AD) by a flat amount and also grants a percentage increase in Ability Haste [16].

     When a team slays four Drakes, they will acquire an additional powerful buff called "Chemtech Soul." This unique effect grants the team the "Chemtech Surge" buff when they are participating in a kill. The Chemtech Surge buff

grants bonus movement speed and resets the cooldown of a champion's basic abilities (excluding the ultimate ability) by a percentage[16].

2. **Cloud Drake** grants each member a stacking buff called "Cloudbringer's Grace." This buff provides a percentage increase in ability Haste for their ultimate ability, allowing champions to use their ultimate abilities more frequently [17].

   When a team slays four Drakes, they gain an additional buff called "Cloud Soul." This unique effect grants the team a significant movement speed bonus for a few seconds after casting their ultimate ability. This buff enhances a champion's mobility during fights and allows for better positioning and repositioning[17].

3. **Hextech Drake** grants each member a stacking buff called "Hextech Might." This buff increases the champion's adaptive force (a combination of AD and AP, depending on the champion's already gained stat of that kind) by a flat amount [18].

   When a team slays four Drake, they gain an additional buff called "Hextech Soul." This unique effect grants the team a chain lightning effect on their basic attacks and abilities, which can bounce to additional nearby enemy targets, dealing magic damage. The chain lightning effect has a short cooldown per individual champion[18].

4. **Infernal Drake** grants each member a stacking buff called "Infernal Might." This buff provides a percentage increase in attack damage and ability power, making champions deal additional damage with their abilities and basic attacks [19].

   When a team slays four Drakes, they gain an additional buff named "Infernal Soul." This unique effect grants the team an explosion on their basic attacks and abilities that deal adaptive damage to enemies in an area. The eruption has a short cooldown per champion[19].

5. **Mountain Drake** Its unique stacking buff, Mountainous Vigor, provides each team member a percentage gain in armour and magic resistance. This makes champions more durable and able to withstand incoming damage [20].

   After slaying four Drakes, the team obtains the Mountain Soul buff. This powerful buff grants the team a shield that regenerates after a short period of not taking damage. This shield is precious because its strength is based on the champion's maximum health, which is especially beneficial for tank champions with much health[20].

6. **Ocean Drake** Killing an Ocean Drake gives each team member a stacking "Oceanic Will" buff that increases health regeneration based on missing health[21].

   Killing all four Elemental Drakes grants the team the "Ocean Soul" buff, providing additional health and resource regeneration when dealing damage to champions or structures, enhancing champion sustain during fights [21].

- **Elder Dragon**: upon either of the two teams playing slays their fourth Elemental Drake, The Elder Dragon appears in the dragon pit. Unlike the Elemental Drakes, the Elder Dragon has a 6-minute respawn timer and does not grant a stacking buff. Instead, it gives a powerful temporary buff "Elder Immolation, similar to "Hand of Baron", to the team that defeats it[22]. The Elder Immolation buff lasts for up to 2 minutes and grants:
  - Boosted damage to champions and structures through a percentage increase in attack damage and ability power[23].
  - An execute effect that activates when enemy champions' health falls below 20% of their maximum health, instantly destroying them[23].

Fixed Objectives in LoL, each team has several fixed objectives within their base and across the lanes. These objectives are essential to the game's progress and are determinal for victory. The main static objectives for each team consist of the following:

- **Nexus** is the primary objective in the game; it is located at the centre of each team's base. The main goal for both teams is to destroy the enemy Nexus. When a Nexus is destroyed, the game ends, and the team that destroyed the enemy Nexus is declared the winner. The Nexus cannot be directly attacked from the beginning of the game. To reach and damage the Nexus, a team must first destroy the outer, inner, and inhibitor turrets in at least one lane and the inhibitor in that lane. Only after destroying these structures can a team attack the enemy Nexus [24].
- **Turrets** are defensive structures placed along all three lanes and in front of the Nexus. Turrets deal damage to enemy champions and minions that come within their range. Turrets' target priority is very simple: first attacking neutral monsters spawned by champions, then minions, and lastly, the champions. If there are no minions in range or an enemy champion attacks an allied champion within turret range, it will prioritize attacking the champion that attacked the

enemy champion. Their damage increases with each consecutive hit on the same target. In recent years the concept of *Turret plating* has been introduced to the game to defeat strategies that would force push the first towers way too early in the game. During the early game, outer turrets have a protective mechanism. *Turret plating* consists of five plates, each granting additional health and gold when destroyed. *Turret plating* falls off at the 14-minute mark, making the outer turrets more vulnerable. Turrets also have backdoor protection to discourage champions from bypassing minion waves and attacking turrets directly. This protection grants turrets bonus armour and magic resistance when no enemy minions are within turret range [25]. The turrets in League of Legends are divided into four types based on their location. The first type is the **Outer Turrets**, also known as tier 1 turrets. These turrets are unique compared to others because they have *Turret Plating*, which provides additional defence for a short period of time. The second type is the **Inner Turrets**, or tier 2 turrets, which are located closer to the centre of the map. The third type is the **Base Turrets**, also known as textbfInhibitor Turrets, which are located in each team's base and have a special ability to regenerate 3 health per second after being damaged. Finally, the **Nexus Turrets** are located near each team's nexus and are the most difficult turrets to destroy. They have the ability to regenerate health at a doubled rate of 6 health per second if they have been damaged[25].

- **Inhibitors** are located within a team's base in each of the three lanes. They are positioned behind the base turrets (inhibitor turrets). When an inhibitor is destroyed, the team that destroyed it will begin spawning super minions in the respective lane. Super minions have higher health, deal more damage, and are harder to kill than regular minions. Thus they significantly aid in pushing the lane towards the enemy base. Inhibitors are not permanently destroyed. They respawn after 5 minutes, and once they do, super minions will no longer spawn in the respective lane [26].

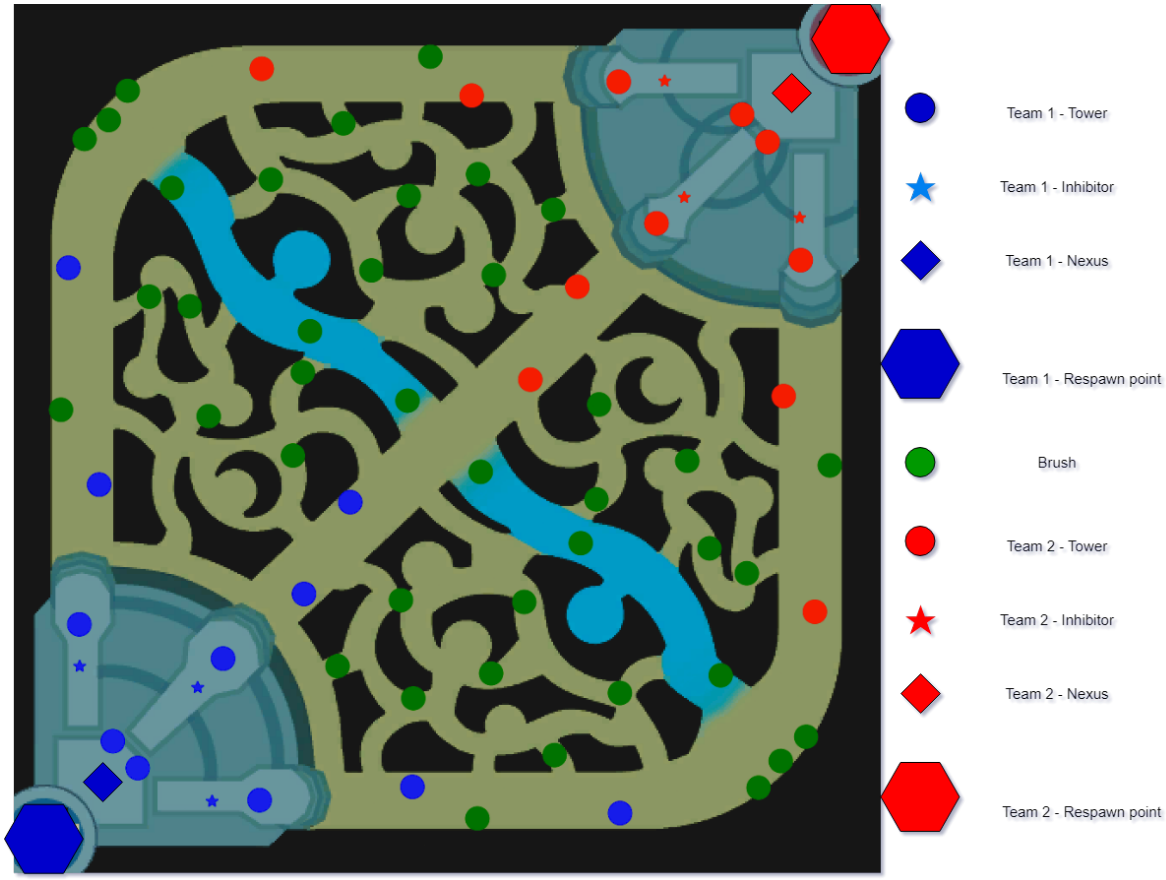

Figure 1.2 Diagram of Summoners rift map with buildings and brushes

Figure 1.3 Diagram of Summoners rift map with neutral monsters

1.2.2 DOTA 2

Defense of the Ancients 2 is a widely-played multiplayer online battle arena (MOBA) video game developed and published by Valve Corporation. The game unfolds in real-time, and its complexity originates from the multitude of strategies and combinations of heroes, items, and objectives that can be used to establish superiority over the enemy team. Players in this game must consider the positioning and movements of their opponents to make informed decisions on when to initiate fights or objectives.

Since DOTA is a predecessor of LOL, they have very similar game styles and game phases.

AI systems that aspire to play DOTA 2 must be proficient at identifying the different

heroes and their abilities and comprehending the game's various objectives and strategies. Furthermore, the AI would have to be capable of predicting and reacting to the actions of other players in real-time.

The game features various game modes, such as All Pick, Single Draft, and All Random, but the most commonly played game mode is All Pick, particularly Ranked All Pick. As of 23 March 2023, there are currently 124 released heroes. Valve Corporation typically adds 1-3 new heroes per year. [27]

Ranked All Pick is the most common game mode for ranked matches. In this mode, all heroes are available for selection. Players may pick any hero as long as it hasn't been chosen by another player. The process includes a ban phase before the picking phase, which is described in detail.

Pick-Ban Phase After selecting Ranked all Pick and finding opponents through matchmaking the champion selection begins.

Phase 1 - Bane Phase:

- All participants are given a time span of 15 seconds to nominate a hero for banning.
- Each hero can only be nominated once. No two players can nominate the same hero.
- Nominations are publicly displayed, but the identity of the nominating player remains undisclosed.
- Post the 15-second window, every nominated hero has an independent 50% probability of getting banned.
- The game system automatically implements additional bans on random heroes based on their ban rates within the particular MMR (Matchmaking Rating) bracket of the ongoing match, ensuring a total of 16 heroes are banned [28].

Phase 2 - Pick Phase:

- The hero selection process is conducted over three stages.

- The initial two stages allot 25 seconds each, during which two players from each team make their choice of hero.
- The final stage lasts for 20 seconds, where one remaining player from each team makes a selection.
- If the countdown expires, players yet to choose their heroes start losing 2 gold per second.
- Hero selections are kept concealed until the end of each round. If a hero is simultaneously chosen by two players, that hero gets banned, and the round recommences. This scenario can occur twice at most. If such a situation arises for the third time, the player who first chose the hero retains it, while the other player is granted additional time for a fresh selection.
- Players who fail to make a selection and continue to lose gold for over 30 seconds are automatically assigned a random hero.
- If a player doesn't enter the battlefield with their selected hero until the 0:00 mark, their gold is confiscated and evenly distributed among their team members. This is similar to the player abandoning the game, but it does not lead to the game being classified as safe to leave [28].

Interestingly, if a hero is locked in during phase 1, it can't be picked by the opposing team in the later phases to ban it. Additionally, if both teams pick the same hero, it will be banned, and the pick phase counter for that round resets. This could happen twice. During the third time, whoever picks first gets the hero, and the other player gets more time to pick a different hero [28].

Game Phases After pick ban phase the game begins. **The Early game** that sets the foundation for all the action that follows. It starts as soon as the game timer starts and typically lasts for about 10-15 minutes, though the exact duration can vary based on the specific strategies used by each team. The decisions and actions made during this stage can significantly affect the game's trajectory, setting the tone for the mid and late-game.[29]

The map in DOTA 2 is divided into three lanes: top, middle, and bottom, which can be seen in Figure 1.4. Each lane is guarded by defensive structures called towers (red and green squares), and it is along these lanes that the majority of the early game combat

takes place. These lanes are the battlegrounds where the heroes face off against each other with the assistance of AI-controlled units known as creeps [29].

