Problem Solving in the Game of Go

Student ID: 220026574

Caterina Esposito

August 15th, 2023

Supervisor: Edwin Brady

Word Count: 12,253

Table of Contents

Abstract

With a history spanning over 2,500 years, *Go* is a game that has captured the curiosity of players worldwide due to its profound strategic complexity. This game entails players strategically positioning black and white stones at intersections on a 19x19 grid, forming connections and asserting control over territory on the square board. This dissertation aims to elevate the game of *Go* through the integration of AI, providing players with insightful guidance and recommendations. The paper's overarching goal is the development of an AI system capable of scrutinizing the game's state and delivering precise, relevant advice on stone placement. To objectively assess its performance, the Minimax algorithm and comprehensive testing was implemented from simulated games and interactions with the AI. This data-driven approach allowed for a detailed analysis of the AI's decision-making patterns, performance measurement against different opponents, and identification of areas for improvement. Through data analysis, valuable insights into the AI's strengths and vulnerabilities were gained. Modifications were implemented to amplify the overall gameplay experience. The experimentation and data aggregation played a pivotal role in refining the AI's performance, enabling it to furnish players with accurate and invaluable suggestions for moves. This, in turn, aids players in honing their skills and deepening their understanding of the intricate game of *Go*.

Introduction

Go is a game with a more than 2,500-year history that has captivated players all over the world with its astounding strategic depth and intricacy. Players in the game of *Go* must carefully place black and white stones on the intersections to construct links and control territory on a square board with a 19x19 grid. *Go* offers a wide variety of possible board layouts and strategic options despite having what appear to be straightforward rules, making it a hard and thought-stimulating game. Modern developments in artificial intelligence (AI) technology have transformed several industries, including board games. With the use of sophisticated algorithms and computer capacity, AI systems have surpassed human skill and achieved unparalleled success in games like *Go* and Chess. This development has created new opportunities for enhancing games and giving players insightful information.

The goal of this dissertation is to use AI to improve the game of *Go* by offering players helpful tips and advice. The purpose is to create an AI system that can analyze the status of the game and make precise and pertinent recommendations about where to lay stones. Players, especially newcomers and those looking to improve their skills, can gain from intelligent coaching and strategic suggestions that may otherwise prove elusive by using cutting-edge AI algorithms and methods. By incorporating AI into the game of *Go*, experts seek to improve players' ability to make decisions, increase their comprehension of strategic concepts, and increase their enjoyment of the game. Players can investigate new strategic choices, predict opponents' moves, and create ever-more complex plans by obtaining real-time clues and guidance. The research will examine the design, application, and evaluation of a *Go* hint system powered by AI to accomplish this goal. Diverse AI methods, such as the assessment of strategic implications, considerations of area control, and detection of capturing possibilities, will all be covered by the inquiry.

The Monte Carlo Tree Search (MCTS), a well-known AI method that will be examined, has proven to perform exceptionally well in several board games. The usefulness of the well-known Minimax method in *Go* will also be taken into consideration. The created AI system will offer beginner players tools on learning the basics of *Go*. This is a learners guide on the Japanese way of playing *Go*, since there are many ways of playing the game. The AI system will also offer players precise and contextually pertinent guidance suited to their unique skill levels, enabling a fluid and highly responsive gaming experience. Players will be able to make better judgments, develop their ability to think strategically, and eventually increase their playing ability with the help of the AI system's tips and direction. In the following sections of this dissertation, a thorough literature review will be given. It will look at current results and thoughts regarding the use of AI approaches in *Go* game development.

The review addresses the evolution of AI in *Go* board games, examining the many AI strategies used in *Go*, and highlighting the most significant developments in AI algorithms, such as the MCTS and Minimax. AlphaGo, which is an AI program that has advanced the player movement of winning the game, will also be discussed for completeness. The development process for the

AI-powered hint system will be described in full in the methodology chapter, along with the information structures, algorithms, and methods used. The technical details of converting the AI algorithms into a working system within the *Go* game will be covered in detail in the implementation chapter. The conclusion will include a summary of the results and suggestions for further research.

Literature Review

2.1 History of Go

The board game *Go* was created in China more than 2,500 years ago; it is additionally referred to as Igo in Japan and Baduk in Korea. Although the exact origins of it are still obscured by mythology and tradition, historical records show that Chinese aristocracy, scholars, and soldiers enjoyed playing the game throughout various dynasties [1].

Go's origins can be dated to the Zhou era, but the Tsin dynasty is when the game really started to gain in popularity. Instead of being only a pastime, playing *Go* was of such importance that disputes would occasionally be settled over a round of the game. *Go* was first a game only available to the nobles until it was introduced to Japan in 754 AD by the Japanese Ambassador Kibidaijin. A century later, however, after the realization of a Japanese prince who sought learning from the Chinese Go master Koshi-gen, the game of *Go* saw a surge in popularity [2]. By the 13th century, the game of *Go* had become popular among warriors as it had permeated every level of society. After fighting, troops would play tranquil games of *Go*, which helped the game become more well-known and spread to the wealthy class and then to the general public. During the 16th and 17th centuries, the game underwent a crucial development, with monks, writers, and businesspeople becoming famous for their proficiency at the game. These people were frequently invited to the Daimios courts where they would play games or compete against one another [2].

The founding of a national Go academy in Japan during the rule of Tokugawa Iyeyasu was a crucial turning point in institutional development. The job of principal of this academy, which was essential in bringing the game to a professional level, was given to Honinbo Sancha, a former monk and outstanding *Go* player. Sancha was also behind the creation of a rating system, a system that is still used today to classify professional masters and the best amateurs. By offering a uniform measurement of a player's strength, this approach promotes fair play [2].

Go is very well-liked and has a large player base in the modern world. Modern tournaments, which can run for many hours, highlight both the tactical complexity of the game and the requirement for physical endurance. Leading masters started publishing books with their ideas as the body of literature around the game grew, significantly enhancing the strategic tapestry of *Go*. The birth of the digital age brought us a new frontier for *Go*: the world of "computer Go," even as the conventional game maintained its rich evolution. Even though they were simple at first, early algorithms set the stage for later ones. Significant turning points were the introduction of online *Go* platforms and more sophisticated algorithms[8]. However, it was the success of DeepMind's AlphaGo, which defeated some of the best *Go* players in the world and showed how powerful AI is, that really broke down barriers. Its victories over some of the best *Go* players in the world represented a turning point and showed the enormous potential of artificial intelligence in learning the intricate rules of this centuries-old game.

While a substantial portion of recent work has carefully examined AlphaGo's tactics and accomplishments, a deeper examination of the effects of computer *Go* on traditional *Go* groups has mostly gone largely unexplored. Further academic research is needed into many facets of this interaction between traditional *Go* principles and modern computing skill, especially the teaching impacts to new players.

2.2 Computer Go

2.2.1: The Genesis of Computer Go

Computer *Go* was first developed in the 1960s, and since then the field has been continuously developing. The sophisticated and difficult game of, with its vast potential, presented a significant challenge to early AI researchers. The first attempts to develop *Go*-playing programs were made during this decade, with D. Lefkovitz credited for one of the earliest *Go* programs. Due to the complexity of the game and the limited computer resources, the development was originally slow. An important turning point occurred in 1963 with the publication of the first academic work on Computer *Go*, which unlocked the possibility of using machine learning strategies to enhance *Go* program performance.

When Edward Zobrist's computer program defeated a human player in the 1970s, Computer *Go* made a significant advancement. Despite the fact that the human opponent was a novice, this achievement was important for the developing field of AI. Based on a potential function that roughly approximated the influence of stones on the board, Zobrist's software was developed. Additionally, he developed a powerful hashing technique that was eventually used in many Computer *Go* strategies [3].

2.2.2 Tree Search and its Importance in Computer Go

The Tree Search (TS) is one of the most useful algorithms in Computer *Go*. Although many algorithms, such as Minimax and Alpha-Beta, for example, have used more contemporary techniques, the conspiracy numbers approach and proof-number search have lately gained popularity. Because of the focus on locality in the game of *Go*, TS is only used in this context. Handtalk, Many Faces of *Go*, Go4++, Go Intellect, and GoAhead are just a few examples of contemporary Go apps that use TS. However, employing local TS has difficulties, in particular defining the 'distance' to distinguish between a 'near' and 'far' motion and figuring out when to stop the local TS. There are unique challenges for Minimax search in *Go* since, unlike Chess, the objective of TS in *Go* is to assess the value of a scenario rather than choose a move [3].

2.2.3 Computer Go: Challenges and Insights

The use of artificial intelligence in gameplay has proved to be successful in a number of fields. However, there are particular difficulties in the game of *Go*. *Go* differs from other games because of its intricacy and strategic components, as Müller's thorough analysis of Computer *Go* indicates. The ability of *Go* systems still falls well short of human skill, despite considerable developments in AI and machine learning. Strong *Go* programs often perform at amateur human players' levels, a discrepancy that highlights the subtle intricacy of *Go* [4].

Go has challenges that extend beyond the game's big search area. *Go* maintains its significant complexity even on smaller 9x9 boards with a chess-like branching factor. This mainly pertains to static position evaluation, where *Go* necessitates a great deal of local tactical searching for a correct full-board evaluation. This feature emphasizes the game's innate complexity and depth of strategy further.

A proficient Go game requires players to search with a specific goal in mind. Single-goal and multiple-goal searches are distinguished by Müller. Single-goal searches concentrate on certain tactical elements, but multi-goal searches integrate fundamental aims to accomplish higher-level objectives. A more strategic view of the game is reflected in this strategy.

In contrast to previous games, Computer Go lacks a full-board Minimax search strategy for standardized decision-making. Due to this absence, a combination of search-and knowledge-based methodologies has been developed, enabling the creation of distinctive game-specific strategies. Müller emphasizes the creation of specialized techniques to address Go subproblems in more detail. The 'Life and Death' analysis is one such example. It is used to improve the accuracy of heuristics or to find exact solutions. Integrating these solutions into entire Go systems, which frequently exist as isolated modules, is a significant difficulty.