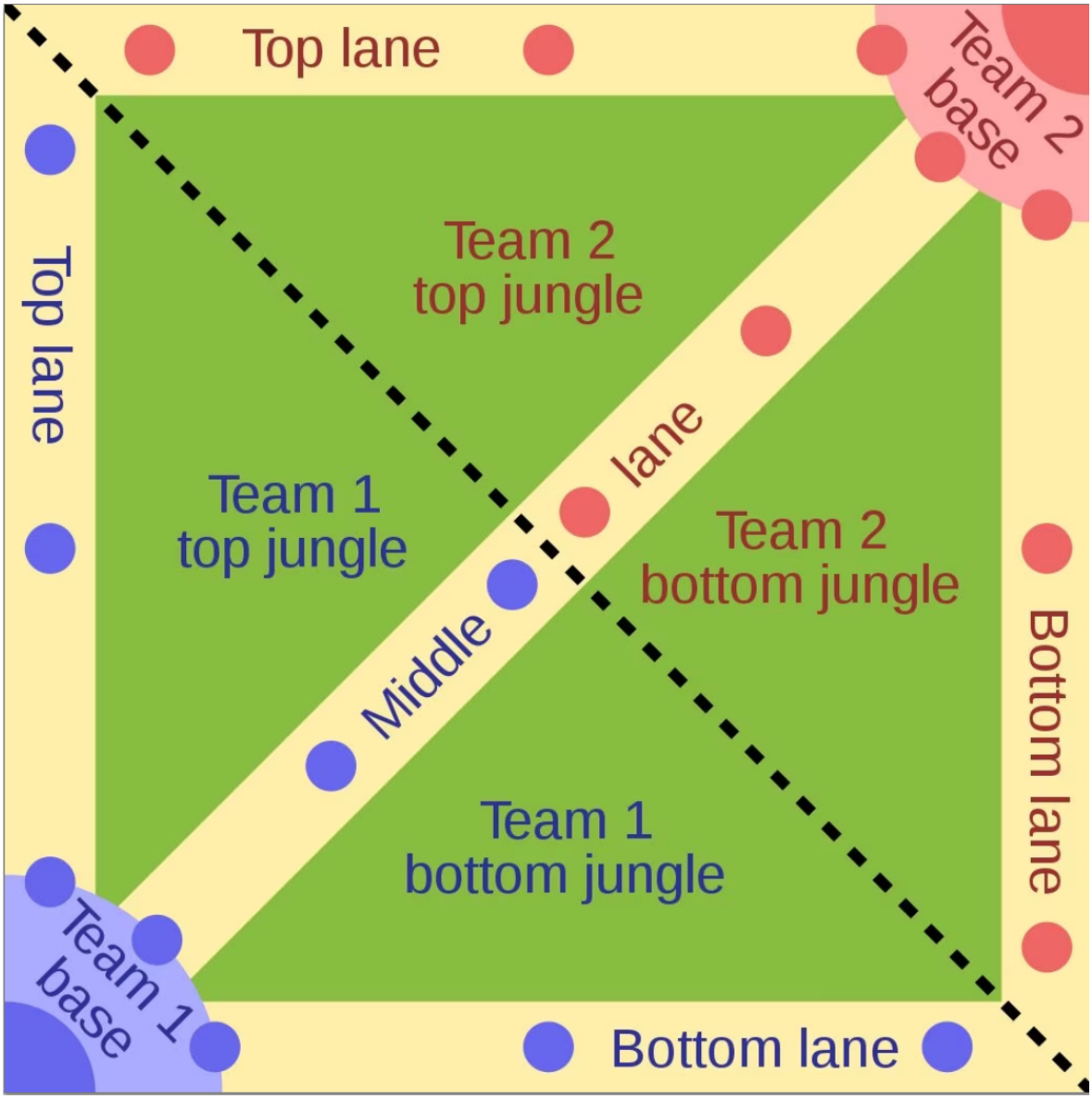

Figure 1.4 DOTA map with depicted lanes [30]

The creeps spawn from both bases regularly, clashing in the middle of the lanes. During the laning stage, the primary objective for players is to accumulate as much experience and gold as possible. Experience points are gained by being in the vicinity when the enemy creeps, or heroes die. Gold is earned by landing the killing blow on these units. What is uniquely different from LOL is that you can deny the last hitting by last hitting your own creeps. The laning stage is also a period of intense player-versus-player combat. Players must balance their need to farm creeps with the necessity to harass and potentially kill enemy heroes. Successfully killing an enemy hero not only awards a substantial amount of experience and gold but also temporarily removes that player

from the lane, allowing for uncontested farming and potentially damaging the enemy's structures. During this phase, supports might rotate between lanes to help secure kills or relieve struggling teammates, a tactic known as 'ganking.' Vision control is also crucial at this stage. Teams strategically place observer wards to monitor potential enemy movements and control crucial areas. Another critical aspect of the laning stage is understanding the power spikes of the hero players are playing. Certain heroes are strong in the early game and can be used to exert pressure on the enemy from the beginning. Understanding hero matchups and making successful rotations can make the difference between winning or losing the lane and, potentially, the entire game. It's the first domino that could fall in a chain of events that leads to the destruction of the enemy's ancient (base) [29].

The Mid Game generally begins when heroes roam and group to achieve objectives, moving beyond the initial laning phase. Farming remains a key priority during the mid-game for mid or bot players. Maintaining a balance between fighting and farming is essential to ensure steady gold and experience growth. Effective farming patterns become crucial as they allow players to accumulate resources while staying relatively safe. Players try to achieve power spikes, such as unlocking an essential item or ability, using this to their advantage in skirmishes or team fights. For example, some heroes become significantly stronger after obtaining certain items or reaching a particular level, which can be used to take down objectives or win team fights [31][32].

As in any MOBA game, claiming objective is crucial during mid-game. Recognizing when to take towers, Roshan, or other objectives can bring advantages that help win the game which can be seen in Figure 1.5. Applying pressure by pushing lanes and seizing objectives can force the enemy to respond, often leading to good fights. Keeping control of the map through vision is crucial as well. As the game progresses, more fights happen, and the map becomes more dangerous with less vision. It's essential to ward strategically, providing vision in key areas to spot enemy movements and set up ambushes. It's also important to deny the enemy team's vision by de-warding their wards [31][32].

Figure 1.5 DOTA map with objectives depicted

The mid-game phase is fluid and complex, requiring adaptability and strategic decision-making. Identifying threats, deciding when to fight, farming effectively, and controlling the map is pivotal to gaining an advantage and transitioning smoothly into the late game.

The Late Game typically begins around the 30-minute mark, but this can vary depending on how the match unfolds. In this stage, heroes reach their maximum levels and obtain powerful items and the focus shifts towards team fights and objectives. One aspect of the late game that's critical is positioning. Teams must stick together, as being caught out alone can lead to a quick death and give the enemy an advantage. Vision also becomes increasingly important, so warding key areas of the map to keep

tabs on the enemy's movement is crucial.

Objective control becomes a primary focus, particularly Roshan visible in Figure 1.5 and major neutral objectives. Roshan not only gives a large amount of gold and experience but also drops the Aegis of the Immortal, which allows the hero who carries it to respawn upon death, giving a significant advantage in upcoming team fights. Champions that scale well into the late game become particularly valuable. These heroes often depend on high amounts of a farm (gold and experience) to reach their full potential but can carry their team to victory once they reach their peak [33].

Finally, communication and coordination become increasingly crucial in the late game. Teams must work together to push lanes, secure objectives, and win fights. A well-coordinated team can often overcome a gold or experience deficit and win the game [33].

Additional modes in DOTA 2 DOTA 2 offers a wide variety of game modes, featuring innovative modes that support 1v1 gameplay or even provide players with a new hero upon each death, as examples of the game mode variety. Despite special rules being integral to each game mode, certain core elements such as the number of players and map spawn timings remain consistent. These consistent elements are crucial to the continued advancement of AI within these varied game modes.

All Pick: Players can choose any hero from the full hero pool, random a hero, or swap heroes with teammates. There are 75 seconds for hero selection time and pre-creep time, and players lose 2 gold per second if they haven't picked a hero after the selection timer runs out. Randomizing a hero gives a free and unsellable Enchanted Mango and Faerie Fire, but it ignores the player's 25 least-played heroes [34].

Turbo Mode: This game mode has the same rules as All Pick but with several changes to shorten the match time. All bans are applied immediately, all hero selections are now blind, and there are shorter timers on all ban, pick, and strategy phases [34].

New Player Mode: A simplified game mode ideal for new players. Players may leave these matches without any penalties, and a player who leaves gets replaced by a bot. If new players queue for this mode with a stack, they get instantly placed in a match with bots. This mode uses Turbo Mode modifications, and the hero pool is limited [34].

Play vs Bots: Players can practice by themselves against Bots in a local game or use

matchmaking to play with others against bots. This mode allows you to play against bots with a customized difficulty, including custom scripts created by the community [34].

Single Draft: Players pick from a pool of one Strength hero, one Agility hero, one Intelligence hero, and one Universal attribute hero [34].

1.2.3 CS:GO

Counter-Strike: Global Offensive (CS:GO) is a multiplayer first-person shooter developed by Valve and Hidden Path Entertainment.

As of 2023, CS:GO's monthly player count usually ranges from around 800,000 to 900,000 total players. During peak active hours, CS:GO can reach around 1.5 to 1.8 million players [35].

CS:GO is the fourth game in the Counter-Strike series, following the original Counter-Strike, Counter-Strike: Condition Zero, and Counter-Strike: Source. The game pits two teams against each other: the Terrorists and the Counter-Terrorists. Both teams must complete various objectives to win, with the Terrorists typically trying to plant a bomb and the Counter-Terrorists trying to prevent them from doing so or defusing the bomb if it is planted [36].

CS:GO features several game modes, including Competitive, Casual, Deathmatch, Arms Race, Demolition, Wingman, Flying Scoutsman, and Danger Zone. Each mode has different rules and objectives, but the central theme of Terrorists vs. Counter-Terrorists remains the same. Competitive mode is the most serious and team-oriented, featuring two teams of five players each and a maximum of 30 rounds [36].

In terms of gameplay mechanics, CS:GO is known for its high-skill ceiling and precision shooting mechanics. The game requires good aim, strategy, teamwork, and knowledge of the game's various maps to be successful.

In the competitive scene, strategies play an important role in determining the outcome of a match. If two teams of equal skill face each other, the teams' strategies influence the outcome much more than the individual players' skills. This holds true even when faced with a stronger opponent, as clever strategic thinking can sometimes overcome superior mechanical play.

CS:GO stands out from its predecessors, with a key distinction being its multitude of maps, each offering unique weaknesses and strengths. This gives the game a diversity that adds layers of strategic depth and variety to the gaming experience. The most popular maps each have their own charm and appeal.

The tactics employed in CS:GO are multifaceted, integrating a mix of player movements, weapon selection, and the application of grenades. The initial routes and positions adopted by players in every round form vital elements of a team's game plan.

Gameflow A match in Counter-Strike: Global Offensive (CS:GO) typically takes place between two teams of five players each, with one team playing as Terrorists (T) and the other as Counter-Terrorists (CT). The match is divided into rounds, and the team that wins 16 rounds first wins the match. If both teams win 15 rounds, the match ends in a tie, although overtime rounds can be played to determine a winner in some formats.

Flow of CS:GO Match goes as follows:

- P**istol Round**: A CS:GO match initiates with the much-anticipated 'pistol round.' As the title implies, players begin the game with a limited budget, limiting them to purchasing just pistols and a few ancillary items like light armour or grenades. This round is paramount because its outcome determines the financial upper hand in the subsequent rounds. Winning the pistol round gives the victorious team an initial point lead and awards them a significant economic advantage, which can be leveraged to obtain superior weaponry and equipment in the forthcoming rounds.[37]
- **Following Rounds**: Post the pistol round, the in-game financial system comes into play, and it's one of the key aspects that set CS:GO apart from many other competitive shooters. Players are rewarded with in-game currency based on their individual and team performance metrics, such as a number of kills, successful bomb plantations or defusals, and round victories or losses. This currency can then be invested in procuring advanced weapons, robust armour, and utility items such as different types of grenades. This economic aspect embeds a strategic layer to CS:GO, compelling teams to thoughtfully manage their resources to secure an advantageous position in future rounds. The team's decision-making ability in choosing when to "save" money and when to "buy" items is a crucial determining factor in the team's success [37].

- **Objective**: The game's core objective varies based on the team you're playing. For the Terrorist side, the primary goal is to plant a bomb at one of two predefined locations, referred to as site A or site B, and subsequently guard it until it detonates. Conversely, the Counter-Terrorists are tasked with a defensive role. They must strive to either obstruct the bomb from being planted in the first place or, if the bomb gets planted, they are tasked with defusing it before it explodes. Apart from these bomb-related objectives, a round can also be won by either side if they successfully eliminate all opposing team members [37].
- **Switching Sides**: After completing 15 rounds, the game reaches its halftime, and the teams interchange roles - the Terrorists take on the role of Counter-Terrorists and vice versa. This swap ensures an equitable gaming experience, allowing both teams to strategize and play from both perspectives. The performance in both roles heavily influences the overall outcome of the match [37].
- **Winning the Match**: The victory condition for a match in CS:GO is straightforward - the first team to secure 16-round wins is declared the winner of the match. However, if the match ends up with a 15-15 score after 30 rounds, it may culminate in a tie or advance to an overtime session based on the specific rules of the tournament or game mode being played. During the overtime session, additional rounds are played until a team takes a definitive lead and is crowned as the winner [37].

Economy CS:GO incorporates a complex economy system that significantly influences gameplay dynamics. This system revolves around an in-game currency that dictates the resources available to players, including weaponry, equipment, and utility items.

At the start of each round, players are allocated a specific amount of starting money based on the outcome of the previous round. Winning teams receive a higher starting money while losing teams receive a reduced amount. This initial capital acts as the foundation for subsequent purchases in the round [38].

The outcome of each round has a significant effect on the team's economy. Victory in a round results in additional monetary rewards, providing players with greater purchasing power to invest in more potent weaponry. Contrariwise, a defeat results in a decreased monetary allocation, rendering it more challenging to afford higher-tier items. To maintain game balance, losing teams are granted a loss bonus for each consecutive round lost [38].

This bonus progressively increases the money awarded to the losing team, empowering them to make more robust purchases in subsequent rounds. The loss bonus mechanism mitigates the economic disadvantage faced by struggling teams, giving them a chance to make a comeback [38].

Players can strategically allocate their financial resources to acquire weapons, armor, grenades, and other combat-related equipment. Each item comes with a specific cost, necessitating players to make prudent decisions based on their available funds and the overall strategic objectives of the team. This requires careful consideration of individual preferences, team dynamics, and the anticipated tactics of the opposing team [38].

Effective economy management is of paramount importance in CS:GO. Teams must assess their current financial standing, considering the outcome of the previous round and their long-term strategy. In certain situations, it may be advantageous to conserve funds and opt for minimal equipment in what is known as an "eco round." This enables players to save money for subsequent rounds and potentially secure more powerful weapons. On the other hand, if a team's financial position is favorable, players may opt for a "full buy" strategy, investing in high-quality weaponry and gear to maximize their combat effectiveness [38].