Müller outlines prospective areas for more investigation. The development of search-bound Go programs, thorough local analyses, the use of threats and forcing moves, the construction of reusable source code libraries, the achievement of sure-win scenarios for high handicaps, the incorporation of exact modules into heuristic programs, and the solution of small boards are a few of these [4]. For researchers studying artificial intelligence and game developers, Computer Go continues to be a crucial area. Significant improvements in software engineering are required due to the complexity and interdependence of Go's components. Despite the enormous obstacles, Computer Go's continuous investigation represents the never-ending quest to master this age-old game while also pushing the envelope of artificial intelligence and machine learning.

2.2.4 The Advent of Monte Carlo Go

Monte Carlo methods, widely used in theoretical physics, found a place in Go in the form of Monte Carlo Go. These techniques operate on the variational principle, which states that a system should minimize a quantity, such as energy or activity. Simulated annealing is a method that can be used to solve combinatorial optimization issues. It was inspired by the method of cooling metals to a crystalline structure that reduces energy. The development of Computer Go has been greatly aided by this global minimization strategy, which uses moves applied haphazardly in the state space [4].

2.2.5 A Historical Overview of AlphaGo in Computer Go

A significant advancement in artificial intelligence (AI) was made with the release of AlphaGo by Google DeepMind. Its victory over human Go champion Lee Sedol in a match in March 2016 established a new standard for computer go. The beginning of this domain's development can be dated to 1998, when computer Go programs first came to light. However, it was AlphaGo's victory that has since been used as the benchmark for how computational and human intelligence interact in the challenging board game of Go.

The tournaments hosted by the IEEE Computational Intelligence Society (CIS) helped Computer Go advance. These yearly competitions, which were started in 2009, served as a drive for innovation and the development of computer Go programs. The best brains in the industry could demonstrate and test their innovations at these gatherings, expanding the body of knowledge and technology in the industry. AlphaGo is the finest of the wide range of computer Go programs. Its outstanding powers can be credited to its creative application of deep convolutional neural networks (DCNNs), minorization-maximization (MM), and Monte Carlo tree search (MCTS). The program's architecture integrated these technologies, setting the stage for a novel AI strategy to address Go's complexity [5].

2.2.6 Unraveling AlphaGo's Success

The key to AlphaGo's success was its groundbreaking fusion of superior neural networks. The value network, the reinforcement learning (RL) policy network, and the supervised learning (SL) policy network are the three main networks that DeepMind implemented. The core of AlphaGo's intelligence was built through the meticulous and thorough training of these networks. Along with the networks, a sizable number of self-play game positions were developed. These positions supplemented AlphaGo's training by giving it a wide range of facts from which to draw and from which to modify its tactics. The ability of AlphaGo to analyze game situations and choose the best course of action was boosted by this iterative learning strategy, which ultimately contributed to its win [5].

2.2.7 The Dynamic Between Human and Computer Go

The intricate complexity of the game of Go, as well as the high levels of strategic planning, ingenuity, and intuition required of its players, make it unique. The paradigm of how we view AI's potential was altered by AlphaGo's victory over well-known professional Go player Lee Sedol. The professional Go players, AI researchers, and the general public all started talking about and debating this victory. The match was a momentous event that changed how Go players in Korea, Taiwan, Japan, and China viewed artificial intelligence (AI) and its role in their traditional game [5].

2.2.8 Future Prospects for Computer Go

The success of AlphaGo, a significant development in artificial intelligence, demonstrated the potential of computational intelligence techniques for mastering challenging tasks. However, the goal of future Computer Go research should be to outperform AlphaGo rather than simply duplicate it. AlphaGo performed admirably, but it's important to recognize that computer programs and human players have fundamentally different playing styles. For further investigation and study, this distinction offers an intriguing possibility. It might pave the way for breakthroughs that go beyond the boundaries of the game of Go by fostering a greater understanding of artificial intelligence and its interaction with human intellect [5].

2.2.9 Deep Learning in Computer Go

The developments in Computer Go that are linked to the improvements in deep learning techniques are covered in this section. *2.2.9.1 Introduction to Deep Neural Networks in Go*

Enhancing a variety of fields, such as visual recognition and gaming, has been made possible in large part by deep learning, particularly deep convolutional neural networks. Deep neural networks were used for the first time to play the game of Go. Deep neural networks and Monte Carlo tree search were used in a novel way to improve Go performance [6].

2.2.9.2 The Structure of AlphaGo's Neural Networks

How did AlphaGo work?

The two main neural networks used by the AlphaGo system are:

Value Networks: Used to evaluate board positions.

Policy Networks: Used to predict and select moves.

Both networks underwent extensive training utilizing a blend of supervised learning from games played by human experts and reinforcement learning from games played by the networks themselves. Because Go is highly complicated, AlphaGo employed these networks to shrewdly narrow the search space rather than trying every potential move, which would be computationally impossible.

2.2.9.3 Learning and Evaluation

Silver et al. described the exact training regimen used by AlphaGo. A policy network was initially trained using supervised learning (SL) using expert human movements. Then, an enhanced version was trained utilizing self-playing games and reinforcement learning (RL). The outcome of games played by the RL policy network was then predicted using a value network. AlphaGo demonstrated a victory rate of 99.8% versus other Go systems in internal tournaments, demonstrating its skill.

2.2.9.4 Deep Learning for Go Strategy Innovation

The majority of traditional *Go* programs used Monte Carlo Tree Search (MCTS), which used random simulations to evaluate probable moves. Although this strategy gave computers the ability to play at an amateur level, it proved inadequate when playing against experts. Incorporating the knowledge from the value and policy networks into MCTS was how AlphaGo innovated. As a result, the AI could concentrate on prospective plays and assess them thoroughly and accurately for the first time[6].

In the realm of perfect information games, the optimal value function, $v^*(s)$, is an essential determinant of the outcome of the game from any given position or board state. A standard approach to deciphering these games involves recursively calculating this optimal value in the search tree. However, exhaustive search methods quickly become untenable when applied to open-ended games such as Chess and *Go*. The depth of the search can be reduced by evaluating the position, truncating the search tree, and replacing the underlying subtree with an approximate value function. This technique has proven to be effective in games like chess and checkers. On the other hand, the breadth of the search can be contracted by adopting policies, like $p(a|s)$, to sample actions, thereby narrowing the scope of potential moves. Monte Carlo rollouts do this principle by providing a stable position evaluation, culminating in search performances in games such as Backgammon. Furthermore, Monte Carlo Tree Search (MCTS) has come to the fore by leveraging Monte Carlo rollouts to appraise the value of each state within a search tree. Present-day Go algorithms predominantly harness MCTS in tandem with policies tailored to anticipate human moves, yielding impressive outcomes at an amateur level. The recent rise of deep convolutional neural networks in tasks like image classification and face recognition has also made its mark in the domain of *Go* [6]. By interpreting a 19x19 board as an input, these networks employ convolutional layers to architect a representation of the game`s position. These innovative networks have been instrumental in truncating the depth and breadth of the search tree by evaluating positions through a value network and sampling actions via a policy network. Significantly, AlphaGo emerges as a paragon in this domain, adeptly amalgamating the policy and value networks with MCTS.

2.2.9.5 The Landmark Match: Human vs. Computer

The distributed version of AlphaGo faced off against European Go champion Fan Hui in a historic matchup in 2015. Fan Hui was thoroughly defeated by AlphaGo in all five games, earning a resounding victory. This game was the first time a computer Go program was successful in defeating a skilled human player in a complete game of Go.

2.2.9.6 Deep Learning in Go: Implications and Future

The accomplishment of AlphaGo is significant not only for the field of computer go, but also for the greater AI community. AlphaGo placed a strong emphasis on cognitive position evaluation, similar to human strategy, in contrast to its forerunners, which focused on brute force computations. The methodologies bring in a hopeful era when comparable approaches might be able to resolve other difficult AI problems, in particular the coupling of deep learning with tree search.

Methodology

The AI helper for the *Go* game was created with the primary goal of assisting players in capturing opponent stones and carefully positioning their stones to restrict the opponent's liberties. The purpose of this project is to develop a user-friendly and easily accessible learning tool for Go beginners. The AI helper is intended to be a console-based application that lets users communicate with the AI using a straightforward menu structure. The AI can consider two steps ahead when offering capture techniques according to the Minimax algorithm, which improves the player's strategic thinking and learning process.

2.3 The Rules of Go [7]

How to Play Go: A Basic Guide

- Objective: The goal of Go is to control more territory than your opponent by strategically placing stones on the board.
- Board: Go is typically played on a 19x19 grid, although beginners may start on smaller boards like 9x9 or 13x13 to learn the basics.
- Game basics:
    - Players take turns placing their stones on the intersections of the lines.
    - Stones: We are using a 9x9 board.
    - There are two types of stones: black and white.
    - Black plays first, and then turns alternate.

Gameplay:

1. Opening: The game usually starts with an empty board. Players take turns placing one stone at a time on any unoccupied intersection.
2. Capturing: When a stone or group of stones becomes surrounded by the opponent's stones, it is captured and removed from the board. Stones are captured when they no longer have any liberties (empty adjacent intersections).
3. Liberties: A single stone or a group of stones must have at least one liberty to remain on the board. Liberties are empty adjacent intersections. A group of stones shares its liberties.
4. Ko Rule: The Ko rule is in effect to prevent an endless repetition of moves. If a move creates the same board position as the previous turn, it is illegal.

5. Passing: Instead of placing a stone on their turn, a player may choose to pass. If both players pass consecutively, the game ends, and territory is counted.
6. Eyes: refer to an empty space that is completely surrounded by the stones of the same color. Making at least two separate eyes inside a gathering of stones guarantees safety from capturing
7. Capturing Stones: When a stone or group of stones is entirely surrounded by the opponent's stones, it is captured and removed from the board.
8. Connecting Stones: To prevent your stones from getting captured, you should try to connect them. Connecting stones form a group, and as long as the group has at least one liberty (empty adjacent intersection), it remains on the board.
9. Creating Liberties: During the game, players extend their influence by creating groups with multiple liberties. More liberties make it harder for the opponent to capture those stones.
10. Capturing Races: Sometimes, capturing stones can lead to tactical races where players attempt to capture each other's stones in a sequence of moves. These situations can be quite exciting and require careful calculation.
11. Sacrifice Stones: In some cases, sacrificing a stone or a group of stones strategically can help you gain a better position elsewhere on the board.

2.3.1 Game Design and Objectives

The *Go* game AIHelper is not a complete *Go* game, but rather a customized tool that focuses on particular game elements. The AI helper's primary goals are to teach players efficient capture tactics and to give players insightful information on where to set stones to restrict the opponent's freedoms. The goal of the game's design is to give new players a welcoming setting in which they may learn the strategic aspects of Go.

2.3.2 User Interaction and AI Help Menu

The AI helper has a console-based design and an easy-to-use menu system to ensure a user-friendly experience. When the game first launches, the user is given access to the AI help menu, which has three primary options to choose from:

1. Move Suggestion: The Move Suggestion option makes use of the AI helper method to suggest the optimal move to the player based on the situation of the board. It provides helpful advice for making strategic decisions and suggests the best stone placements to take enemy stones and gain the upper hand.

2. Capture Help: The provideCaptureHelpAI method is used by the Capture Help option to help players determine which opponent stones they can successfully capture. The AI considers future actions that can restrict the opponent's liberty,while also providing the most effective strategy for capturing the stones.

3. Quit AI Help: Users can choose to leave the AI helper menu, which will let them resume their game.

2.3.3 Board Setup and Pattern Selection

The standard 9x9 grid that makes up the *Go* game board provides the best compromise between playability and complexity for new players. For players' convenience, white and black stones are represented in this console by the letters "X" and "O". Users' ability to enter unique board layouts is a feature. Due to the flexibility, players can experiment with various configurations to either try new approaches or replicate particular game situations. For players who are interested in trying out different strategies or reliving specific game moments, such a free-play mode is essential.

Additionally, there are available predefined patterns. These designed situations are meant to improve a player's capturing strategies. Players are forced to strategize and adapt as a result of the patterns' presentation of these particular game circumstances, which helps players improve their performance. The combination of adaptability and well-known patterns makes sure that players have a thorough understanding of *Go*'s complexities. An option to randomly choose from the predetermined patterns is introduced to reduce predictability and make sure players aren't overly dependent on familiar patterns. Each game will present a different challenge because of the randomness of the selection, which encourages players to think quickly and keeps them engaged.

2.3.4 Comparative Analysis and a Balancing Strategy

An in-depth analysis was performed in order to develop an AI helper aimed to help *Go* beginners. This analysis explored a variety of *Go* game patterns, with particular focus on well-known strategies such as the "Eye" formation. For instance, the "Eye" formation is a simple but *Go* strategy that can offer stones a territorial advantage and possibly safety. For beginners, understanding these fundamental patterns is essential since it gives them a base from which to create their gaming.

The analysis showed that the Japanese game of *Go* would be the best example of the AI helper. The Japanese style is known for emphasizing the game's fundamental concepts while balancing simplicity and strategic depth, providing a clear access point for beginners. The choice of the Japanese aesthetic was not just driven by its simplicity. It is distinguished for its capacity to convey the complex techniques of Go without being intimidating to beginners. This guarantees that new players simultaneously are given a brief introduction to the game's fundamentals, receive an understanding of advanced strategies, and continue groundwork for their ongoing learning.

2.3.5 Adjustments to the Traditional Go Gameplay

Several changes were made to the standard *Go* rules in order to adapt the game to new players and promote a positive learning setting

2.3.6 Absence of Endgame & Passing Turns

Endgame and passing turns have been eliminated to guarantee that a stone is always placed on the board and to keep the player constantly engaged. This ensures continual play and ensures that every chance for learning is presented. The traditional finale was also skipped, along with the territory scoring scheme. The endgame was initially coded to include both komi and territory scoring, however, it soon became clear that this added level of complexity would not be difficult to implement and become confusing for beginners. By eliminating the complexities related to endgame strategy and scoring, this choice significantly simplifies the game for beginners.

2.3.7 Exclusion of Komi

The traditional "komi" system, which grants the second player compensation points, was left out. Players can concentrate on the core mechanics of stone capturing without being distracted by extra points due to this decision's simplification of the scoring system.

2.3.8 Change in Scoring Focus

Instead of declaring a winner at the end of the game, the main goal was changed to measure the amount of stones that each player had captured. Due to this modification, this is ideal since players can learn without the pressure of competing, which encourages a more exploratory and less competitive approach.

In summary, the *Go* AI helper is a helpful tool for beginner level players, who want to enhance their capture abilities and gameplay in general. The adoption of the Japanese style of play for *Go* is guaranteed by a thorough comparative analysis, which also provides accessibility and strategic depth. The user interface design places a high priority on the user-friendly features while offering the player helpful information and clear move suggestions.

With its Move Suggestion and Capture Help options, the AI helper menu improves the player's learning experience by providing insightful tactical advice. The AI assistant makes a substantial contribution to the player's development and fulfillment of the game by combining the strength of the Minimax algorithm with an intuitive UI. The AI acts as a dependable guide as players explore the realm of capturing opponent stones and planning their moves, developing their strategic thinking abilities and establishing an ongoing respect for the specifics of *Go*.

Players set out on a path of constant advancement and game mastery with the AI assist of the *Go* game. Players can become skilled *Go* players by experiencing the difficulties and elegance of the old game due to the AI's strategic guidance and insights. Players that use the AI to achieve their strategic potential join the dynamic and diverse global community of *Go* players, continuing the game's illustrious tradition.

2.4 Algorithm and Technology Utilized

2.4.1 Minimax Algorithm

The AI Capture Stone Helper's fundamental decision-making tool is the Minimax algorithm, which works by continuously traversing the game tree and looking at each alternative board configuration or position that might result from recent and future movements. It determines a value for each place using a specified evaluation function. The recursive structure of the Minimax algorithm in *Go* is an advantage. It looks deeply into many different possible move combinations, scoring them according to desirability. Such a technique is crucial in a strategy-heavy game, where every move can ultimately affect the result.

The Minimax algorithm's key feature is its capacity to assess positions from the viewpoints of both maximizing and minimizing players. This means that in the game of Go, it evaluates the plays that would be most advantageous for the player (player maximization) as well as the actions that would be most advantageous for the opponent (player minimization). This dual viewpoint makes sure that the AI considers probable future countermoves as well as the current best move[11].

2.4.2 Detection of Potential Eyes for Group

An algorithm was created to find prospective eyes, or important strategic locations, for a group of stones on the board. The algorithm works by going through each stone in a group iteratively and checking the surrounding positions for probable eyes. Potential eyes are surrounded by opponent stones, which are counted. By using this strategy, it is ensured that the AI will be able to recognize and effectively target important board positions that offer tactical benefits.

Blocking Potential Eyes: Another function was created to block the identified possible eyes and produce moves that would result in capturing the stones of the opponent. The first step of the algorithm is to find the group of stones that are linked to a specific stone. The best capturing move will be chosen after creating capturing moves against this group. Lastly, it makes suggestions to the player regarding where to put their stone in order to eliminate as many of the opponent's stones and future captures of their own stones

2.4.3 Communicating AI Decisions

The feature conveys the AI's choice to the user to make the gameplay experience more interesting and informative. This involves identifying the row and column of the chosen move, emphasizing the need of blocking possible eyes, and emphasizing the move's role in restricting the opponent's freedoms. The player's comprehension of strategic thinking and tactical execution improves as a result of this dynamic dialogue.

2.4.4 Promoting Strategic Understanding

By introducing the ***blockPotentialEyes*** function into my AI, I aim to educate players to the concept of eye-blocking as a critical technique in *Go*. This approach pushes players to think critically regarding their opponent's future actions, predict flaws, and deliberately undermine their strategic positions on the board.

2.4.5 Reasoning for Omitting Machine Learning and Deep Learning

During the development stage, significant decisions regarding technology and approach were made. Deep learning (DL) and machine learning (ML) methods were purposefully left out. While ML and DL have made significant contributions to the creation of advanced *Go* tools like AlphaGo, its main use is in simulating the complexity and tactics of expert-level play. Implementing such advanced technologies would be overkill for the aim of this game, which stresses stone capturing over the entire gameplay of *Go*, and might possibly overcomplicate the learning process for players.

Furthermore, there wasn't sufficient time to fully explore the complexity of ML and DL given the timeline and depth of this project. Although they have their advantages in more complex applications, it was decided that it would be better suited for the purpose of this game and its intended audience to avoid these technologies in favor of simpler and easier to understand concepts.

The main goal was to establish a setting where players could learn the fundamentals of *Go* without being overloaded with the level of concentration required in competitive games. The game prevents introducing algorithms that can make choices too complex for new players to understand by avoiding ML and DL. The AI's logic is more comprehensible and straightforward since it concentrates on the core principles of stone capturing. With this strategy, players are guaranteed to learn naturally without becoming overwhelmed.

2.4.6 Direct Feedback Ul

The Direct Feedback Ul is one of the crucial elements integrated into the game's design. This feature makes sure that players get rapid feedback on their actions on the console, allowing them to understand the impact of their choices in real-time. This loop was created with the knowledge that novices frequently struggle with abstract ideas without seeing any immediate benefits. By including rapid feedback, the player may see the effects of their decisions, both good and bad. This helps in both reinforcing effective tactics and spotting and fixing faults early on. This immediate feedback reduces the learning curve and makes sure that players don't develop bad habits based on bad tactics.