Teams that experience multiple consecutive round losses or consistently poor performance may face an economy reset. During such instances, the team's financial reserves are significantly reduced, making it difficult to purchase equipment. Economy resets can have a profound impact on a team's performance, requiring careful planning and strategic recovery measures to regain a competitive edge [38].

In addition to round outcomes, individual player performance also influences the economy. Eliminating opponents or successfully achieving in-game objectives rewards players with additional monetary incentives. This allows players to bolster their team's overall financial standing, providing more opportunities for tactical decision-making or personal upgrades [38].

Equipment a variety of equipment shapes gameplay. Players wield an arsenal of pistols, rifles, shotguns, and heavy weapons. Armour enhances survivability, while grenades provide tactical advantages such as explosive damage, blinding opponents, or creating smoke screens. Utility items like the Defuse Kit and Zeus x27 offer additional strategic options, such as faster defusing.

Maps CS: GO's gameplay intricacies extend significantly with its diverse map pool, each contributing uniquely to its dynamics. These maps, often based on real-world locales, serve as the battleground where two teams compete for dominance, deploying distinct strategies tailored for each environment [39].

Every map presents unique architectural features, choke points, hiding spots, and open areas that dictate the game's flow. Familiarity with the geography and layout of a map is essential, as it significantly influences the tactics deployed by the teams. Certain maps, for instance, might favour close-quarters combat, necessitating shotguns or submachine guns, while others could cater to long-range engagements, making sniper rifles a popular choice [39].

Each map has designated "bomb sites" where the terrorist team must plant the bomb, and the counter-terrorist team must defend or defuse it. The positioning of these sites, combined with the varying pathways leading to them, establishes a spectrum of strategic possibilities. Teams must consider which routes provide the quickest access, offer cover, and are likely to be heavily guarded [39].

1.2.4 F1 2022: The Game and Its Professional Scene

Formula 1 2022, often called F1 2022, is the official video game of the FIA Formula One World Championship, developed and published by Codemasters. It provides fans and players with an immersive experience that mirrors the thrills and challenges of the real-world F1 season. The game's advanced graphics, physics engines, and meticulous attention to detail have made it the go-to racing simulation for many enthusiasts [40].

Gameplay Dynamics F1 2022 offers a variety of gameplay modes. Players can opt for a quick race, delve into a full season, or even embark on a career mode, where they start as a rookie and aim to secure their place among F1 legends. Each mode offers unique challenges and dynamics, catering to casual players and hardcore racing sim enthusiasts.

The realism in F1 2022 is unparalleled. From the wear and tear on the tires, the impact of weather conditions on the car's grip and aerodynamics, to the exact replication of each track's bumps and curvatures, every aspect is designed to mimic the real-world racing experience [40].

Pro Scene: Esports and Competitive Gaming The professional scene surrounding F1 2022 is expansive and highly competitive. The official F1 Esports Pro Series is the pinnacle, where players worldwide compete for the title of the world's best virtual F1 driver. Teams associated with real-world F1 franchises scout and sign players, highlighting the significant overlap between the virtual and real-world racing scenes [40].

Training and Strategy Professional F1 2022 players train countless hours, refining their racing lines, braking points, and strategies like their real-world counterparts. They often use advanced peripherals, including racing wheels and pedals, to enhance their driving precision. Moreover, they engage in extensive telemetry analysis, collaborating with their teams to make data-driven decisions on car setups and race-day strategies [41].

Data Analysis in F1 2022 In professional F1 2022 gameplay, data analysis has become an indispensable tool for players and teams. With the game's emphasis on realism and precision, victory and defeat can often hinge on minute details and adjustments, many of which are driven by rigorous data analytics [41].

Telemetry and Performance Metrics The game's advanced simulation engine generates vast amounts of telemetry data during races. This includes information on tire wear, fuel consumption, braking points, throttle application, and more. By examining this telemetry, players can identify areas for improvement, whether it's a specific corner they're struggling with or a gear shift they're consistently missing [41].

For example, if a player notices they're losing time in a particular section of a track, data analysis can pinpoint the exact cause — be it late braking, suboptimal acceleration, or an inefficient racing line.

Strategic Decision Making Beyond individual performance, data analysis plays a pivotal role in strategic decisions during races. Real-time analytics often informs understanding of when to pit, which tire compound to choose, or how to manage fuel and engine modes. Teams may employ dedicated analysts to monitor races and advise players on the fly, ensuring they're equipped with the latest data to make informed decisions [42].

Comparative Analysis Professional players also use data to compare their performance against rivals. By overlaying their telemetry data with that of competitors, players can gain insights into different racing lines or strategies being employed. This comparative analysis allows players to adapt and evolve their approach, countering their opponents' strengths and capitalizing on their weaknesses [41].

1.3 History and previous AI implementation in games

From the basic AI mechanisms in early arcade games to the sophisticated algorithms behind modern open-world experiences, the contribution of AI to gaming has undergone a monumental shift. Beyond just influencing non-player character (NPC) actions, AI now shapes rich environments, adaptive storylines, and interactive gameplay elements.

1.3.1 The Evolution of Gaming AI

From the simplicity of Pong and Pac-Man, game AI has come a long way. Later titles, like The Elder Scrolls and Grand Theft Auto, introduced NPCs with varied routines and emotions, demanding more dynamic AI algorithms. The Sims gave characters individual desires and personalities, ensuring unique gameplay experiences. Meanwhile, games like Minecraft utilized AI for procedural world generation, creating diverse and vast landscapes for players. Another innovation, adaptive difficulty, as seen in Resident Evil 4, adjusts gameplay based on player skill, catering to both novices and experts. Today, game AI is pivotal in crafting immersive and tailored gaming journeys.

1.3.2 The Rise of Advanced Game AI

As the turn of the millennium approached, growing computational capabilities paved the way for more refined game AI. Notably, the 'Half-Life' franchise showcased NPCs adept at environment assessment, teamwork, and dynamic responses to players.

The introduction of the A* pathfinding algorithm revolutionized real-time strategy titles like 'Age of Empires' and 'Starcraft.' This algorithm allowed in-game units to navigate intricate terrains, sidestepping obstructions with ease proficiently.

1.3.3 The Evolution of Artificial Intelligence in Competitive Gaming

In the vast panorama of technological evolution, competitive games have remained a steadfast metric for evaluating the strides made in artificial intelligence. From board games to televised quizzes and digital strategy challenges, AI's continual progress in this realm underscores its rapidly advancing capabilities.

Game overview As previously mentioned, 1997 marked a transformative moment in chess when IBM's Deep Blue challenged the world's top player, Garry Kasparov. Their match wasn't merely a game but a symbolic clash between human intellect and machine capabilities. While Deep Blue's remarkable ability to assess millions of positions in a flash was noteworthy, its true advantage lay in its comprehensive database covering both opening strategies and endgame possibilities, providing it with the edge to effectively predict and counter Kasparov's strategies [5].

In 2011, another significant event in the realm of AI took place during the quiz show 'Jeopardy!'. IBM's Watson, utilizing advanced natural language processing, competed against esteemed participants Brad Rutter and Ken Jennings. Watson's subsequent win highlighted its sophisticated ability to understand difficult questions and efficiently extract acceptable answers from its extensive data repository [43].

As noted in the section dedicated to GO. The intricate game of Go was once again thrust into the limelight in 2016. This time, the stage was set for DeepMind's AlphaGo to test its mettle against the legendary Lee Sedol, who boasts a commendable 18 world championships. The outcome, a decisive 4-1 victory for AlphaGo, was awe-inspiring and startling. AlphaGo's utilization of deep convolutional neural networks was a cornerstone of this achievement. This fusion of time-honoured human strategies with self-taught gameplay insights underscored AI's remarkable strides [44].

Showcasing a formidable prowess in DOTA 2, an acclaimed multiplayer online game. When OpenAI's Five was initially tested against human professionals, the outcomes were heterogeneous; there were instances of both dominance and defeat. However, its adaptability was remarkable. Within a mere span of a year, by 2019, OpenAI's Five was competing with and frequently outpacing world-renowned teams [9].

The heart of OpenAI's Five's accelerated progress lay in adopting Proximal Policy Optimization. This technique not only allowed it to refine its strategies but also to internalize vast gameplay dynamics. To provide a perspective on the intensity of its

training regimen, consider this: OpenAI's Five immersed itself in a simulation equivalent to 180 years of daily gameplay. Such an expansive self-play framework enabled the AI to encounter, analyze, and adapt to myriad in-game scenarios, making its gameplay competitive and innovative [9].

Lastly, with its inherent unpredictability and bluffing, the world of poker saw the rise of an AI named Pluribus in 2019. Developed jointly by Facebook and Carnegie Mellon University, Pluribus went head-to-head against elite human players in Texas Hold'em and emerged victorious. The AI's online search algorithms allowed it to tackle poker's uncertainties, demonstrating a groundbreaking capability to excel in games of imperfect information [45].

The Overview of Improvement of AI in Gaming Initially, AI's domain was limited to games with a relatively clear decision-making framework, such as chess. The gameplay in such environments is rooted in "perfect information" scenarios, where all players are fully aware of the entirety of the game state. IBM's Deep Blue, which triumphed over Garry Kasparov in 1997, utilized a brute-force algorithm to evaluate chess positions every second. But even then, it was a remarkable achievement.

Natural Language Processing and Data Retrieval: Transitioning from the structured environment of chess, AI was tested next in the unstructured domain of natural language processing in 'Jeopardy!'. IBM's Watson, through its victory, showcased that AI could comprehend intricate language nuances and swiftly retrieve accurate answers from vast databases.

Addressing Imperfect Information: With Go, the complexity was amplified. The game, known for its vast strategic depth, was always seen as a grand challenge for AI. DeepMind's AlphaGo was not only about deep learning; it was about blending structured strategic gameplay with the ability to adapt in real-time, something many believed was still years away for AI.

Multiplayer Dynamics: DOTA 2, a multiplayer online battle arena game, introduced an environment of chaos, real-time decision-making, teamwork, and ever-changing dynamics. OpenAI's Five's success in such a game emphasized AI's ability to adapt to complex multiplayer scenarios and rapidly evolving situations.

Incorporating Bluffing and Deception: Finally, poker posed an entirely different challenge: it's a game of hidden information and psychological play. Pluribus' success

in Texas Hold'em poker signified a groundbreaking development, with AI mastering a game deeply rooted in human intuition, bluffing, and deception.

In Summary: This ongoing journey from simpler to complex games has become a testament to AI's incredible adaptability and growth. AI has demonstrated the capacity to process and compute vast amounts of information and has also showcased cognitive abilities like strategy formulation, adaptation, and even elements of psychological gameplay. The narrative that unfolds through these games suggests a future where AI's capabilities may transcend many realms previously believed to be the exclusive domain of human expertise.

II. PRACTICAL PART

2 DATA HANDLING (LEAGUE OF LEGENDS DATA ANALYSIS)

In this thesis, I have focused on advancing the work of Műller Štěpán[46] by addressing the previously unexplored aspect of vision in MOBA games, specifically through the introduction of a concept called 'fog of war'. Building upon the solid foundation established by his initial research, I aimed to enhance and expand his findings by incorporating the 'fog of war' element.

2.1 Data selection

The data source is derived directly from Riot Games, the creator of League of Legends. This unique dataset has been meticulously compiled by selecting professional games played on patch 13.4, released on February 22, 2023, and lasting until March 8, 2023.

776 games were collected from this patch. Games with missing events or other issues that might have compromised the integrity of the data were then filtered out. Focusing on this specific patch and the advanced gameplay demonstrated in professional matches guarantees that the analysis is based on the most pertinent and top-tier data available. This approach allows us to provide valuable insights into the strategies and techniques employed by top players, shedding light on their decision-making processes and the factors that contribute to their success in the game. Games that were too brief for meaningful analysis were also excluded from the dataset. In the study, it was determined that games lasting less than 10 minutes did not provide sufficient data for a thorough examination of player strategies and behaviours. With this time constraint imposed, the selection was refined to 568 games that met the criteria. Ultimately, an extensive dataset, encompassing over 84 GB of data from the selected games, remained.

2.2 Data Composition

The data for the analysis are comprised of a series of JSON files. For a more streamlined understanding, these files can be broadly categorized into three distinct types: Header Data, Pick Phase Data, and Game Data.

1. **Header Data**: These contain essential metadata about the game, introducing the other files and providing vital information for the analysis.
2. **Pick Phase Data**: These files encapsulate details of the pick-ban phase of the

game. Though rich in information, they are deemed irrelevant to the current analysis and thus have been excluded from the study.

3. **Game Data**: This segment constitutes the core of the dataset, capturing all the in-game events, player statistics, and other aspects deemed vital for the examination.

For the research objectives, the emphasis will be placed exclusively on the Header Data and Game Data.

2.2.1 Header data

The process of data acquisition involves downloading segmented JSON files. Each file contains a unique ID and a "nextPageToken", facilitating seamless navigation through the dataset. The first file, known as the header, contains essential information about the game, such as the "gameID" and "sequenceIndex." These data points make it possible to detect any missing files and evaluate the completeness of the game data. The "sequenceIndex" is displayed as a negative number in the header file but shifts to non-negative values (starting from 0) as the game progresses beyond the pick-ban phase.

The "rfc460Timestamp" is used to ensure that the game data corresponds with the specified patch 13.4.1. Although the "platformID" isn't used in the current analysis, it can be significant for future investigations. Categorizing games by region could reveal distinct strategies adopted by players from various regions.

While the header file contains details like champion bans, champion selections, player names, skins, and selected summoner spells, it often starts with partial data. This incomplete information necessitates supplementation from the subsequent data segments. Relying on these subsequent segments ensures the comprehensive collection of data.

2.2.2 Game data

Within the structure of the game data, a prominent feature is the division into various events. These events encompass a wide array of information, capturing specific actions, interactions, and changes occurring within the game. While most of these events serve as valuable data points, a notable aspect is the presence of redundancy among certain

classes of information. This redundancy is not without purpose, as it often serves as a means of ensuring consistency and validation.