2.4.7 Alternative Approaches Considered

Several alternative AI approaches were investigated before the Minimax algorithm's acceptance was decided on. For example, neural networks were taken into consideration due to their strength in pattern recognition, a key component of *Go*. Another option was to use evolutionary algorithms, which are flexible and dynamic in nature and repeat and evolve solutions over time. The Minimax method was selected, however, because of its direct applicability to turn-based strategy. Although neural networks are excellent at identifying patterns, they need a lot of training data and may be overkill for beginners just starting out. On the other hand, while adaptive, evolutionary algorithms might not always provide the best answers when dealing with *Go*'s constrained move sequences.

2.4.8 AI Move Suggestion

The move suggestion AI helper's major objective is to aid players who are unsure about their next move or choose not to capture a certain stone. This AI assistant evaluates the board's current status and proposes a potentially strategic position for the player to place their stone and potentially capture a stone.

Corner, Side, and Center Board Strategy: One of the *Go* strategies is to prioritize the board's corners, then the sides, and finally the center of the board. This strategy is founded on the idea that dominating the corners can offer a stable foundation for territory growth since corners take. Sides are less difficult to protect than center regions, because of fewer stones to defend.. This method is included into the move suggestion function, which aims to recommend moves that adhere to these standards.

2.4.9 Move Analysis

The AI helper tool's first priority is to determine if there are any stones on the board that can be captured. Capturing opponent stones not only increases the player's score but also has the ability to weaken the opponent's position/ liberties. If the AI detects probable captures, it will prioritize recommending such moves. If no quick captures are found, the AI returns to the standard board prioritizing technique. It evaluates the best locations to play based on the notion of dominating the board's corners first, then the sides, and finally the center of the board. This approach is supported by the fact that corners take fewer stones to defend than open spaces, making them important territory.

2.5 AI Capture Stone Helper Design

2.5.1 Pedagogical Design of the AI Helper

The importance of the *Go* AI help goes beyond just providing immediate gameplay assistance. It is essential for encouraging players' strategic development and critical thinking abilities. By using the Minimax algorithm, the AI encourages players to consider more complex strategic options, taking into account both their movements and the possible effects on subsequent turns. This all-encompassing method promotes a deeper comprehension of *Go*'s complex strategy, enabling players to make wise decisions and predict their opponents' moves.

Through repeated interactions with the AI helper, players gain invaluable insights into capturing strategies and stone placements that limit opponent liberties. These insights serve as building blocks for players to develop their unique playing styles and explore more complex tactical maneuvers. As players become accustomed to the AI's guidance, they gradually transition from relying solely on the AI's recommendations to formulating their strategies, thereby nurturing their strategic thinking capabilities.

Due to its commitment to the Japanese style of *Go* games, the AI plays a greater role in serving as a strategic instructor. The AI makes sure that new players understand the core strategic ideas without being daunted by complicated rules or tactics by placing a priority on simplicity and essential principles. Players can build on this foundation when they gain a solid understanding of the fundamentals of *Go* and then study more complex strategies like fuseki (opening strategies), joseki (corner patterns), and life-or-death situations.

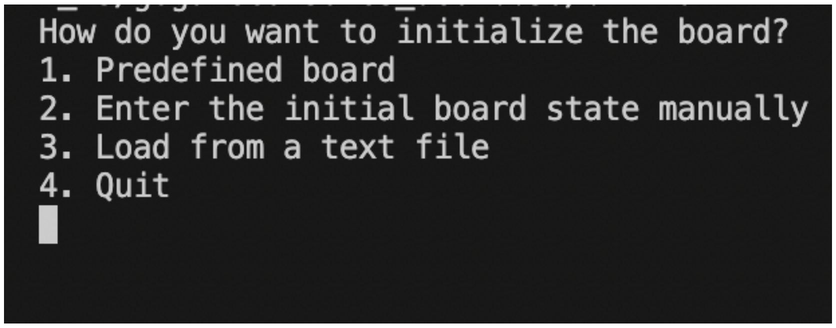
The AI assistant's dynamic and user-friendly UI also helps the player develop their strategy. In order to better understand the effects of their choices, players can experiment with different movements and watch how the AI reacts. The AI's move suggestions are accompanied by concise explanations that improve the learning process and make players more conscientious of their strategic decisions.

2.5.2 User Interface Design and Experience

The user experience was a central consideration throughout the development process. As the sole developer of the *Go* game AI helper, the goal was to create an intuitive and effective tool that assists beginners in learning capturing strategies in *Go*. The user interface was designed to be user-friendly and interactive, allowing players to easily input their moves or request AI assistance through intuitive commands. The AI's responses were presented in a clear and concise format, explaining the rationale behind each suggestion and providing valuable insights into capturing strategies. By focusing on user experience, the AI helper serves as a valuable learning tool that enhances players' understanding of *Go*'s strategic elements, particularly in capturing opponent stones. Java was chosen for the programming language, because of my knowledge of the language. A website was not developed for this project.

2.5.3 Design

When designing the game, I took into account the simple menu board for the user to choose from, as shown in Figure 1. The design focus of the console game, taking into consideration the two player board and ease of use for the player, while maintaining full functionality of the game, as illustrated in the screens presented below.



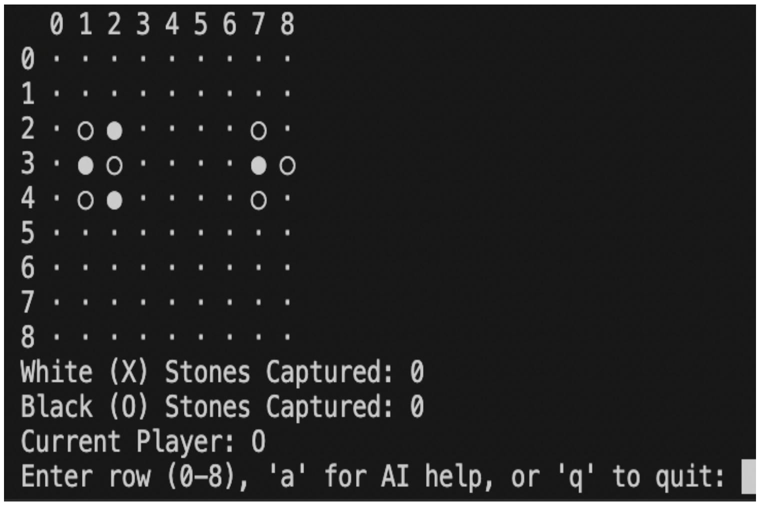
```
How do you want to initialize the board?
1. Predefined board
2. Enter the initial board state manually
3. Load from a text file
4. Quit
```

Figure 1: Starting Menu on the console.

Go Problems / Patterns:

For the main board shown in Figure 2, the output of the stones were changed to look like a *Go* board in order for the player to see and understand what move is being played; however if the user doesn't like this configuration, they can change the images of the white and black circle to

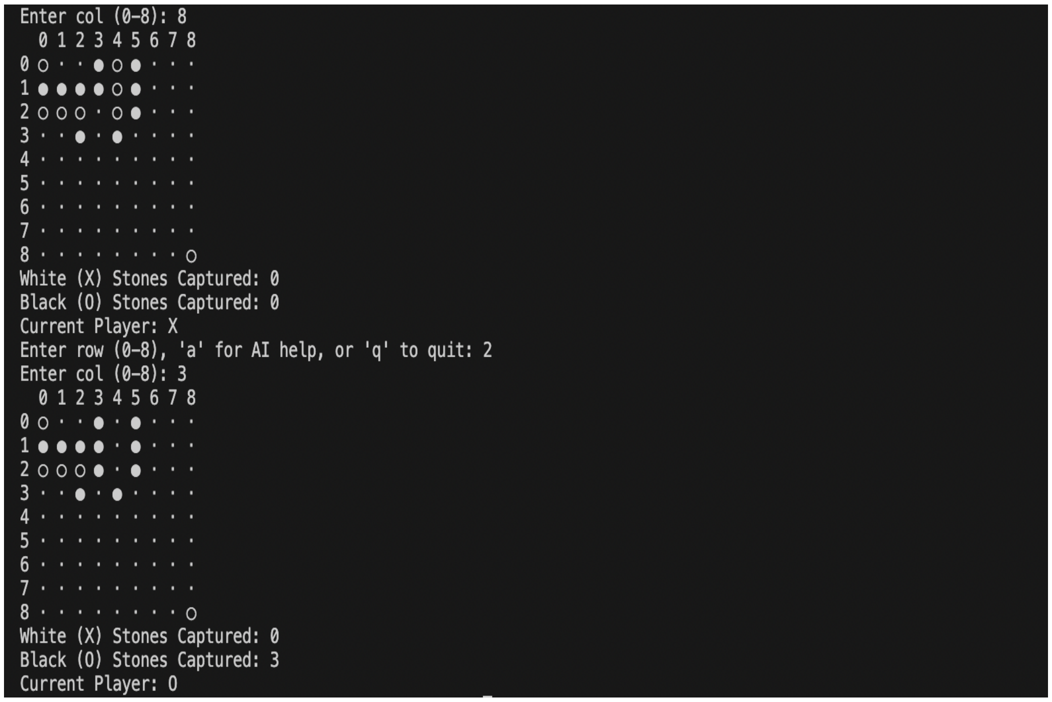
"X" and "O". To input the stone, the user will enter "X" as the white stone and "O" as the black stone. The board displays numbers instead of letters for the coordinates for row and col so that the user can easily understand where the current player's stone is being placed and for the player to understand where to capture an opponent's stone.



```
  0 1 2 3 4 5 6 7 8
0 · · · · · · · · ·
1 · · · · · · · · ·
2 · ○ ● · · · · ○ ·
3 · ● ○ · · · · ● ○
4 · ○ ● · · · · ○ ·
5 · · · · · · · · ·
6 · · · · · · · · ·
7 · · · · · · · · ·
8 · · · · · · · · ·
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Current Player: O
Enter row (0-8), 'a' for AI help, or 'q' to quit:
```

Figure 2: Design of X and O player

Players capture stones by limiting the opponents' liberties as shown in Figure 3. In Figure 3, the white stone player has captured three black stones by placing their stone at row 2, col 3, with the board being updated with now empty spaces where the captured black stones were.

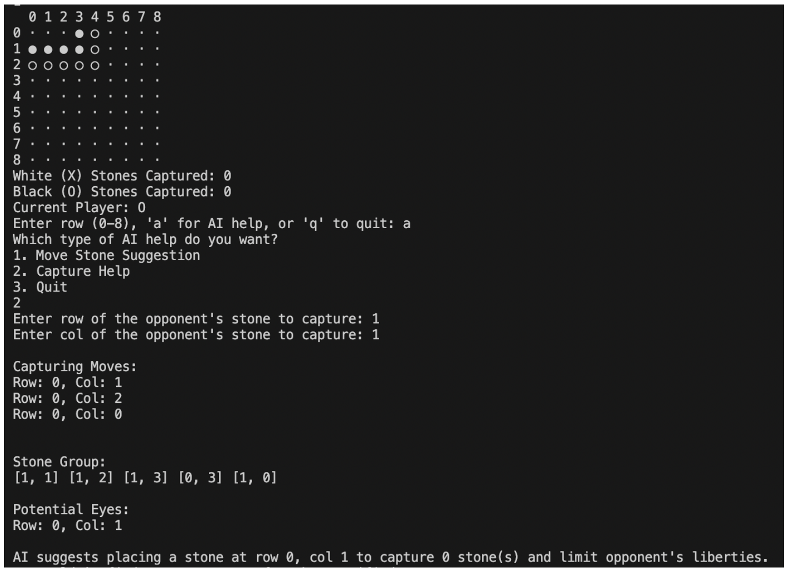


```
Enter col (0-8): 8
  0 1 2 3 4 5 6 7 8
0 ○ · · ● ○ ● · · ·
1 ● ● ● ● ○ ● · · ·
2 ○ ○ ○ · ○ ● · · ·
3 · · ● · ● · · · ·
4 · · · · · · · · ·
5 · · · · · · · · ·
6 · · · · · · · · ·
7 · · · · · · · · ·
8 · · · · · · · · ○
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Current Player: X
Enter row (0-8), 'a' for AI help, or 'q' to quit: 2
Enter col (0-8): 3
  0 1 2 3 4 5 6 7 8
0 ○ · · ● · ● · · ·
1 ● ● ● ● · ● · · ·
2 ○ ○ ○ ● · ● · · ·
3 · · ● · ● · · · ·
4 · · · · · · · · ·
5 · · · · · · · · ·
6 · · · · · · · · ·
7 · · · · · · · · ·
8 · · · · · · · · ○
White (X) Stones Captured: 0
Black (O) Stones Captured: 3
Current Player: O
```

Figure 3: Black stones being captured

Eyes Design on Console:

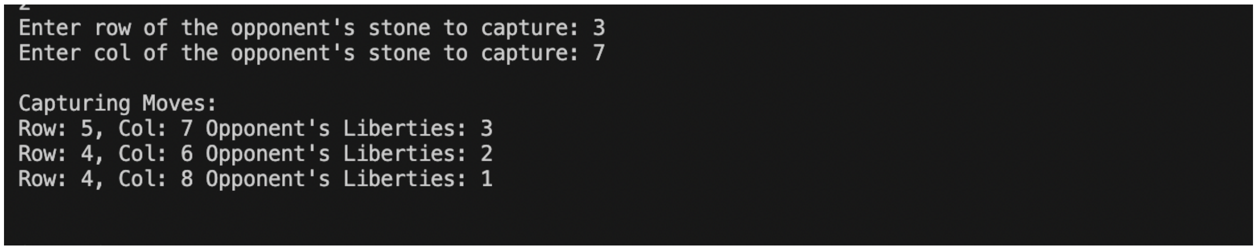
As illustrated in Figure 4, the player is shown how to prevent the opponents from making eyes on row 0 col 1 because if the opponent forms an eye in that position , the player of the black stone can't capture the group of white stones.



```
  0 1 2 3 4 5 6 7 8
0 · · · ● ○ · · · ·
1 ● ● ● ● ○ · · · ·
2 ○ ○ ○ ○ ○ · · · ·
3 · · · · · · · · ·
4 · · · · · · · · ·
5 · · · · · · · · ·
6 · · · · · · · · ·
7 · · · · · · · · ·
8 · · · · · · · · ·
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Current Player: O
Enter row (0-8), 'a' for AI help, or 'q' to quit: a
Which type of AI help do you want?
1. Move Stone Suggestion
2. Capture Help
3. Quit
2
Enter row of the opponent's stone to capture: 1
Enter col of the opponent's stone to capture: 1

Capturing Moves:
Row: 0, Col: 1
Row: 0, Col: 2
Row: 0, Col: 0


Stone Group:
[1, 1] [1, 2] [1, 3] [0, 3] [1, 0]

Potential Eyes:
Row: 0, Col: 1

AI suggests placing a stone at row 0, col 1 to capture 0 stone(s) and limit opponent's liberties.
```

Figure 4: Eyes being blocked

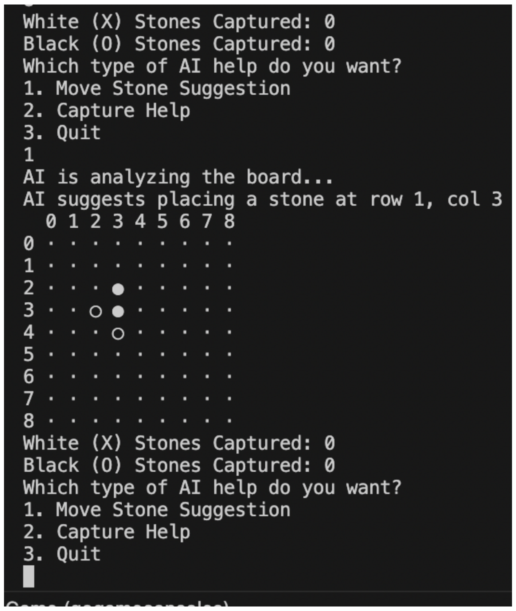
As illustrated in Figure 5, where the player can place their stone to limit the opponents limits, this is displayed if the user wants to learn why these are the best moves to make for capture of the opponents stone(s). Also the player can see where their stones are for moves that don't have eyes.



```
Enter row of the opponent's stone to capture: 3
Enter col of the opponent's stone to capture: 7

Capturing Moves:
Row: 5, Col: 7 Opponent's Liberties: 3
Row: 4, Col: 6 Opponent's Liberties: 2
Row: 4, Col: 8 Opponent's Liberties: 1
```

Figure 5: Capturing moves and opponent’s liberties

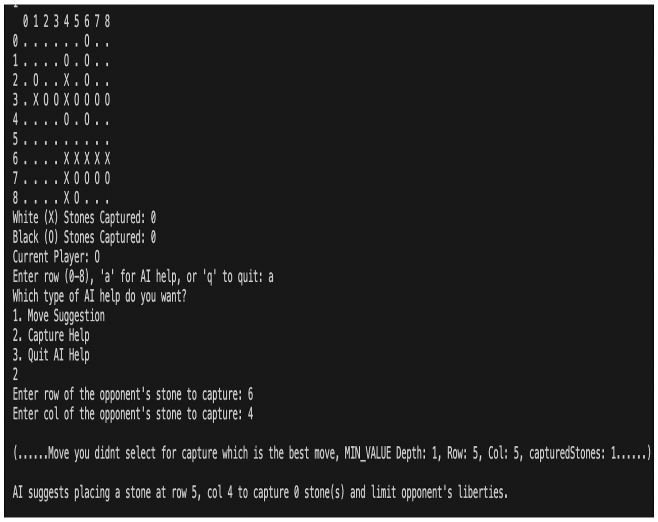
As illustrated in Figure 6, the AI Move Suggustion from Option 1 of the AI menu shows the AI helper selecting a move the current player can select to future capture an opponent’s stones.



```
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Which type of AI help do you want?
1. Move Stone Suggestion
2. Capture Help
3. Quit
1
AI is analyzing the board...
AI suggests placing a stone at row 1, col 3
  0 1 2 3 4 5 6 7 8
0 · · · · · · · · ·
1 · · · · · · · · ·
2 · · · ● · · · · ·
3 · · ○ ● · · · · ·
4 · · · ○ · · · · ·
5 · · · · · · · · ·
6 · · · · · · · · ·
7 · · · · · · · · ·
8 · · · · · · · · ·
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Which type of AI help do you want?
1. Move Stone Suggestion
2. Capture Help
3. Quit
```

Figure 6: Move Suggestion AI Helper Analyzing

As illustrated in Figure 7, the minimax shows that the
eye function works better than my minimax, it provides info on my opponents stones and how to capture them when there is more than a two empty spaces.



```
  0 1 2 3 4 5 6 7 8
0 . . . . . . 0 . .
1 . . . . 0 . 0 . .
2 . 0 . . X . 0 . .
3 . X 0 0 X 0 0 0 0
4 . . . . 0 . 0 . .
5 . . . . . . . . .
6 . . . . X X X X X
7 . . . . X 0 0 0 0
8 . . . . X 0 . . .
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Current Player: O
Enter row (0-8), 'a' for AI help, or 'q' to quit: a
Which type of AI help do you want?
1. Move Suggestion
2. Capture Help
3. Quit AI Help
2
Enter row of the opponent's stone to capture: 6
Enter col of the opponent's stone to capture: 4

(......Move you didnt select for capture which is the best move, MIN_VALUE Depth: 1, Row: 5, Col: 5, capturedStones: 1......)

AI suggests placing a stone at row 5, col 4 to capture 0 stone(s) and limit opponent's liberties.
```

Figure 7: Eye and Minimax output

Implementation

2.6 Application of Minimax Algorithm in Go

2.6.1 Minimax Algorithm in GoGame

The Minimax algorithm was implemented to find the optimum move for a specific game state in order to improve the AI of the game. A decision-making technique, Minimax is used in two-player games like *Go* to find the best move by thoroughly simulating all potential outcomes.