The main event known as **stat_update** emerges as a central reference point. Its importance lies in its regularity, being updated approximately every 1000 ms. This frequent updating makes it a reliable source for tracking various in-game statistics. Many of the details captured in other events can also be found in this main event, as highlighted in yellow in Table 2.1.

A clear illustration of this redundancy can be found in handling player levels. When seeking to understand when a player has earned enough experience points to gain a new level, there are multiple ways to access this information. One could examine the specific class **champion_level_up,** dedicated to this occurrence. However, this same information can be tracked within the **stat_update** class, where all players are listed along with their current levels.

Besides the redundant event classes, additional information is required from certain events to serve the intended purposes. Examples of these irrelevant data include player nicknames, skill names, account IDs, and similar details. These classes are marked in red and labelled irrelevant in Table 2.1.

Event Name	Type
stats_update	important
item_purchased	redundant
item_destroyed	redundant
ward_placed	important
skill_level_up	redundant
champion_level_up	redundant
epic_monster_kill	important
ward_killed	important
champion_kill	redundant
item_sold	redundant
building_destroyed	important
turret_plate_destroyed	redundant
item_undo	redundant
champion_kill_special	irrelevant
epic_monster_spawn	redundant
queued_dragon_info	important
mute_action	irrelevant
game_end	important
quit	redundant
pause_ended	irrelevant
pause_started	irrelevant
reconnect	irrelevant
surrender	irrelevant
champion_transformed	redundant

Table 2.1 Event Classes in Data Files

Thus, ultimately, the task of identifying and discarding these redundant or immaterial details was deemed vital for dataset management. By emphasizing crucial components and eliminating repetitive ones, significant data simplification was achieved. This action not only rendered the dataset more streamlined and concentrated but also markedly influenced its volume.

Through a methodical removal of these extraneous parts, a data size reduction by a factor of 6 was attained. Such a decrease, while upholding the data's integrity and pertinence, enhanced the efficiency of the analytical processes.

2.3 Creating Additional Data

Upon delving into the game files for analysis, it was promptly discerned that some components were absent. The available data provided substantial insights, yet there remained elements essential for a comprehensive understanding that were absent. Therefore, the decision was made to generate additional data aligned with the research objectives. The ensuing sections introduce these supplementary elements, bridging existing gaps and offering a more lucid depiction of in-game dynamics.

2.3.1 Fog of war data

A significant enhancement to Muller's research is the inclusion of the 'fog of war' data. This necessitated the reconstruction of specific game segments, given that the vision element was not directly extractable from the existing data. Yet, the essential components to assemble this information were available. In "League of Legends," vision primarily originates from four sources:

1. **Champions:** Player-controlled characters that offer vision as long as they remain alive and are not **blinded** by an adversary's actions. Every champion has a vision range spanning 1350 units around them (for context, the map roughly measures 15000x15000 units). Their x and y coordinates were harvested from the supplied dataset for subsequent vision computations.

2. **Turrets:** Static edifices that share the same vision radius as champions, set at 1350 units. Being stationary entities, their precise positions were not inherently present in the dataset. Their locations were deduced by scrutinizing over 100 games spanning different patches and observing the instances and timings of turret deaths. Since turrets don't reswpawn post-destruction by adversaries or minions, their contribution to the vision is rendered obsolete.

3. **Minions:** Non-player characters (NPCs) that traverse a preordained route founded on specific waypoints. Their course can be disrupted by factors like skirmishes with enemy champions, turret engagements, or encounters with opposing minions. Endeavours to replicate these waypoints benefited from Riot Games' documentation [47]. Since they're intrinsically embedded within the game, exact coordinates were difficult to track down. However, pivotal waypoints, checkpoints such as **turrets**, and additional waypoints on the map's upper and lower sides were discerned. This methodology is depicted in Figure 2.1. Their consistent

spawning commences at 1:05, with subsequent spawns every 30 seconds. Their foundational movement is set at 350 movement units, which sees variations as detailed in Table 2.2.

Minutes	**10**	**15**	**20**	**25**	**25+**
Additional Movement Speed	25	50	75	100	100

Table 2.2 Additional movement speed at different minutes for minions

Given the predictable nature of minion movement and their predetermined waypoints, it becomes feasible to compute their precise coordinates on the map, except for interruptions by champions or adversarial minions. This computation can be achieved using the formula in Equation 2.1.

$$\text{Minion Distance Traveled} = \left(350 + \min\left(25, \frac{\text{Game Time (ms)}}{60000}\right) \cdot 25\right) \cdot (\text{Game Time (ms)} - \text{Minion Spawned (ms)}) \quad (2.1)$$

However, as their movement can be obstructed, enemy structures were chosen as the primary factor impeding their progress, given that minion waypoints are closely tied to these structures. When minions arrive at a waypoint near an enemy structure, they pause for a variable amount of time. This duration is calculated based on whether continuing forward without stopping would result in reaching another waypoint, in which case they would be considered dead, having been eliminated by a champion, tower, or opposing minions.

Minions possess a different vision range than champions, featuring a vision radius of 1200 units instead. This distinction must be taken into account during the vision calculations to ensure an accurate representation of the fog of war and the overall vision dynamics within the game.

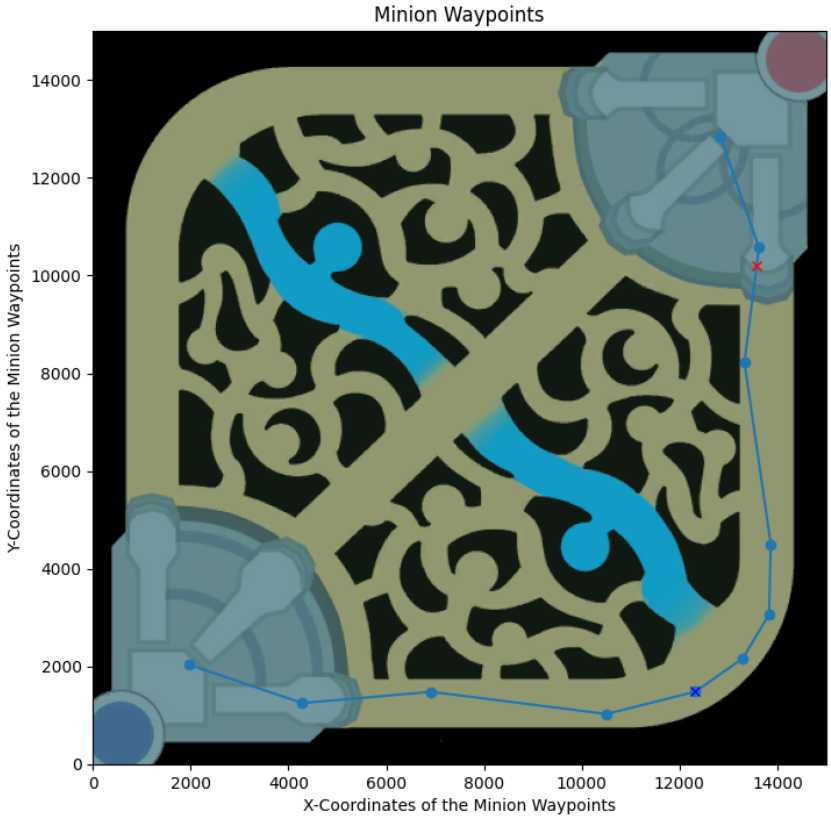

Figure 2.1 Minions Waypoints on bottom side for blue team

Wards are small, usually temporary, structures placed by players on the map to grant vision of an area. Wards are essential strategic tools that help teams maintain map control, monitor enemy movements, and avoid potential ambushes. there are different types of wards:

1. **Totem Wards:** These wards grant 900 units of vision and remain invisible to opposing players. Serving as the standard ward for this study, they can be destroyed by enemy champions and have a limited duration Equation 2.2 [48].

   To determine the expiration formula, the average level of all champions at the time the ward was placed had to be computed.

$$\text{Totem Ward expiration} = 90 + \frac{30}{17}(\text{average of all champion levels} - 1) \quad (2.2)$$

   In addition to the ward expiration times, there are limits on the number of wards

each player can place on the map, which also need to be taken into account. The basic limit for placing wards is three per player. If a player places a fourth ward, the first ward they placed will be destroyed, ensuring that the maximum number of active wards per player remains at three. It is essential to consider this ward limit when analyzing vision dynamics within the game [49].

In addition, there is an in-game item called "Vigilant Wardstone" that can be purchased and increases the warding limit for **Totem**, **Stealth**, and **Control** wards by one. When a player has this item, they can simultaneously place up to four wards of the mentioned types on the map. If they place a fifth ward, the first one will be destroyed, just like the standard warding limit scenario [50].

2. **Stealth Wards:** For the thesis objective, stealth wards were treated analogously to Totem Wards because of their inherent similarities.

3. **Control Wards:** Control Wards are an essential strategic tool they provide 900 units of vision[49] and have the unique ability to reveal and disable enemy **Stealth** Wards, **Totem** Wards, and other invisible traps. Although they are visible to the enemy team and can be destroyed, they do not have a set expiration time and can remain on the map until destroyed by an opponent [49].

   Players are typically limited to placing one Control Ward on the map at a time. However, with the "Vigilant Wardstone" item, this limit increases to two Control Wards per player [50].

4. **Zombie Wards:** These wards are spawned when a player with the Zombie Ward rune destroys an enemy ward. Zombie Wards grant the same 900 units of vision. For the purposes of this analysis, Zombie Wards are treated the same as **Totem** Wards. Both ward types share the same vision range and serve similar purposes in terms of providing vision and maintaining map control [49].

5. **Farsight Wards:** Also known as Farsight Alteration, these wards are unlocked at level 9 and replace the player's Totem Ward. They reveal an area of 900 units for 2 seconds. The Farsight Ward itself has a vision radius of 500 units and lasts indefinitely but is visible to the enemy team and can be destroyed [49].

6. **Ghost Poros:** Spawned by the Ghost Poro rune when a player enters brush, these wards have a vision radius of 450 units and last for 5 minutes. Ghost Poros are visible to the enemy team and can be destroyed by walking over them

or placing a Control Ward nearby. Each player can have only one Ghost Poro active at a time [49].

In addition to the primary sources of vision in League of Legends, it's crucial to take into account vision deniers such as **Brushes**. Scattered throughout the game's map, brushes are patches of tall grass that obstruct vision for both teams. Champions or wards situated within a brush are concealed from the opposing team's view unless the enemy has vision inside the same brush or employs abilities or items that reveal the area.

Brushes hold significant strategic importance in the game, as they enable players to set up ambushes, dodge enemy vision, and orchestrate surprise attacks. To mitigate the effects of brushes on vision control, players frequently place wards within or close to brushes, thereby maintaining better map control.

All these elements can be seen in the following Figure 2.2

Figure 2.2 Example of vision objectives from a selected game at game time 10:00

The Euclidean distance is employed as a measurement tool to ascertain if each player is within the **fog of war** for each champion. This calculation helps us verify if a player is within the vision range of each vision source, which includes champions, wards, minions, and turrets, each with their specific vision ranges.

If the Euclidean distance between the player [**p**] and the vision source [**q**] Equation 2.3 is less than or equal to the source's vision range, the player is considered visible. Conversely, if the distance exceeds the vision range, the player is deemed to be in the **fog of war** and is invisible to the opposing team.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2.3}$$

This approach allows us to accurately simulate the vision dynamics in League of Legends, contributing to a more comprehensive and realistic analysis of the game.

2.3.2 Additional data

This section provides an overview of the additional data that were missing from the original game files but were specifically added to enhance this model.

- **Buff states**

  Keeping track of the various buffs held by champions during the progression of a game. Buffs like the Baron Nashor buff, Red Buff, etc., play a pivotal role in the game's outcome. A detailed summary was created of each buff state to capture this, reflecting which champion possesses a particular buff at any given time.

- **Building Status**

  A variable representing the building respawn timings was developed. A detailed summary of each building's respawn state was created to capture this, including information such as time elapsed since the destruction, respawn time, and location within the game map.

- **Ward Status**

  A detailed examination of each ward's state was conducted. This included placement location, duration, the player responsible for the ward placement, and eventual expiration or removal.

- **Skill Status**

  This involved capturing details like the activation time of specific skills, the duration and the remaining cooldown of abilities at any given game update.

- **Summoner Spell Status**

  Dataset encompasses essential details regarding summoner spells in the game, including the remaining cooldown for a specific spell, the total duration of that cooldown, and the name of the specific spell itself.

- **Team Statistics** In the dataset, the "Team Statistics" category provides an invaluable snapshot of various team-related metrics that paint a picture of a team's performance during a game. These statistics include:

  - Tower Kills Representing the number of enemy towers a team has destroyed.
  - Assists The total assists credited to the team, showcasing teamwork and collaboration.
  - Inhibitor Kills The count of enemy inhibitors taken down by the team, often an indicator of map control.
  - Total Gold The sum of the gold earned by the team, reflects overall resource acquisition.
  - Champion Kills The cumulative kills made by the team against enemy champions.
  - Deaths Total deaths within the team.
  - Dragon Kills A number of dragon kills, highlight a team's control over specific map objectives.
  - Baron Kills Reflecting the number of barons kills secured, another measure of strategic dominance.

  These variables collectively offer a multifaceted view of a team's strategic execution, objective control, collaboration, and overall performance.

- **Data Dragon Integration**

  For a more detailed and visually engaging presentation of game data, extra information was imported about champions, including their statistics, cooldowns, skills, perks and images. This importation was made possible through Data Dragon (or ddragon), a rich repository that offers static data files encompassing essential elements of the game.

"Data Dragon serves as an invaluable resource for translating champion IDs to names, providing images, and giving details about champions' abilities, runes, and items. The full set of data, amounting to approximately 1 GB, can be downloaded as a tarball from the official site by specifying the desired patch version [51]."

- **Map Spliting** In the model, there is an inclination to determine players' decision-making process regarding their movement on the game map. A precedent in this area has been set by a Műller, whose findings have been integrated based on [46].

  This method, introduced by Műller, was inspired by Ye et al.'s work on DOTA [52]. The square map was thus segmented into defined sections, with the choice being a 12x12 grid, resulting in 144 distinct sectors.

$$Y_{t,c}^{\text{pos}}(x) = i \tag{2.4}$$

  of the sector where the champion c is present in the next game state x_{t+1} [46].

  This approach facilitated estimating the probability of a champion appearing in a given sector next. Leveraging the insights from Műller's prior testing, this rudimentary technique was adapted and refined for the current model.

2.4 Data Overview

This section is dedicated to providing an overview of the final dataset, detailing the specific attributes and characteristics that were compiled for the model. A comprehensive example of pseudo data, representing a random timestamp within an average game in League of Legends, can be found in the following diagram Figure 2.3.

It's worth noting that an average game in League of Legends typically spans between 25 to 30 minutes. As a result, the data points represented in the aforementioned table may recur for each game approximately 1500 to 1800 times.

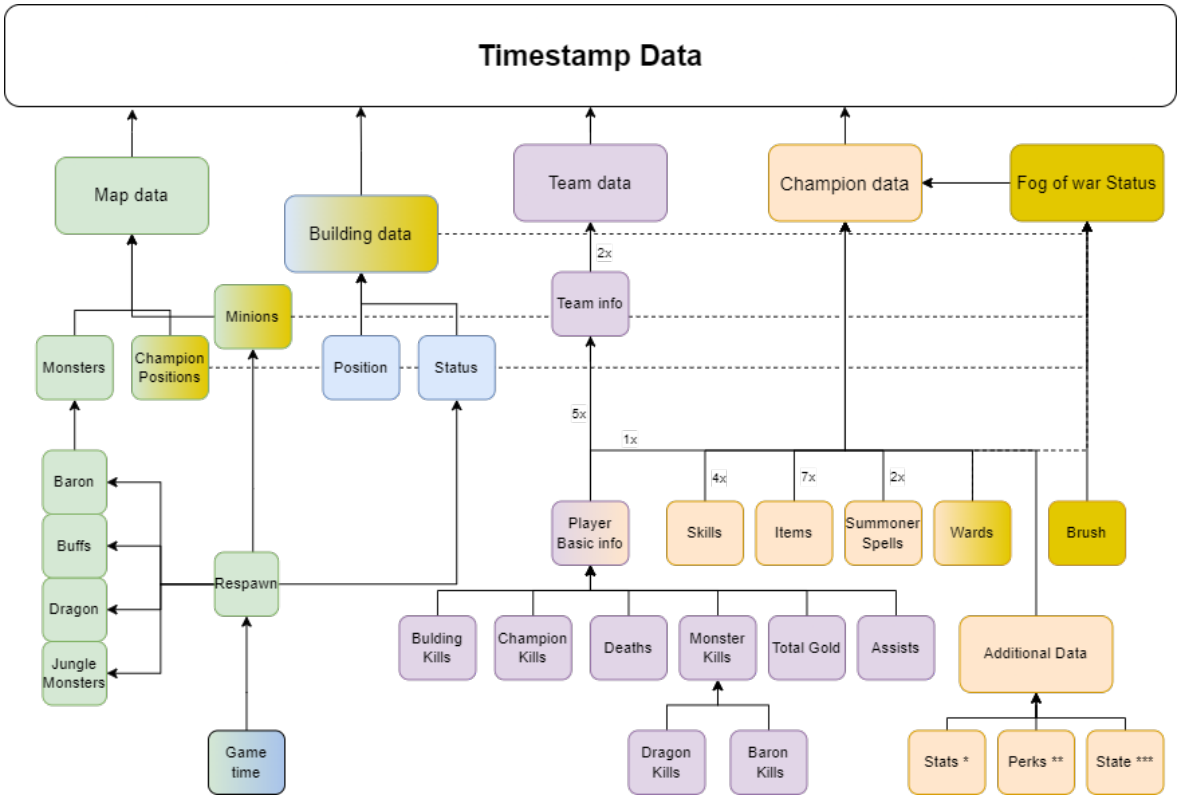

Figure 2.3 Example of timestamp data

Note: Details about **Stats** and **State** are to be found in Table 2.3 and more details about **Perks** can be found in Table 2.4

Table 2.3 Champion State and Statistics

State	Index	Stat
championName	1	minions killed
alive	2	neutral minions killed
respawnTimer	3	neutral minions killed your jungle
level	4	neutral minions killed enemy jungle
XP	5	champions killed
position	6	num deaths
health	7	assists
healthMax	8	ward placed
healthRegen	9	ward killed
magicPenetration	10	vision score
magicPenetrationPercent	11	total damage dealt
magicPenetrationPercentBonus	12	physical damage dealt player
armorPenetration	13	magic damage dealt player
armorPenetrationPercent	14	true damage dealt player
armorPenetrationPercentBonus	15	total damage dealt to champions

currentGold	16	physical damage dealt to champions
totalGold	17	magic damage dealt to champions
goldPerSecond	18	true damage dealt to champions
shutdownValue	19	total damage taken
primaryAbilityResource	20	physical damage taken
primaryAbilityResourceMax	21	magic damage taken
primaryAbilityResourceRegen	22	true damage taken
attackDamage	23	total damage self mitigated
attackSpeed	24	total damage shielded on teammates
abilityPower	25	total damage dealt to buildings
cooldownReduction	26	total damage dealt to turrets
lifeSteal	27	total damage dealt to objectives
spellVamp	28	total time crowd control dealt
armor	29	total heal on teammates
magicResist	30	time ccing others
ccReduction	31	
ultimateName	32	
ultimateCooldownRemaining	33	

For Perks, players can choose from five different categories for customization, each with a unique focus on defence, offence or utility.

Each player's perk page contains perks from two paths: a primary and a secondary. The primary path includes one keystone and three lesser perks, while the secondary path includes two lesser perks. In addition, three Shard slots follow a separate path from the main five categories. Each slot can be filled with one selection.

These choices have 55,566 possible combinations, allowing for deep customization to fit different play styles and strategies.

Table 2.4 Perks

Index	**Perk**
0	perkID
1	var1
2	var2
3	var3

Due to the wide variety and intricacy of these options, it is not feasible to integrate the complete table into the analysis. As a solution, these selections are represented by

var 1, 2, and 3. This abstraction streamlines the data yet preserves the fundamental facets of players' decision-making in their customization, as illustrated in Table 2.4.

3 MODEL CREATION AND IMPLEMENTATION

In an effort to provide players with a comprehensive understanding of in-game win probability, a method previously established was adopted and adapted. This model, originally devised by Maymin et al. [53], utilized logistic regression to consider various game parameters: game time in minutes, team kill counts, the number of turrets destroyed by both teams and the tally of epic monsters that each team defeated.

Basing the approach on their methodology that was later adapted for League of Legends by Műller Štěpán [46], in the output can be viewed a graph showcasing the fluctuating win rate throughout the match in Figure 3.1.

Figure 3.1 Win prediction Graph example of the game [KNF vs MFSK (game id 2773154) from the perspective of blue side (Lose)]

After a thorough analysis of the dataset, the focus was directed towards the most pertinent statistics, discerned from testing and evident data correlations, as depicted in Figure 3.2. This allowed us to refine the dataset by eliminating superfluous data. Drawing inspiration from Muller's 2022 methodology [46], the method of categorizing monsters into distinct groups was integrated, introducing features that aggregate data on turrets, inhibitors, and other structures while differentiating various dragon types. Additionally, experimentation with the "fog of war" feature was undertaken, and its performance was benchmarked against a model without this feature.

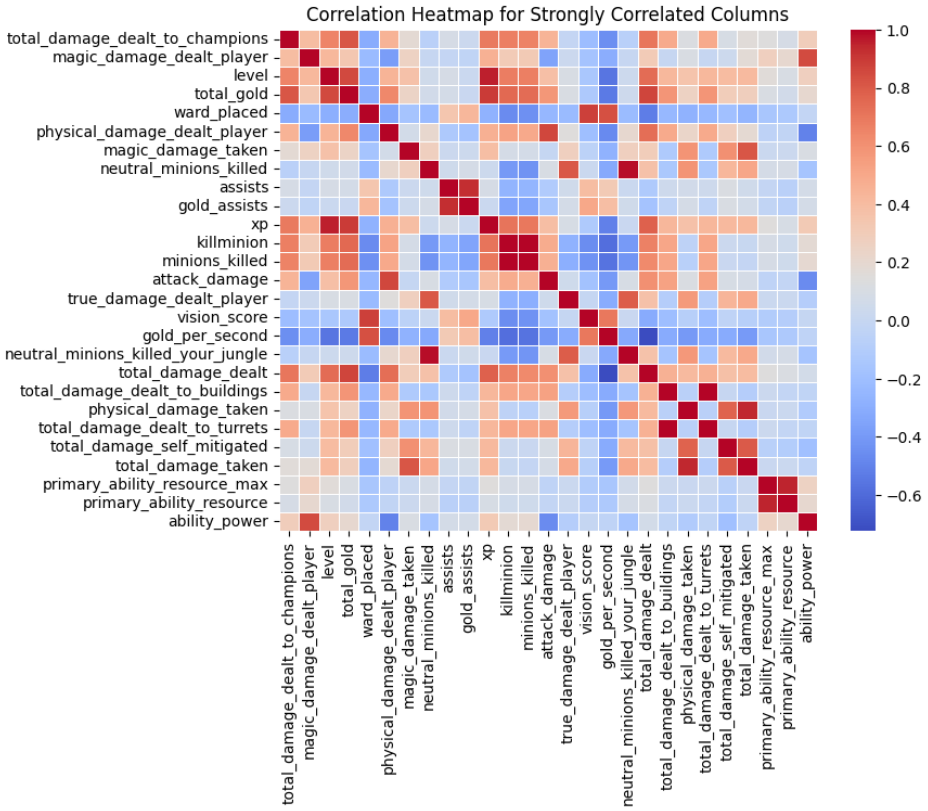

Figure 3.2 Selected data correlation example from the datasets

Building upon Muller's foundational work, the approach was modernized to align with the current game patch. This update came with a variety of new champions and distinctive features, including dynamic map alterations targeting specific areas. Recognizing that League of Legends evolves continuously, with each patch introducing fresh changes, this study delves into the profound impact these patches have on outcomes and conclusions.

In the ever-evolving landscape of League of Legends, the concept of a "stronger team composition" is prevalent. This suggests that during specific patches, certain champion synergies prove to be exceptionally effective, resulting in an elevated likelihood of securing a victory.

The K-modes clustering algorithm was integrated, specifically utilizing the 'Huang' initialization[54], starting with five randomized initiations. Based on insights from Muller's methodology, it was decided that k = 4 clusters would be used. Where he suggested a distinct champion attribute characterized by the variance between the 5-hot encoded vectors representing the champions of the competing teams. To discern specific team compositions and evaluate their potential win rates between them.[46]

A slightly adjusted version of the algorithm from Maymin pseudocode[53] was employed to reassign player roles in Algorithm 1.

Algorithm 1 Automatic categorization of champion roles implemented by [53]

```
1: procedure CategorizeRoles(Champions, earlyMinions, neutralJungleMinions,
   mapCenterMinions, proximityData)
2:     Define earlyMinions as minions killed by a champion between levels 2 and 6.
3:     for each team in Champions do
4:         jungler ← champion with most neutralJungleMinions killed
5:         Remove jungler from team
6:         support ← champion with fewest earlyMinions killed
7:         Remove support from team
8:         mid ← champion with most mapCenterMinions killed
9:         Remove mid from team
10:        ADC ← champion most frequently near support in first 10 minutes
11:        Remove ADC from team
12:        top ← remaining champion
13:    end for
14:    return role categorization for each champion
15: end procedure
```

Efforts were made to go through games, emphasizing traditional roles like top-jungle-mid-marksman(bot)-support, while filtering out unconventional gameplays. A lack of minion death data necessitated a creative approach. Instead of relying on minion death statistics, attention was shifted to the duration a player spent near their assigned lane during the initial 10 minutes of gameplay. This approach, reminiscent of Muller's adaptation, was deemed fit for this purpose.

Furthermore, while allowing junglers to farm the mid lane was previously in vogue, with the mid lane essentially functioning as a secondary support character, the 13.4 patch incorporated penalties for such jungler tactics. The strategy became inefficient and obsolete in professional matches as a result. With these revised player roles, it became feasible to methodically organize features at both the team and individual role levels.

The adjusted pseudocode looks as follows for categorizing roles looks as follows in Algorithm 2.

Algorithm 2 Automatic categorization of champion roles adjusted

```
1: procedure CategorizeRoles(Champions, earlyMinions, neutralJungleMinions,
   mapCenterMinions, proximityData)
2:     Define earlyMinions as minions killed by a champion between levels 2 and 6.
3:     for each team in Champions do
4:         jungler ← champion with most neutralJungleMinions killed
5:         Remove jungler from team
6:         support ← champion with fewest earlyMinions killed
7:         Remove support from the team
8:         mid ← champion with most time spent mid lane in first 10 minutes
9:         Remove mid from team
10:        ADC ← champion most frequently near support in first 10 minutes
11:        Remove ADC from team
12:        top ← remaining champion
13:    end for
14:    return role categorization for each champion
15: end procedure
```

Feature	Description
gold	Total gold per team.
kills	Total number of kills per team.
level	Level of each champion.
level mean	The average level of champions within a team.
fog of war	Boolean information whether each champion is in the fog of war or not.
respawn	The time left for each champion to respawn, measured in seconds, with a value of 0 indicating champions that are currently alive.
alive	Count of alive members for each team.
champions n-hot	A vector sized by the total unique champions, with each entry signifying a champion's team affiliation: 1 denotes the champion belongs to the blue team, while -1 signifies the red team. [46]
champion clusters	Denoted by the difference in one-hot encoded vectors that represent the team clusters for both teams.[46]
dragons	The total number for each team defeated a specific dragon type.
dragons total	The total number of dragons slain by each team.
barons	The total number of barons slain by each team.
epic buffs	The remaining duration, in seconds, for the global buffs granted to both teams after defeating Baron Nashor and the Elder Dragon.
epic monsters	The count of epic monsters, either dragons or Baron Nashor, defeated by each team.
turrets	Boolean information for each turret for each team, indicating whether the turret was destroyed.
turrets per lane	The count of turrets taken down in each lane by each team.
turrets total	The count of turrets taken down by each team.
inhibitors total	The count of inhibitors taken down in each lane by each team.
inhibitors per lane	The count of inhibitors taken down in each lane by each team.
inhibitors respawn	The remaining respawn duration, in seconds, for the inhibitors in every lane for both teams.
wards total	The total number of active wards for each team. (alive wards)

Table 3.1 Features and their descriptions

Using 5-fold cross-validation, the models' performance was evaluated across different

features. Initially, the game data was divided into five balanced subsets. One subset was earmarked for testing, while the other four facilitated model training. This procedure was executed five times, rotating the test subset in each instance. The outcomes were then averaged over these iterations.