Explanation of how it was implemented:

The researchers, Quentin Cohen-Solal and Tristan Cazenav, challenged the dominance of the Monte Carlo Tree Search (MCTS) in complete information games. They introduce Athénan, a Minimax-based reinforcement learning framework, which they propose as a competitive, and in many cases superior, alternative to MCTS-based algorithms. The research juxtaposes AlphaZero, which utilizes MCTS and has surpassed even grandmaster-level programs in chess without the need for prior knowledge, with their Minimax-based approach. The authors advocate for a re-evaluation of Minimax algorithms, especially when they are paired with strong evaluation functions derived from reinforcement learning [9]. The study highlights the potential for a paradigm shift in AI game strategies and calls for a reconsideration of established methodologies. necessitated an iterative development philosophy, embracing continuous improvement to refine the AI's algorithms and strategies.

2.6.2 Depth and Evaluation

The parameter that is used for the recursive function is the depth. The function returns a board evaluation score after the depth has been achieved. Based on the current player and the board layout, the GoGame.evaluateBoard method offers a systematic evaluation of the game state.

2.6.3 Switching Players

The two players are switched by the variable opponentPlayer. The function now simulates both offensive and defensive plays while taking into account both maximizing and minimizing perspectives due to this switch.

2.6.4 Maximizing Player

When maximizing the player's position, the algorithm seeks out the move with the highest evaluation score. It iterates over every move that is conceivable after initializing the bestScore to the smallest integer value (Integer.MIN_VALUE). It uses the GoGame.simulateMove method to simulate the results of each move before recursively evaluating the resulting board. Following each move, the configuration of the board and the assessment score of the board are printed.

2.6.5 Minimizing Player

In contrast, while the algorithm is deliberating on the perspective of the opponent, it searches for the move with the lowest evaluation score. It simulates each possible move and iterates until the bestScore is set to the greatest integer value (Integer.MAX_VALUE) in an effort to reduce the score. For each move, the final board and assessment score are printed.

2.6.6 Return Value

The function returns the bestScore, which represents the best result for the current player up to the given depth, after iterating through all potential movements and evaluating them.
BestScore is equal to the best max then other moves for the player. Forces on the best score for the max score

2.6.7 Fine-tuning the AI for the Capture Stone Method

If a stone is captured, then you exit the recursion.
The development approach placed a lot of attention on building an AI that is friendly to new players. In order to prevent the AI from intimidating newcomers with extremely complex tactics that can be discouraging, the depth of the Minimax algorithm's search was carefully chosen.

2.6.8 Limited Depth Search for Stone Capturing

The Minimax algorithm's search was carefully limited in depth so as not to overwhelm novices with the huge possibilities of stone capturing. This deliberate restriction confirms that the AI does not become too deep into complicated techniques and instead, concentrates on immediate stone-capturing opportunities.

2.6.9 Tailoring the AI for different skill levels

As part of the AI's adaptability, I also implemented different modes that adjust the AI's playing style based on the user's experience level. For beginners, the AI prioritizes safe and defensive moves, gradually introducing them to the game's tactical aspects. On the other hand, for more experienced players, the AI employs more aggressive strategies, offering a challenging gameplay experience.

2.6.10 Diverse Stone Capturing Strategies

The AI was carefully designed to offer a variety of stone capturing techniques. This allows beginners to explore numerous tactics and thereby comprehend the variety of *Go*'s potential strategies by introducing them to various methods of capturing stones.

2.6.11 Strategic Insights for Stone Capturing

The AI gives straightforward information on the strategic ramifications of each stone-capturing maneuver to help with the learning curve. Stone-capture-related hazards and opportunities are underlined in order to make sure that players understand the reasoning behind each choice.

2.6.12 Clear Console Outputs for Stone Capturing
The console output was improved to detail each phase of the AI's stone capturing process in an attempt to make the procedure for capturing stones more transparent. Players can then comprehend the AI's reasoning and the choices that led to each stone-capturing navigate by doing this.

2.6.13 Evaluation Function and Strategic Styles
The evaluation function, which rates the merits of a particular game state, is an essential component of the Minimax algorithm [10]. I customized the evaluation system to direct the AI's strategic approach by including elements like board control, seized stones, and liberties. Now that players may interact with several AI characters and gameplay ideas, they are able to more understand the subtleties of *Go*'s strategic complexities.

2.6.14 Challenges and Limitations of Implementing Minimax in a Capture-Focused Go AI
Despite the fact that the Minimax algorithm has a lot of potential for building a competitive and strategic AI opponent for *Go*, there are certain difficulties and constraints I encountered during the development process. It was crucial to comprehend and resolve these problems to ensure the AI would be a useful learning tool.

The path of implementing the Minimax algorithm to build a *Go* AI that is capture-focused has been enlightening, but the tool hadits share of obstacles and restrictions. It was vital to address these concerns to verify that the AI effectively fulfills its mission since the AI's main goal is to help beginners understand capture methods and control opponent liberties.
Challenge 1: Defining a Targeted Strategy
Even though conventional Minimax implementations frequently strive for overall strategic perfection, configuring the algorithm to give priority to capturing movements and opponent liberty reduction posed a specific challenge. The AI must assess movements according to their potential to grab stones and restrict the opponent's strategic options rather than aiming for outright success. It took much deliberation and changes to the evaluation function to define this targeted strategy.

Challenge 2: Strategic Positioning vs. Capturing
Capturing stones and limiting liberties, while important, tactics must be evaluated against more general strategic considerations. By striking this balance, one can prevent the AI from being too preoccupied with winning individual stones at the risk of missing out on more significant strategic possibilities. A well-rounded AI required the development of an evaluation mechanism that equally values tactical and strategic aspects.

Challenge 3: Teaching Value to Beginners

The main goal of the AI is to teach new users about capture and liberty control, which presents a special problem. The AI's ideas and actions must be understandable and applicable to new players, avoiding overly complicated or esoteric strategies that can be intimidating. It was difficult to translate complex strategic ideas into engaging learning situations, instead carefully selected actions and explanations were picked.

2.7 Strategic Planning and Anticipation with Minimax in Go

The Minimax algorithm's use in the game of *Go* enables players to plan ahead several steps, giving them the capacity to recognize and react to their adversaries' strategies. The AI assists players in identifying patterns and carefully placing stones to build up future captures by foreseeing potential outcomes. This long-term goal necessitates a greater comprehension of the mechanics of the game, including identifying potential dangers to the player's stones and avoiding traps. Furthermore, for innovative effectiveness, a delicate balance between computing efficiency and search depth, supported by a carefully designed evaluation function, is essential. Players can explore the many facets of *Go* and develop skills that will help them succeed in this age-old game by utilizing Minimax's forward-thinking capabilities. Implementing the Minimax algorithm for *Go* requires blending computational effectiveness with strategic depth, embodying the complexity that makes the game both fascinating and compelling.

Limitation 1: Specialized Strategy

A drawback of the AI's specialization in capture moves and opponent liberty reduction is that it could perform poorly in situations where these strategies are less important. The AI's capture-focused strategy, for instance, might not work best when dealing with influence-based strategies or strategic board building. While it excels in the area for which it was designed, its capacity to adapt to a wider range of strategic situations could be limited.

Limitation 2: Ignoring Other Aspects

The AI's exclusive focus on liberty and capture inherently implies that other important game elements, including endgame strategies or intricate life-and-death situations, could not be properly addressed. This restriction results from the AI's particular objective and can call for further development or adaptation to increase the scope of its strategic knowledge.

Limitation 3: Narrow Applicability

Due to its specialization, the AI performs best when capturing and restricting liberties are essential to the game's development. The AI's recommendations could not be consistent with overarching strategic objectives in situations when other strategic factors take precedence. The AI's utility may therefore be constrained in games that need for a wider variety of strategic thinking, yet being extremely helpful for learning capture strategies.

2.7.1 Handling the Ko Rule in Strategic Planning

The Ko rule's application, which adds a substantial layer to the strategic planning necessary when using the Minimax algorithm, is one of *Go*'s special complications. By capturing and recapturing the same stones, this rule stops the board from repeating itself and ensures that players cannot establish an endless cycle. The *Go* game becomes more complex because of the Ko rule, which offers a subtle element that the AI must pick up on and react to.

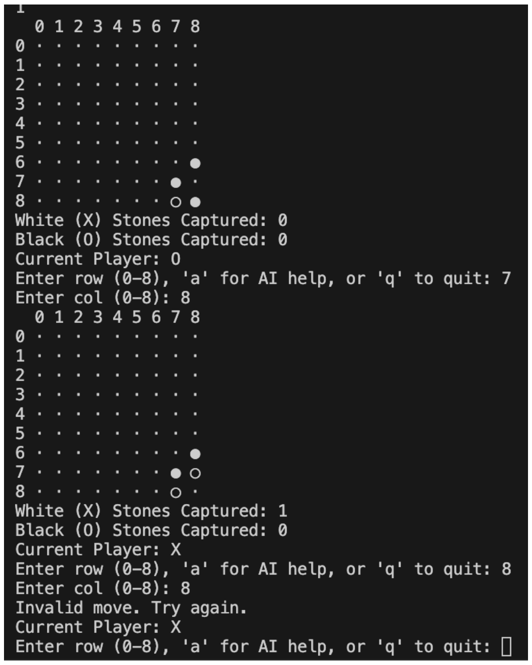
The Minimax algorithm for *Go* requires the AI to be aware of Ko situations and avoid illegal moves in order to maintain the authenticity of the gameplay. It challenges the AI to predict not just the immediate effects of its decisions but also potential future board states brought on by Ko situations. The AI can strategically position its stones using predictive strategic planning to take advantage of the Ko rule while preventing the opponent from doing the same.