The advantage of k-fold cross-validation is its ability to reveal the model's adaptability to new, unviewed data. This methodology is particularly beneficial for smaller datasets with just 568 games. Instead of a singular dataset split, k-fold cross-validation provides a more comprehensive average over multiple partitions.

While gradient-boosted trees and random forests showcased commendable training accuracy, they faltered in adapting to new data with Muller's approach [46]. And multilayer dense networks had the same outcomes with longer training. Logistic regression was a top performer. Thus, the analysis chiefly revolves around it and its varying feature combinations. Comprehensive results are elaborated in Table 3.2.

Beginning the investigation, established foundational features and the potential of each were systematically examined. The evaluation process involved: the removal of an existing foundational feature, the introduction of a new one, or the enhancement of an already-present feature to provide more detailed insights. The systematic ablation was conducted on the baseline model, structured following Maymin et al.'s design [53]. This model predominantly hinged on parameters such as the elapsed in-game time, the collective kills by each team, the tally of turrets dismantled by either side and the total of epic monsters defeated.

In contrast, Műller's research offered an alternative approach, excluding the 'elapsed in-game time' after determining its negligible impact on outcomes [46].

For the primary model, labelled as Base_fow, an enhancement was introduced: the incorporation of the 'fog of war' mechanism, augmenting the foundational Base model. The results, as outlined in table Table 3.2, indicated that the 'fog of war' feature did not significantly improve prediction accuracy when considered in isolation. However, this outcome was expected. The strategy was then shifted towards combining it with other features, hoping to find a synergy that could enhance performance.

The baseline model set the stage with a commendable training accuracy of approximately 74.35% and a test accuracy close to 73.83%. Upon delving deeper into feature engineering, some intriguing observations were made.

When the fog of war feature was singularly added, there was a sharp dip in training

and test accuracies, plummeting to around 53.93%. However, this wasn't the end of its story. When harmoniously blended with other features in the Base_fow model, the fog of war showcased a mild improvement compared to the base model.

While experimenting with various features, gold and level mean emerged as standout contributors. Their addition to the Base_fow model, both individually and in conjunction, consistently heightened the performance metrics. On the contrary, the addition of features like vision, alive, and respawn, or the standalone champions n-hot, led to a conundrum — their high training accuracies contrasted with noticeably lower test results, hinting at potential overfitting.

Moreover, when the monsters feature was supplanted by more specific entities like barons or dragons or combined as epic buffs, barons, and dragons, there was a discernible uptick in performance. Yet, introducing certain granular features like wards per type, wards total, and specifics about inhibitors seemed to have a muted impact on the model's prowess.

A moment of revelation came with the integration of all features, pushing the training accuracy to a laudable 76.89%. However, a test accuracy of about 74.76% signalled caution, nudging us towards the spectre of overfitting that such comprehensive models often grapple with.

Among the cavalcade of models tested, a few deserved a special mention. Models enriched with combinations such as Base_fow devoid of kills but enhanced with gold and level mean or further adorned with 'alive' or 'epic buffs' consistently outperformed, crossing the 75% threshold in both training and test results.

Upon retrospection, it's evident that while numerous models showcased impressive prowess during training, they often struggled when introduced to novel data. This underlines the ever-present conundrum of overfitting in model training. Amid this intricate interplay of features, two emerged components and shoulders above the rest: gold and level mean, consistently shining in their ability to enhance the model's predictive strength.

Upon reviewing the outcomes, models built on Base_fow, excluding 'kills' and enriched with 'gold' and 'level mean', stand out. Particularly, "Base_fow - kills + gold, level mean" and its derivatives with 'alive' or 'epic buffs' present as optimal choices for wider use which can be seen in Figure 3.3.

Selecting the "Base_fow - kills + gold, level mean" model aligns well with in-game

dynamics. Kills in a game often equate to dual advantages: the immediate gold and experience bonuses for the victor and a concurrent deprivation of gold and experience for the vanquished due to their temporary absence from play. Thus, by focusing on the gold and experience (level mean) metrics, this model potentially comprehensively encapsulates kills' multifaceted impact.

Features	**Train Acc**	**Test Acc**
Baseline	74.347%	73.827%
Base	74.342%	74.143%
Fog of war	53.926%	53.830%
Base_fow	74.368%	73.934%
Base_fow - monsters	72.148%	71.681%
Base_fow - turrets total	73.439%	73.045%
Base_fow - kills	73.439%	73.045%
Base_fow + vision	71.949%	71.541%
Base_fow + inhibitors total	74.394%	73.881%
Base_fow + gold	75.757%	75.245%
Base_fow + level	74.802%	73.875%
Base_fow + level mean	75.498%	74.256%
Base_fow + respawn	74.455%	73.956%
Base_fow + alive	74.393%	73.891%
Base_fow + champions n-hot	84.501%	67.634%
Base_fow + champion clusters	74.709%	73.625%
Base_fow + epic buffs	74.419%	74.043%
Base_fow + inhibitors per lane	74.399%	73.831%
Base_fow + inhibitors respawn	74.353%	74.136%
Base_fow + wards total	74.310%	73.732%
Base_fow + wards per type	74.355%	73.728%
Base_fow - monsters + barons, dragons total	74.467%	73.891%
Base_fow - monsters + barons, dragons	74.733%	73.600%
Base_fow - monsters + epic buffs, barons, dragons total	74.528%	73.978%
Base_fow - turrets total + turrets	74.788%	73.768%
Base_fow - turrets total + turrets per lane	74.429%	73.789%
Base_fow - turrets total + turrets per tier	74.413%	74.080%
Base_fow - kills + gold, level mean	75.838%	75.593%
Base_fow - kills + gold, level mean, alive	75.835%	75.609%
Base_fow - kills + gold, level mean, epic buffs	75.780%	75.628%
Base_fow + gold, level mean	75.903%	75.316%
Base_fow + gold, epic buffs	75.791%	75.316%
Base_fow + level mean, epic buffs	74.961%	74.321%
Base_fow - monsters + gold, epic buffs, dragons total	76.005%	75.433%
Base_fow - monsters + gold, barons, dragons total	76.121%	75.659%
All features	76.893%	74.764%

Table 3.2 Train and test accuracies of logistic regression obtained from 5-fold cross-validation averaged over 15 runs.

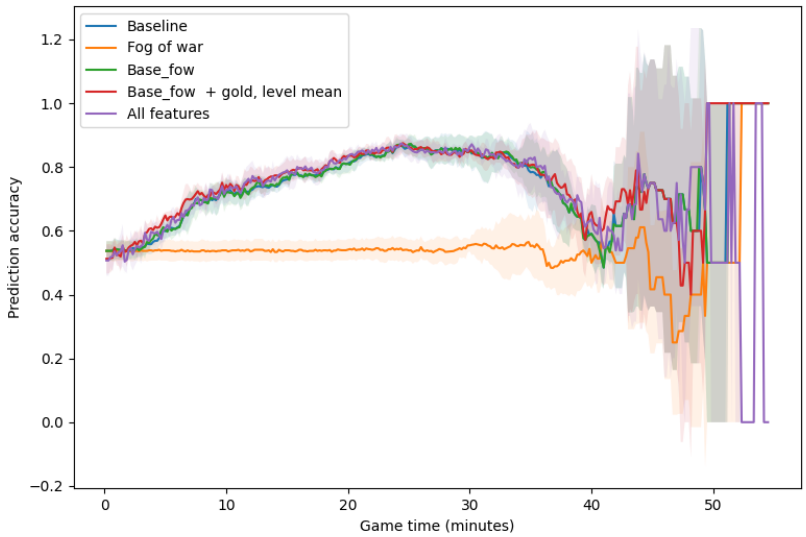

Figure 3.3 Prediction accuracy of win prediction models, depending on game time with selected features Table 3.2

3.1 Model features

Given the multifaceted and ever-evolving nature of game data, the primary objective was to distill this information into a streamlined vector representation for each time step. Based on the depth of historical data needed, two different neural network architectures were chosen: an LSTM for analyzing a sequence of the past N game states and a simpler, fully connected network when only the latest state was the focus. The outcome was a condensed representation of the game's complex history as a distinct vector.

For a structured approach to the thesis, the collected games were meticulously divided into three distinct categories: training, testing, and validation. The ratios were fixed at 80% for training, with the remaining 20% evenly divided between testing and validation, resulting in a 10% allocation for each. The train-test-split module from the sklearn library was utilized to achieve this split. The technique employed for encoding the game state bears a resemblance to the approach taken by Berner et al.[9] and Műller [46] in their comprehensive analysis of DOTA 2 and the subsequent analysis of LOL by Műller. A notable feature of the data preprocessing was the normalization of

all numerical data. Whether they were boolean values, floating numbers, or any other numeric type, they were adjusted to have a mean of zero and a standard deviation of one. This was achieved by calculating the mean and standard deviation of each feature within the training data, and then subsequent features were recalibrated using the formula provided in Equation 3.1.

$$\text{feature}_{(i)} = \frac{\text{feature}_{(i)} - \mu_{(i)}^{[\text{train}]}}{\text{std}_{(i)}^{[\text{train}]}} \tag{3.1}$$

Certain features posed unique challenges. For example, features that consistently had a single value, such as specific champion stats like percentages for bonus armour/magic penetration and cooldown reduction, faced potential issues with conventional normalization due to the risk of dividing by zero. To address this, the method by Műller was employed, where these features were given a standard deviation of one, ensuring every instance of this feature in the dataset had a value of zero[46].

The methodology also involved handling categorical data points. These encompassed various game elements, including in-game items, summoned spells by characters, champions with their unique abilities or skills, and special dragon types with buffs. To encapsulate the nuances of these categorical data points, fixed-size embedding vectors were designed and integrated during the learning stage of the neural networks. On the spatial aspect, employing a proven two-layer convolutional neural network was deemed the most effective method to encode map-focused features.

Game states in titles such as League of Legends (LoL) frequently face the complexity of variable data sizes. As player strategies and game dynamics evolve, champions may have a varying number of wards placed or items acquired. This inconsistency can present substantial challenges during data processing. To tackle this issue, a methodology was employed that used zero-padding on these varying sets, ensuring alignment with the game's predefined upper boundaries. An illustrative example is evident in the handling of farsight wards. While LoL doesn't strictly enforce a maximum for these wards, a conservative limit of 100 was chosen. This limit was not chosen randomly; it was considered sufficiently high to ensure players wouldn't likely exceed this number during a typical game. The metrics for other wards were set based on the specific in-game items a player might possess.

In addressing game elements such as items, wards (of all types), summoner spells, skills, and champions, a set-based representation was utilized, as implemented by [46].

Each set underwent specific processing. Initially, a two-layered, fully connected neural network, using ReLU as the non-linearity function, was applied to each item in the set, including zero-padded items. Both these layers had an identical number of neurons. After this transformation, 1-dimensional max-pooling was used to merge this set of items into a single vector representation. Notably, this process was not affected by the sequence of items within each set.

With champions, the treatment takes on a slightly different approach. Pushing the champions through the dual, fully-connected layers produces an intricate champion embedding. The updated embedding is comprehensive, drawing data from a diverse array of sources – including items, summoner spells, skills, wards, and multiple attributes specific to champions. This encompasses their current state, stats, the previously discussed Fog of War attribute, and type.

To further refine this, for every champion c participating in game x at a given time t, the state (denoted as $x_{t,c}$) exhibits slight variations. This processed champion embedding is then appended to the overarching game state prior to encoding. This ensures the network possesses the requisite context when predicting macro-level decisions tailored to specific champions. Additionally, the champion's state feature undergoes a minor tweak. The "allies" attribute is modulated depending on the current champion c under consideration.

3.2 Model recreation

With the primary objective being the analysis of the 'fog of war' (FOW) impact on game predictions, the same encoder components as those proposed by Műller [46] were implemented. This enabled a thorough comparison between the two models.

3.2.1 Macro Movement

The game map was segmented into an NxN grid structure, inspired by Ye et al.[52]. With N set to 12, this led to $N^2 = 144$ distinct sections. To maintain consistency with the model developed without the FOW data, 144 sections were retained. For each champion c at a given timestep t, the goal was to define $Y^{t,c}_{\text{pos}}(x) = i$, where i denotes the predicted location of champion c in the subsequent game state. A specialized component then computed the probability of this anticipated position using the encoded game information, emphasizing its importance as highlighted by [46].

3.2.2 Minions and Monsters

In the game mechanics, dispatching minions and neutral monsters increments the kill tally, labelled as 'creeps slain' (cs). This statistic acts as a critical gauge of a player's in-game performance. It's paramount to recognize that this metric is not valued uniformly across all roles. For instance, cs is often allocated by those in support roles to their marksman counterparts, introducing a nuance to the evaluations. [46].

To tackle this, two prediction components were incorporated. Each component aims to project the adjusted quantity of dispatched minions and monsters. Given a time horizon T and a discount factor δ, the projection equations are:

$$Y^{t,c}_{\text{minions}}(x) = \sum_{i=t+1}^{min(t+T,N)} \delta^{i-(t+1)} \text{minions}^{i,c}(x) \tag{3.2}$$

$$Y^{t,c}_{\text{monsters}}(x) = \sum_{i=t+1}^{min(t+T,N)} \delta^{i-(t+1)} \text{monsters}^{i,c}(x) \tag{3.3}$$

[46]

3.2.3 Objectives

Participation in combats and capturing objectives are crucial elements of game strategy. The model is designed to forecast the adjusted counts of champion kills, destroyed turrets, inhibitors, and significant objectives like Rift Heralds, dragons, and Baron Nashors.

For each given event type e, and for every champion c at each timestep t, excluding the final one, the prediction is formulated with a span T and a discount factor $\delta \in [0, 1]$ as:

$$Y^{t,c}_{e}(x) = \sum_{i=t+1}^{min(t+T,N)} \delta^{i-(t+1)} e^{i,c}(x) \tag{3.4}$$

Here, $e^{i,c}$ represents the occurrences of event e for champion c between timesteps $i-1$ and i. It's important to note that an event is considered to have occurred for a champion

if they were involved, irrespective of whether it was a kill or an assist. This design choice prioritizes player intentions over exact outcomes, mirroring real-world game dynamics [46].

3.2.4 Implemented multitask loss

Prediction components can be optimized concurrently with the encoder. The model seeks to adjust the network parameters θ to minimize a weighted sum of losses on the training dataset X. The loss function $L(\theta)$ can be expressed as:

$$L(\theta) = E\left[w_{\text{pos}} l_{CE}(f(x_{t,c}|\theta)_{\text{pos}}, Y^{t,c}_{\text{pos}}(x)\right] + \sum_{\text{event } e} \ldots \tag{3.5}$$

Where $f(x_{t,c}|\theta)$ signifies the network output for game state $x_{t,c}$ based on parameters θ. The loss weights w_{pos} and w_e correspond to distinct events e such as minions, monsters, kills, turrets, and more [46].

Given the unique variance of each prediction target, it's crucial to assign appropriate weights to losses. Initial evaluations indicated that significant contributions to the total loss came from positions, minions, and monsters, while less frequent events like Baron or Rift Herald had a negligible effect. As a result, the weight of the position loss w_{pos} was empirically set to 1. The weight for each event e was then calculated as the reciprocal of the variance of its corresponding prediction target:

$$w_e = \frac{1}{\text{Var}(Y^{t,c}_e)} \tag{3.6}$$

[46]

Thus, a simplistic predictor estimating the target as the mean from the training dataset would yield an average weighted loss of 1.

3.3 Model

Utilizing PyTorch as the foundational framework, the original model was crafted to predict game dynamics with a series of embeddings and processors. Hyperparameters like a discount factor of 0.7 and a time horizon that spanned 60 seconds were set, with training carried out in batches of 64 using the Adam optimizer.

Diving into the architecture, the model initially boasted convolutional layers for spatial data processing. These layers had kernels of size 3 with zero padding, accompanied by 2D max pool layers and Leaky ReLU activations. Embeddings, which are essentially compact representations of game entities like champions, items, wards, etc., had been trimmed to just 2 neurons each to prevent overfitting. Additionally, there were specific processors to manage sets of these entities, and the game state was encoded using 128 neurons.

One significant deviation from this architecture is the addition of the Fog of War feature, embodied in the map_processor. This alteration changes the way spatial data is processed, particularly how the model understands the game map, making it more attuned to in-game strategies and patterns obscured by fog.

The evolved MacroPredictor now functions with 41,044 trainable parameters, a slight uptick from the original's 40,340. To incorporate the Fog of War, the map processor employs two convolutional layers, the first with eight kernels and the subsequent with just 1 – aiming to reduce feature numbers post-processing. The intricacies of warding, crucial for strategic gameplay, have been magnified by distinguishing between three ward types - control, sight, and farsight, each with its dedicated set processor.

Despite the augmentation of the model's architecture, its complexity hasn't ballooned. However, with these enrichments, the model's training was more resource-intensive. Using the NVIDIA RTX 4080, the training spanned 6 hours, 4 minutes, 33 seconds, and 272 milliseconds.

In summation, while the backbone of the model remains faithful to its original design, revisions, especially the integration of the Fog of War, enhance its ability to interpret obscured game dynamics, ensuring predictions are strategic and informed.

3.4 Used technologies

This section is a brief overview of used libraries and technologies without which it wouldn't be possible to finish this model.

3.4.1 Python

The reason Python was picked for this project is manifold. Foremost, Python boasts a comprehensive ecosystem of specialized libraries, such as Scikit-learn, TensorFlow,

PyTorch, and Pandas, offering streamlined data manipulation, model development, and visualization solutions. This vast arsenal of tools expedites the development process and fosters swift and iterative experimentation. Moreover, Muller's previous work was also conducted in Python, and the intent to compare models necessitated a consistent platform. Python's clear and intuitive syntax fosters collaboration, ensuring seamless code-sharing and review among team members. The language's widespread adoption in both academic and commercial spheres guarantees robust community support, an indispensable asset for resolving challenges and gaining insights. Additionally, Python's flexibility, demonstrated by its ability to interface seamlessly with other languages and platforms, ensures the project's adaptability for scaling purposes or integration with other systems. In sum, the choice was anchored in Python's adaptability, rich toolset, collaborative-friendly nature, and the need for consistency with prior work.

3.4.2 Machine Learning and Data Processing

Scikit-learn (sklearn):

- **RandomForestClassifier:** Used for classification tasks employing the Random Forest algorithm, known for its robustness and handling of large datasets.
- **LogisticRegression:** Deployed for binary outcome classification tasks. A fundamental algorithm in machine learning.
- **StandardScaler:** Standardizes features by removing mean and scaling to unit variance.
- **KFold:** Implements k-fold cross-validation for robust model evaluation.
- **accuracy_score:** Gauges model accuracy.
- **train_test_split:** Splits datasets into training and test subsets.

XGBoost (xgboost): A gradient boosting library efficient for structured data.

KModes: Used for clustering categorical variables.

UMAP: Dimensionality reduction tool visualizing high-dimensional data.

3.4.3 Visualization Tools

Matplotlib (matplotlib.pyplot): A plotting library in Python for various data visualizations.

Plotly (plotly.express): Provides interactive plotting capabilities.

OpenCV (cv2): Potentially used for image processing or other vision tasks.

3.4.4 Deep Learning with PyTorch

PyTorch (torch): Deep learning framework suggesting design, training, and optimization of neural networks.

TensorBoard (torch.utils.tensorboard.SummaryWriter): Visualizes neural network training runs and metrics.

torch.utils.data: Essential tools for efficient dataset loading and batching during training.

3.4.5 Miscellaneous

Pandas (pd): Essential for data manipulation and analysis.

JSON: Handles a large amount of JSON data.

ThreadPool (multiprocessing.pool.ThreadPool): Indicates parallelized operations for performance enhancement.

Enum: Defines named enumeration sets.

4 RESULTS

This chapter presents the culmination of the thesis's research and analysis. Here, the findings are laid out, highlighting the differences between the models with and without the FOW component. Grounded in data and analytical techniques, the results aim to address the question if incorporating this feature is beneficial.

4.1 Comparative Analysis of Model Outputs

In the detailed analysis comparing the model enhanced with the 'fog of war feature to its earlier version devoid of the FOW component, there's a clear observation: the introduction of the fog of war doesn't bring about a profound shift in predicting player macro behaviours. While deviations are noted in some instances, they steer the model towards a path characterized by prudence and caution. Instead of opting for more confrontational tactics, the model, with the fog of war, seems to favour defensive manoeuvring. The strategic implications of such a cautious approach offer a rich tapestry for contemplation.

For illustrative purposes, consider the visual representation in Figure 4.3. Specific competitive matches were analysed to provide a more detailed explanation, including the clash between team FlyQuest and Team Liquid in the 2023 LCS spring split. From this match, five critical moments were selected that, based on linear regression analysis, most profoundly impacted the game's outcome. These moments are visually depicted in Figure 3.1. To achieve a deeper comprehension of these situations, expert viewpoints were solicited.

4.1.1 Profesional perspective of the outcome

Esteemed professional player, Václav "VaSKED" Benda, with nine years of experience in playing *League of Legends* on profesional level and boasting multiple championship titles, including the winner of Championship of the Czech Republic, provided insights on the outcomes from both models, particularly focusing on the five situations depicted in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4, and Figure 4.5.

According to him, the model with the Fog of War (FOW) and the one without it are very comparable in situations such as these. However, in scenarios where a team is generally losing — as exemplified in Figure 4.4 for the blue side — the model with

FOW suggests that the blue team should confront their opponents in a battle they are unlikely to win. Conversely, the alternative model recommends a strategic retreat to minimize losses. Václav Benda's professional opinion leans towards the latter strategy. This behaviour persists upon observation. Though the differences between the models might seem nuanced at first glance, the FOW model tends to favour more assertive strategies over a conservative approach.

Figure 4.1 Comparison of model with and without the fog of war FLY vs TL | Week 7 Day 1 S13 LCS Spring 2023 2:10

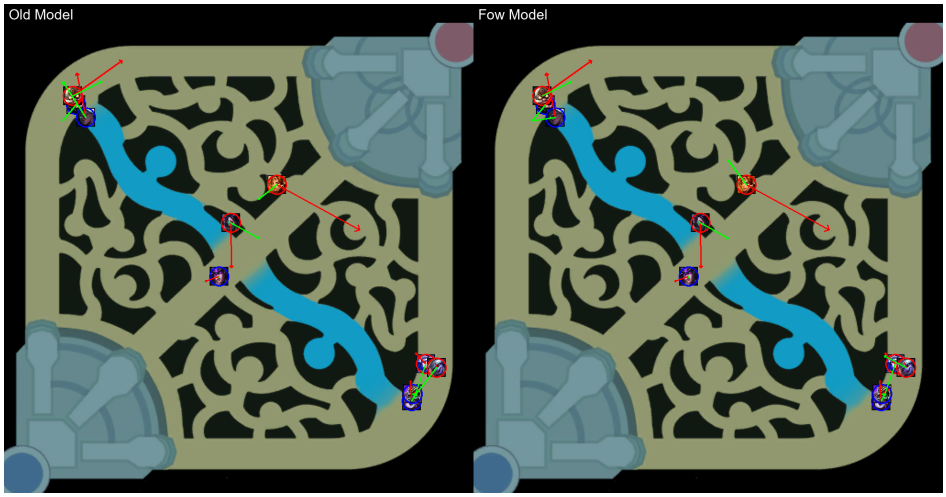

Figure 4.2 Comparison of model with and without the fog of war FLY vs TL | Week 7 Day 1 S13 LCS Spring 2023 2:25

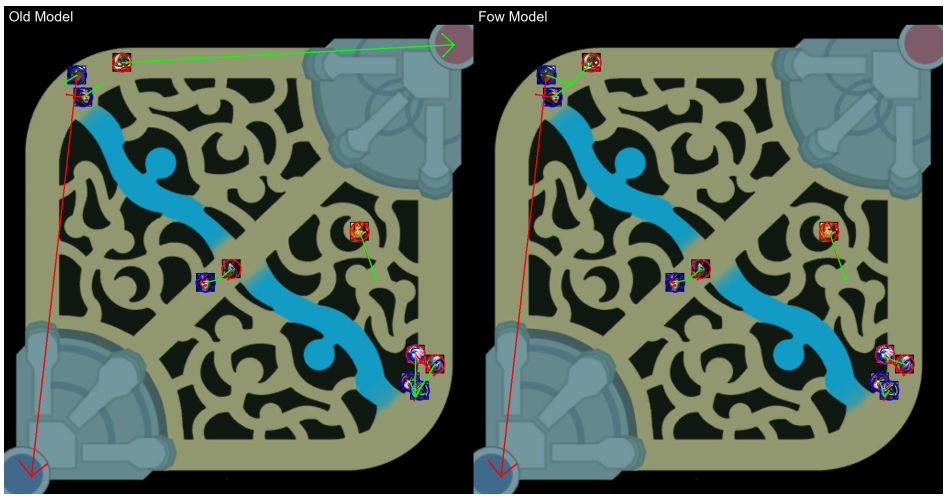

Figure 4.3 Comparison of model with and without the fog of war FLY vs TL | Week 7 Day 1 S13 LCS Spring 2023 2:35

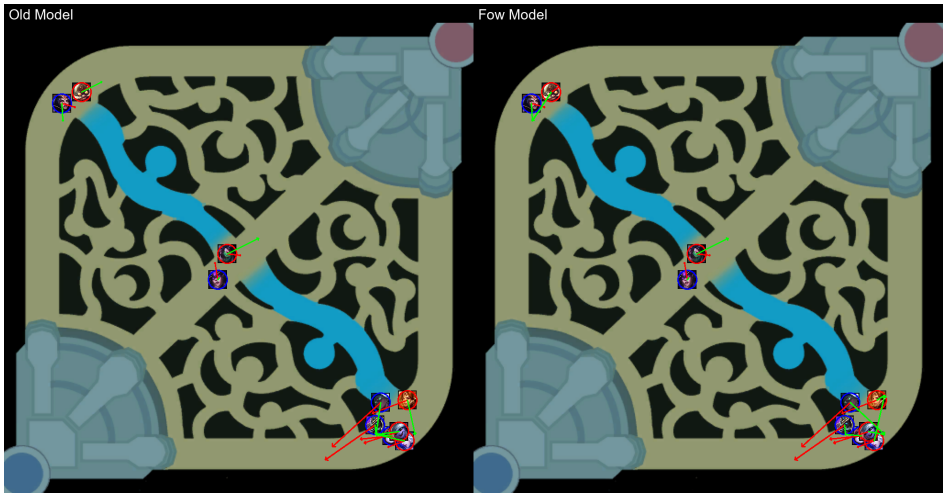

Figure 4.4 Comparison of model with and without the fog of war FLY vs TL | Week 7 Day 1 S13 LCS Spring 2023 4:20

Figure 4.5 Comparison of model with and without the fog of war FLY vs TL | Week 7 Day 1 S13 LCS Spring 2023 5:30

The figures Figure 4.1 through Figure 4.5 offer a visual representation of the predictive capabilities of both models — one with the Fog of War (FOW) and one without it. When placed side by side with the red and green lines, a clearer understanding of the disparity between actual player movements and the macro predictions of the models emerges. Here are some potential observations and interpretations based on this description:

1. **General Trends:** In many instances, both models' predicted movements (green lines) align with the actual player movements (red lines). This congruence suggests that both models have been trained sufficiently and have a good grasp of typical player movement patterns during a match.