The complexity of AI development is increased by the need to strike a balance between adhering to the Ko rule and maintaining a successful strategy. The inclusion of this rule in the algorithm guarantees that the AI replicates the complexity of *Go* in the real world, providing players with a realistic gameplay experience. The Minimax algorithm's versatility and depth are further highlighted by its success in capturing the subtle nuances of *Go* by successfully managing the Ko rule.

2.7.2 Code Implementation for Handling the Ko Rule

To ensure that the AI abides by the Ko rule, a specific check has been implemented in the code. In order to determine whether a move is legal, the ***isValidMove*** function is used. It determines whether the move falls within the limits of the board before confirming that the desired position is open. In the end, it determines whether the proposed move will lead to a board state that exactly replicates the one just before, which would be against the Ko rule if a stone is placed there .

Additionally, the primary purpose of the ***isKoRuleViolation*** function is to confirm alleged Ko rule violations. The algorithm simulates the result of the move without permanently changing the main board by employing a temporary board. A violation of the Ko rule has occurred, and the move is judged unlawful if it leaves the board in the same state as before(figure 8). This strategy guarantees that the AI adheres to the Ko rule, preserving the core and nuances of the game of *Go*.



```
1
  0 1 2 3 4 5 6 7 8
0 · · · · · · · · ·
1 · · · · · · · · ·
2 · · · · · · · · ·
3 · · · · · · · · ·
4 · · · · · · · · ·
5 · · · · · · · · ·
6 · · · · · · · · ●
7 · · · · · · · ● ·
8 · · · · · · · ○ ●
White (X) Stones Captured: 0
Black (O) Stones Captured: 0
Current Player: O
Enter row (0-8), 'a' for AI help, or 'q' to quit: 7
Enter col (0-8): 8
  0 1 2 3 4 5 6 7 8
0 · · · · · · · · ·
1 · · · · · · · · ·
2 · · · · · · · · ·
3 · · · · · · · · ·
4 · · · · · · · · ·
5 · · · · · · · · ·
6 · · · · · · · · ●
7 · · · · · · · ● ○
8 · · · · · · · ○ ·
White (X) Stones Captured: 1
Black (O) Stones Captured: 0
Current Player: X
Enter row (0-8), 'a' for AI help, or 'q' to quit: 8
Enter col (0-8): 8
Invalid move. Try again.
Current Player: X
Enter row (0-8), 'a' for AI help, or 'q' to quit: ▯
```

Figure 8: In the console the ko rule is implemented.

2.7.3 Minimax Algorithm Implementation

The basic justification of AI help is based on the Minimax algorithm. The Minimax algorithm is a well-liked decision-making technique in artificial intelligence and game theory. In a game of *Go*, attempting to capture an opponent's stones enables the AI assistant to anticipate moves two steps ahead. The AI can recommend actions that both give quick opportunities to capture the opponent's stone(s) and restrict the opponent's options in subsequent turns. Players who employ this tactic have an advantage over their rivals since they are better at thinking strategically and planning ahead.

2.7.4 Comparison Between Minimax and Monte Carlo Methods for Go Game AI

The complexity of *Go*'s strategic depth and the game's enormous decision space provide challenges when creating an AI for it. The fact that I took on the responsibility of being the only developer of the AI aid tool made this problem more difficult for me. In addition to creating an AI, my objective was to make sure that it offered helpful and accurate move recommendations, particularly for players who were unfamiliar with the game [12]. Two main methods—Minimax and Monte Carlo—presented themselves in the search for the optimal algorithmic strategy. Both approaches have their particular benefits and peculiar difficulties.

My choice to use the Minimax approach was influenced by a number of factors, including:

Minimax operates by thoroughly examining the game tree, which enables it to weigh the effects of alternative movements and countermoves. This methodical study makes sure that the AI takes into account several game paths before making a choice.

2.7.5 Simplicity and Readability

Minimax is deterministic in contrast to Monte Carlo methods, which use random simulations and probabilistic evaluations. It offers a simpler strategy that is simpler to comprehend and modify, especially for someone who is the single developer. *Go* contains a lot of capturing moves, and the Minimax tree structure makes it possible to analyze these capture possibilities more thoroughly. The AI can decide more wisely by considering probable captures and their effects.

Although Monte Carlo methods can provide a reliable solution, particularly in games with a large decision space, they also need a significant amount of computer power, especially when simulating multiple game possibilities. On the other hand, Minimax offers a more computationally efficient strategy, particularly when working inside a specified depth.

2.7.6 Probabilistic vs. Deterministic

Minimax thoroughly examines each move and counter-move that might be made in the game tree. This predictive strategy allows the AI to plan strategically by taking multiple steps into account. The best move is determined using random simulations through Monte Carlo methods, such as Monte Carlo Tree Search (MCTS), which are probabilistic [10]. Minimax's deterministic nature is a good fit for the game of *Go* because it requires careful strategic preparation and foresight of your opponents' moves.

2.7.7 Complexity of the Game

Go with each turn allows for an average of about 250 potential movements. Monte Carlo techniques can effectively sample random plays in games with huge decision spaces, but the decision space in *Go* is still very enormous, even with MCTS.

The article "Monte-Carlo Strategies for Computer Go" by Guillaume Chaslot, Jos W. H. M. Uiterwijk, Bruno Bouzy, and H. Jaap Van Den Herik offers information on numerous Monte Carlo tactics for computer *Go*, highlighting the difficulties and potential advantages of employing such techniques. Minimax, on the other hand, delivers computational efficiency without compromising strategic depth, making it more ideal for managing *Go*'s complexity. Minimax can efficiently explore a sizable percentage of the game tree within a constrained depth.

2.7.8 Summary of Minimax vs. Monte Carlo Methods

There are a number of variables to take into account when comparing the Minimax and Monte Carlo strategies for *Go* beginners. Because Minimax is deterministic, its move suggestions are predictable and consistent, which is crucial for beginners learning the game's core concepts. For people who are new to *Go*, it is more relevant and intuitive because of the way it logically weighs potential outcomes in accordance with human decision-making. The depth of Minimax's search can also be adjusted to match a beginner's ability level, resulting in a steady learning curve and allowing for strategic insight and depth. On the other hand, Monte Carlo techniques rely on random simulations that could add variability and inconsistent results, notwithstanding their effectiveness in handling complex decision spaces. This stochastic nature could make it harder for beginners to grasp and may not reflect the strategic diversity of the game as well as Minimax. Additionally, compared to the simpler and more manageable Minimax strategy[9], the computing intensity needed for Monte Carlo's numerous simulations could not result in a noticeably improved learning experience for beginners [12]. Minimax thus appears as the most appropriate option in the context of creating an AI that fosters comprehension and appreciation of *Go*'s complexity and beauty among novice players.

Careful consideration was given to the deterministic and strategic character of Minimax, its computing effectiveness, and the accuracy of its move suggestions before the Minimax algorithm was chosen to be used in the *Go* game AI aid [10]. Minimax is an excellent alternative for offering an efficient and user-friendly tool because of its capacity to plan many moves and take both conservative and aggressive plans into account. I wanted to build an AI that not only provides helpful move ideas but also inspires new players to discover and appreciate the complexity and beauty of the old game of *Go*, therefore I chose Minimax over Monte Carlo techniques.

The Minimax algorithm's use in *Go* has proven to be a useful tool, offering a difficult and tactical gameplay experience. Despite the difficulties experienced throughout the creation phase, the AI succeeded in becoming a trustworthy and knowledgeable learning companion. The Minimax algorithm and evaluation functions are improved, allowing the AI to simulate realistic game circumstances and predict the opponent's moves[11]. This harmony between gameplay and player participation has the power to improve the *Go* playing experience, motivate new players, and support the expansion of interest in this game.

2.7.9 User Interface

The graphical user interface of the game is simple to use. The AI's ideas, produced by combining the Minimax algorithm and the custom functions, are highlighted as users add stones by clicking on the board. Users can interact with the game by offering feedback for moves in real-time, including warnings for invalid moves or other game-specific violations.

2.7.10 Blocking Potential Eyes Strategy Implementation

In the game, the idea of blocking prospective eyes is a key tactic to constrict the adversary's choices and obtain an advantage on the board. This method is implemented in the ***blockPotentialEyes*** function, which gives my *Go* AI the ability to foresee and mitigate possible risks posed by the opponent's stone groups.

2.7.11 Detecting Vulnerable Stone Groups

The blockPotentialEyes function begins by locating the collection of stones that are linked to a certain target stone. An important connectivity analysis is used in this step, where stones that are adjacent or connected by a chain of stones and have the same hue are grouped together. This cluster of stones is a possible source of control for the opponent, so it should be properly eliminated.

2.7.12 Generating Capturing Moves

The function then creates capturing moves after isolating the weak stone group. Stone placements that would restrict the stone group's freedom and raise the possibility of capturing it in later moves are known as capturing moves. The function evaluates the different ways it can weaken the group's position and exploit its vulnerabilities by taking into account all potential capturing moves against the group.

2.7.13 Analyzing Potential Eyes

In addition to creating capturing movements, the function investigates probable eyeballs inside the indicated stone group. An "eye" in *Go* refers to an empty location surrounded by stones of the same color. A group with two or more eyes is regarded as "alive" and so cannot be caught, but a group with fewer than two eyes is considered "dead" and hence vulnerable to capture. The eye function and the capture ai stone function obtains a better awareness of the group's life-and-death situation by recognizing probable eye placements among the group.

2.7.14 Choosing the Best Move

The core of the function is to choose the best move to effectively block possible eyes while maximizing strategic benefit. This choice is determined by an evaluation procedure that considers the influence of the capturing maneuvers on the group's freedoms as well as the urgency of blocking possible eyes. The AI automatically chooses the most advantageous action to perform by measuring and comparing these aspects.