2. **Divergences:** While many predictions mirror the actual movements, some noticeable discrepancies exist. These divergences could be the result of various factors, including:

    - The unpredictable nature of player decisions in specific match situations.
    - The inherent differences between the models, with the FOW model potentially being more aggressive or taking risks due to its limited visibility.

3. **Key Decision Points:** These variances between predictions and actual movements are especially salient at critical junctures in the game. For instance, when teams consider contesting objectives, setting up ambushes, or deciding whether to engage or retreat from battles. These moments can drastically influence the game's outcome; thus, any disparity in predictions at these points is of significant interest.

4. **Model Behavior:** If, as previously discussed, the FOW model tends to suggest more aggressive plays, it might predict movements pushing towards enemy territory or objectives even in situations where players are usually more cautious. On the other hand, the non-FOW model might predict more defensive or standard movements, mirroring common professional strategies.

In conclusion, these visual comparisons underline the complexities of modeling player behaviour in dynamic games like League of Legends. They show the strengths and limitations of both models, highlighting the potential value and pitfalls of incorporating features like the Fog of War. Refining the models based on feedback from professional players (like VaSKED's input) and deeper analyses of these divergence points could benefit future improvements.

The results of a comprehensive analysis indicate that the introduction of the "Fog of War" element, when contrasted with its absence, does not substantially impact the study's outcomes. Moreover, given the additional computational resources needed to integrate this feature, its inclusion is less justifiable. Marginal variations, quantified as mere tenths of percentage points between scenarios with and without the "Fog of War", do not seem to warrant the augmented processing costs from an operational standpoint.

In the future, should a dataset encompassing the "Fog of War" feature become available, allowing for its straightforward incorporation without the burden of prohibitive realization costs, a re-evaluation of its potential integration in subsequent research endeavours might be considered.

CONCLUSION

In this thesis, statistical models were developed for League of Legends, integrating the Fog of War feature and updating them to the latest available patch, 13.4, at the time of thesis completion. These models were demonstrated for the "Detection of Team Synergies based on Individual Game Playstyle". Supervised learning tasks were formulated, and necessary datasets for win prediction and macro decision prediction were generated from raw data. These datasets served as the foundation for training and evaluating the models.

A logistic regression model was included to predict the likelihood of winning the game. This model provided insights into the influence of individual game events on game outcomes.

Results indicated that the Fog of War feature did not significantly enhance the predictive outcome to warrant extensive processing resources for its acquisition. Moreover, insights were offered on how players might identify their macro-level errors and gain a deeper understanding of their mistakes. Although MOBA games, predominantly League of Legends and DOTA 2, were detailed, other game genres such as F1 2022 racing game and shooter game CS:GO were also discussed. It is believed that adapting this model to another MOBA game like DOTA 2 would be straightforward. However, genres lacking features like Fog of War might necessitate a reconfiguration of the model or would be better suited with a specialized model.

For subsequent studies, it is advised that these models be tested with games and players beyond the professional domain. Moreover, the incorporation of more extensive datasets could potentially refine the model's efficacy. Additionally, it is proposed that future studies undertake a comparative evaluation of models across different regions and gameplay levels.

REFERENCES

[1] Dalgaard, M.; Motzoi, F.; Sørensen, J. J.; et al.: Global optimization of quantum dynamics with AlphaZero deep exploration. *npj Quantum Information*, volume 6, no. 1, January 2020, doi:10.1038/s41534-019-0241-0.
URL `https://doi.org/10.1038/s41534-019-0241-0`

[2] Stockfish 15.1.
URL `https://stockfishchess.org/`

[3] Crandall, J. W.; Oudah, M.; Chenlinangjia, T.; et al.: Cooperating with machines. *Nature Communications*, volume 9, 2017.

[4] Britannica, E.: Chess and Artificial Intelligence. 2021, accessed: April 15, 2023.
URL `https://www.britannica.com/topic/chess/Chess-and-artificial-intelligence`

[5] Greenemeier, L.: 20 years after Deep Blue: How Ai has advanced since Conquering Chess. Jun 2017.
URL `https://www.scientificamerican.com/article/20-years-after-deep-blue-how-ai-has-advanced-since-conquering-chess/`

[6] Bouzy, B.: History and territory heuristics for Monte-Carlo Go. *New Mathematics and Natural Computation (NMNC)*, volume 02, 07 2006: pp. 139–146, doi:10.1142/S1793005706000427.

[7] Manyika, J.; Bughin, J.: The promise and challenge of the age of Artificial Intelligence. 2018.
URL `https://www.mckinsey.com/~/media/McKinsey/Featured%20Insights/Artificial%20Intelligence/The%20promise%20and%20challenge%20of%20the%20age%20of%20artificial%20intelligence/The-promise-and-challenge-of-the-age-of-artificial-intelligence-In-Brief.ashx`

[8] Girgis, A.: Evaluating the Efficacy of Deep Neural Networks in Reinforcement Learning Problems. *American Scientific Research Journal for Engineering, Technology, and Sciences*, volume 46, 2018: pp. 260–272.

[9] Berner, C.; Brockman, G.; Chan, B.; et al.: Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[10] Champions - League of Legends.
URL `https://www.leagueoflegends.com/en-us/champions/`

[11] Minion (League of Legends).
URL https://leagueoflegends.fandom.com/wiki/Minion_(League_of_Legends)

[12] League of Legends Wiki: Blue Sentinel. 2023.
URL https://leagueoflegends.fandom.com/wiki/Blue_Sentinel

[13] League of Legends Wiki: Red Brambleback. 2023.
URL https://leagueoflegends.fandom.com/wiki/Red_Brambleback

[14] League of Legends Wiki: Rift Herald (League of Legends). 2023.
URL https://leagueoflegends.fandom.com/wiki/Rift_Herald_(League_of_Legends)

[15] League of Legends Wiki: Baron Nashor (League of Legends). 2023.
URL https://leagueoflegends.fandom.com/wiki/Baron_Nashor_(League_of_Legends)

[16] League of Legends Wiki: Chemtech Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Vigilant_Wardstone

[17] League of Legends Wiki: Cloud Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Cloud_Drake

[18] League of Legends Wiki: Hextech Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Hextech_Drake

[19] League of Legends Wiki: Infernal Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Infernal_Drake

[20] League of Legends Wiki: Mountain Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Mountain_Drake

[21] League of Legends Wiki: Ocean Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Ocean_Drake

[22] League of Legends Wiki: Elder Dragon. 2023.
URL https://leagueoflegends.fandom.com/wiki/Elder_Dragon

[23] League of Legends Wiki: Dragon Slayer - Aspect of the Dragon. 2023.
URL https://leagueoflegends.fandom.com/wiki/Dragon_Slayer#Aspect_of_the_Dragon

[24] League of Legends Wiki: Nexus. 2023.
URL https://leagueoflegends.fandom.com/wiki/Nexus

[25] League of Legends Wiki: Turret. 2023.
URL https://leagueoflegends.fandom.com/wiki/Turret

[26] League of Legends Wiki: Inhibitor. 2023.
URL https://leagueoflegends.fandom.com/wiki/Inhibitor

[27] Dota 2 Heroes. https://www.dota2.com/heroes, accessed: 2023-05-21.

[28] Fandom, D. .: Game Modes - Ranked All Pick. 2023, accessed: 2023-05-21.
URL https://dota2.fandom.com/wiki/Game_modes#Ranked_All_Pick

[29] Çakır, G.: How to lane in Dota 2. Mar 2021.
URL https://dotesports.com/dota-2/news/how-to-lane-in-dota-2

[30] Mr.Nobody: Dota 2 Lane Information. https://steamcommunity.com/sharedfiles/filedetails/?id=261299172, 2014, accessed: 2023-05-21.

[31] zlosterr: Dota 2: How to play mid on any patch. Nov 2022.
URL https://blix.gg/news/dota-2-how-to-play-mid-on-any-patch

[32] DMarket: Dota 2 Positions and Roles Explained. Nov 2022.
URL https://dmarket.com/blog/dota-2-positions/

[33] Ignatov, V.: Dota 2 - What not to do in the ultra late game. 2020.
URL https://weplay.tv/news/dota-2-what-not-to-do-in-the-ultra-late-game-20049

[34] Wiki, D. .: Game modes. 2021, accessed: 2023-05-21.
URL https://dota2.fandom.com/wiki/Game_modes

[35] Trahan, P.: How many people play CSGO? Player count in 2023. May 2023.
URL https://www.dexerto.com/csgo/how-many-people-play-csgo-player-count-record-2071859/

[36] Counter-Strike: Global Offensive. Accessed: 2023-05-21.
URL https://counterstrike.fandom.com/wiki/Counter-Strike:_Global_Offensive

[37] https://counterstrike.fandom.com/wiki/Competitive. Accessed: 2023-05-21.
URL https://counterstrike.fandom.com/wiki/Counter-Strike:_Global_Offensive

[38] Leetify: Understanding CS:GO Economy. Blog post, August 2021.
URL https://blog.leetify.com/understanding-csgo-economy/

[39] Karakurt, E.; Willatzen, J.: Competitive Level Design: A study on Counter-Strike: Global Offensive Level Design. May 2020, program: Game Design & Programming.
URL https://www.diva-portal.org/smash/get/diva2:1453749/FULLTEXT01.pdf

[40] Durant, T.: F1 2022 Game: Release date, new cars, graphics, gameplay trailer, PS5, Xbox, PC, career mode, My Team, multiplayer & more. 2023.
URL https://racinggames.gg/f1/f1-2022-game-release-date-new-cars-graphics-gameplay-trailer-ps5-xbox-pc-career-mode-my-team-multiplayer-more/

[41] Unknown: Are F1 Esports drivers quicker than real Formula 1? 2023.
URL https://www.overtake.gg/opinion/feature/are-f1-esports-drivers-quicker-than-real-formula-1/

[42] Hislop, M.; Sivanathan, A.; Lim, T.; et al.: Lecture Notes in Computer Science: Beyond simulators, Using F1 Games to Predict Driver Performance, Learning and Potential. 2013: pp. 157–171, doi:10.1007/978-3-319-12157-4_13.

[43] Ferrucci, D.; Levas, A.; Bagchi, S.; et al.: Watson: Beyond Jeopardy! *Artificial Intelligence*, volume 199, 2013: pp. 93–105.

[44] Silver, D.; Huang, A.; Maddison, C. J.; et al.: Mastering the game of Go with deep neural networks and tree search. *Nature*, volume 529, no. 7587, 2016: pp. 484–489.

[45] Brown, N.; Sandholm, T.: Superhuman AI for multiplayer poker. *Science*, volume 365, no. 6456, 2019: pp. 885–890.

[46] Štěpán Műller: *Detection of disadvantageous individual decisions for a game with fantastic elements*. Master thesis, Czech Technical University in Prague. Computing and Information Centre., 06 2022.

[47] Reinboom, R.: Minion AI Rules Documentation. 2019, accessed on 5 9, 2023.
URL http://web.archive.org/web/20200308182225/https://boards.na.leagueoflegends.com/en/c/developer-corner/qRHotV9k-minion-ai-rules-documentation

[48] League of Legends Wiki: Ocean Drake. 2023.
URL https://leagueoflegends.fandom.com/wiki/Ward#Totem_Ward

[49] League of Legends Wiki: Sources of Sight. 2023.
URL `https://leagueoflegends.fandom.com/wiki/Sight#Sources_of_sight`

[50] League of Legends Wiki: Vigilant Wardstone. 2023.
URL `https://leagueoflegends.fandom.com/wiki/Chemtech_Drake`

[51] Community, R. A.: Data Dragon | Riot API Libraries. 2023, accessed: 2023-08-01.
URL `https://riot-api-libraries.readthedocs.io/en/latest/ddragon.html`

[52] Ye, D.; Chen, G.; Zhao, P.; et al.: Supervised Learning Achieves Human-Level Performance in MOBA Games: A Case Study of Honor of Kings. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[53] Maymin, P.: Smart kills and worthless deaths: eSports analytics for League of Legends. *Journal of Quantitative Analysis in Sports*, volume 17, no. 1, 2021: pp. 11–27.

[54] Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, volume 2, no. 3, 1998: pp. 283–304.

LIST OF ABBREVIATIONS

A*	Pathfinding and graph traversal algorithm
AD	Attack Damage
AI	Artificial Intelligence
AP	Ability Power
ARAM	All Random All Mid
CD	Cooldown
CS	Creep Score
CT	Counter-Terrorist team in CS:GO
T	Terrorist team in CS:GO
CS:GO	Counter-Strike: Global Offensive
DOTA	Defense of the Ancients
F1	Formula 1
FLY	FlyQuest (eSports team)
ID	Identifier
IBM	International Business Machines Corporation
JSON	JavaScript Object Notation
K-fold	K-fold Cross Validation
LCS	League of Legends Championship Series
LoL	League of Legends
Lstm	Long Short-Term Memory
ML	Machine Learning
MOBA	Multiplayer Online Battle Arena
Ms	Milliseconds
NNUE	Efficiently Updateable Neural Network
NPC	Non-Player Character
Pd	Pandas
Relu	Rectified Linear Unit
RTX	Ray Tracing technology by NVIDIA
TL	Team Liquid (eSports team)
UMAP	Uniform Manifold Approximation and Projection

LIST OF FIGURES

LIST OF TABLES