Development and Testing

2.8 Immersion & Playtesting

Go, a two-player strategy game, offers participants a strategic challenge. As the creator of a beginner-friendly *Go* AI, it was critical for me to create a helpful tool that not only introduces new players to the game but also provides meaningful feedback for those trying to improve their performance. To achieve this purpose, I employed the Minimax algorithm, a deterministic approach known for its ability to make strategic decisions. I started the project by immersing myself in the game of *Go*, playing a lot of matches against AIs generated by computers and human opponents.

Because of my practical knowledge, I was able to comprehend the game's main basis, understand its key principles, and identify frequent beginning mistakes. Playtesting highlighted the issues that players face as well as the minor strategic distinctions that distinguish good moves from less-than-optimal ones. I learnt about *Go* games through extensive playtesting, which included understanding more about the rules, capture mechanics, and strategic considerations. It also allowed me to identify the places where an AI assistant may have the most impact, assisting players while keeping the heart of the game's complexity.

After researching extensively into *Go*'s structure, I found myself needing a larger user-base who would appreciate the game and interact with the AI. While I have played several matches, testing and developing the system, the AI's actual potential resides in its interaction with a wide set of players. Although my independent playtests provided me with useful information, I appreciate the necessity of different player interactions. Observing new players interact with the AI would give unique viewpoints, showing issues they confront and indicating places for future enhancement. It is my hope that, in the future, this AI will not just be a tool I created for my personal progress, but will expand into a complete platform that will enable growth with real players.

2.9 Iterative Testing

An iterative testing method was used to improve the capturing skills of the AI helpers for *Go*. This entailed a lengthy series of studies focused at measuring the AI's behavior under various game settings, with a particular emphasis on areas where capturing maneuvers were needed. The primary goal was to identify areas that needed more improvement rather than just gauging the validity of the AI's suggestions.

During each iterative assessment step, a thorough comparison was performed by pitting AI-generated move suggestions against proven ideal capture techniques. Such evaluation had a dual purpose: first, it confirmed the AI's effectiveness as an instructional tool in the field of *Go* by identifying congruencies between the AI's judgments and the ideal moves. Second, any apparent discrepancies required a thorough analysis of the AI's core decision-making processes. Such assessments were aimed at determining the basic logic and identifying possible areas of risk inside the system. For example, certain scenarios were created in which a black stone was intentionally placed on the board and the AI move suggestion was then tracked. In cases where the AI provided incorrect move suggestions, further modifications were implemented with a particular focus on updating how liberties were processed and interacted with the overall board layout.

Iterative testing provided insights that served as the basis for precise changes to the AI's parameters and algorithms. The key steps to gradually improve the AI's capturing abilities were optimizing the strategic decision-making process and fine-tuning the capturing evaluation function. This iterative approach ensured that the AI provided valuable and accurate move suggestions. For example, wanting to capture two black stones on the board. The AI capture helper would take where to place the first white stone to limit the most of the opponents liberties then for the second move, the white stone will be placed where the stone had lesser liberties.

Every cycle included different preliminary attempts with assorted ongoing interaction situations, and I broke down the AI's reactions. In view of these perceptions, I made fundamental changes, for example, upgrading the Minimax calculation to deal with deeper depth and improving on the assessment capability to more readily catch vital contemplations. However, expanding the intensity will dramatically build the calculation time. The quantity of potential moves the calculation needs to assess increments dramatically with each additional profundity level. Thus, if your Minimax capability isn't advanced for more ventures or on the other hand in the event that the board is enormous, this can prompt perceptible postpones in the simulated intelligence's move choice.

2.10 Eye Testing

When I created the eyes in the game, my initial step involved creating a visual representation of the board, laying the groundwork to discern various eye formations. However, recognizing these patterns computationally presented significant challenges. Rather than depending exclusively on algorithms, I decided to submerge myself in real interactivity. This involved methodology uncovered nuanced circumstances and exemptions that were at first ignored. By creating different game environments and changing the algorithm according to the errors that were being encountered, I was able to change the AI's ability to detect eyes. This iterative testing process, even though it was time consuming, I was determined to make the eyes in the game work for the new players.

2.11 AI Helper Testing

As I started carrying out the AI helper, I applied a considerable amount of effort into ensuring it is exact and valuable for aiding players. I assessed the AI's reactions to a different game state, ranging from straightforward opening moves to show capture sequences. I tried the computer based intelligence's ability to make exact and helpful ideas to members by entering specific game positions. I was able to assess the AI's performance and improve the rationale behind its decision-making thanks to this rigorous testing approach.

The AI was developed through rigorous testing and iterative development, aiming to make *Go* accessible to newcomers while maintaining its strategic beauty. Its a supportive gameplay experience that empowers beginners to enhance their skills, learn essential concepts, and appreciate the game's depth. Continuous refinement and user feedback create a learning tool for aspiring *Go* players, fostering an exciting and rewarding journey into the world of *Go.*

Conclusion and Future Work

2.13 Reflection

I wanted to complete the Minimax but it kept crashing when I went into depth beyond level 4. . I wasn't able to override the board. It kept overriding and stated that your best move is here…can you capture this stone? It would only provide stones. I need more time to figure out how to fix this issue. However, I don't have enough time to figure this out. Even though I'm not using Minimax, I'm still trying to use the best move possible. This was more useful for players; this gives a good option. Also I realized the eye function gave me the best results to capture a stone by showing its liberities for the oppoents stone. If, had more time, I would have combined all these functions.

Furthermore, I am doing the minimax without using a predefined board that has moves to be solved. Since this is limited to AI, this can be difficult and incorrect for certain moves, but I ran out of time to complete the project.

While I was the primary source of feedback during the development process, the iterative nature of the project enabled me to continually improve the AI's capabilities and create a learning tool that beginners could use to enhance their *Go* skills. The AI's supportive and instructional experience aligns with my vision of making *Go* accessible to newcomers while allowing them to appreciate the depth and beauty of the game. Throughout the process, I cultivated a profound understanding of *Go* and its strategic nuances, which has been invaluable in creating an effective and beginner-friendly AI solution.

2.14 Conclusion:

Developing the *Go* game AI helper proved to be an immensely educational journey for me. As a developer, I not only had the opportunity to explore the technical aspects of creating an AI system but also to deepen my understanding of *Go* and its strategic nuances.

Playing numerous games against the AI and evaluating its capturing suggestions provided me with profound insights into the intricacies of *Go* strategy. Through hands-on exploration, I honed my knowledge of capturing techniques and gained a deeper appreciation for the game's complexities.

Furthermore, by designing an AI that catered to beginners, I had to think from a pedagogical perspective. I explored how to present capturing strategies in a clear and instructive manner, fostering a positive learning experience for users. This venture into educational AI development expanded my skillset and inspired me to explore further applications of artificial intelligence in educational contexts.

References

1. Smith, A. (1908, July). The Game of Go: The National Game of Japan. The Plimpton Press Norwood Mass. U.S.A. https://www.gutenberg.org/cache/epub/66632/pg66632-images.html#ch1
2. Lasker, E. (1934). Go and Go-Moku: The Oriental Board Games. In *Amazon* (Second Revised edition). Dover. https://archive.org/details/lasker-go.and.go-moku/page/n1/mode/2up?q=history
3. Bouzy, B., & Cazenave, T. (2001). Computer Go: An AI oriented survey. *Artificial Intelligence*, *132*(1), 39–103. https://doi.org/10.1016/s0004-3702(01)00127-8
4. Müller, M. (2002). Computer Go. *Artificial Intelligence*, 134(1-2), 145–179. https://doi.org/10.1016/s0004-3702(01)00121-7
5. Lee, C.-S., Wang, M.-H., Yen, S.-J., Wei, T.-H., Wu, I-Chen., Chou, P.-C., Chou, C.-H., Wang, M.-W., & Yan, T.-H. (2016). Human vs. Computer Go: Review and Prospect [Discussion Forum]. *IEEE Computational Intelligence Magazine*, 11(3), 67–72. https://doi.org/10.1109/mci.2016.2572559
6. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961
7. Shotwell, P. (2011). Go! More Than a Game. In *Google Books*. Tuttle Publishing. https://archive.org/details/gomorethangame0000shot/page/38/mode/2up?q=rules
8. Cobb, W. S. (2002). The Book of Go. In *Google Books*. Sterling Publishing Company, Inc. https://books.google.co.uk/books?hl=en&lr=&id=Y5UeBT0L0MsC&oi=fnd&pg=PA7&dq=+Japanese+%27Go%27game%27+:+A+Beginners+Guide.+Tokyo:++&ots=5ywORmQD1T&sig=PbcjrdtGbOW5gOe_ZMC6mKdduag#v=onepage&q=Japanese%20
9. Cohen-Solal, Q., & Cazenave, T. (2020, December 19). *Minimax Strikes Back*. ArXiv.org. https://doi.org/10.48550/arXiv.2012.10700
10. Kościuk, K. (2010). Is Minimax really an optimal strategy in games. *Zeszyty Naukowe Politechniki Białostockiej. Informatyka*, *Z. 6*, 63–75. https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-article-BPB1-0047-0018
11. Borovska, P., & Lazarova, M. (2007). Efficiency of parallel minimax algorithm for game tree search. *Proceedings of the 2007 International Conference on Computer Systems and Technologies - CompSysTech '07*. https://doi.org/10.1145/1330598.1330615

12. Carlo, M., Chaslot, G., Saito, J.-T., Uiterwijk, J., Bouzy, B., & Jaap Van Den Herik, H. (2006). *Monte-Carlo strategies for computer Go*. https://www.semanticscholar.org/paper/Monte-Carlo-strategies-for-computer-Go-Chaslot-Saito/b26d04bfd935ceb2a95c473d21901ebc04bfa024