Master's Thesis

Detection and labeling of bad moves for coaching Chinese chess

ZHOU Boyao

Supervisor Ikeda, Kokolo

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

December 2024

Abstract

In recent years, AI technology has advanced rapidly. Technologies such as image recognition and speech recognition are being used across various fields, including healthcare, manufacturing, transportation, distribution, education, and entertainment. Recently, AI has evolved not only in recognizing text, images, and audio but also in naturally generating them, allowing it to increasingly support or even replace human creators and operators.

Games, with their clear rules, accessibility, and ease of evaluation, also require intellectual thinking. Due to these characteristics, games have frequently served as testbeds for artificial intelligence. Indeed, a variety of AI technologies have been developed or evaluated with games as their focus.

When applied to games, the most straightforward goal is to create a strong computer player. This goal has been pursued in academic research for many years, leading to the development of various methods and notable achievements, such as DeepBlue in chess and AlphaGo Zero in Go, which have outperformed top human players. Although research continues to focus on building stronger AI players, interest is gradually shifting toward making strong computer players beneficial for humans. Examples of this include creating human-like computer players, implementing AI that can be played to entertain, automatically generating puzzles and game content, and supporting skill improvement.

Supporting the improvement of beginner and intermediate players is crucial for maintaining or increasing the player base of a game. A coach who provides this support needs more than just "skill in the game." Advanced abilities are required, including explaining the core of the game, identifying the current issues a student faces, guiding them along the path to improvement, and maintaining their motivation. For this reason, many sports and well-known games have established roles for coaches focused on player improvement, where students typically need to pay a fee for quality instruction. If part of this coaching role could be replaced by a computer coach, it could not only promote the development of the game itself but also serve as a meaningful step toward building a better relationship between humans and AI.

Ikeda et al. conducted interviews with Go coaches to investigate the types of guidance provided. They reported that among the various coaching methods, one of the core tasks is reviewing a match after it concludes, pointing out the student's bad moves, and explaining the reasons behind them. The moves considered "bad" are not merely "loss-making" moves. Some moves that incur minor losses are allowed if they serve as bold moves to take control or as safe moves to secure victory. On the other hand, even if a loss is minor, a move that reveals flawed thinking on the student's part is often pointed out. To make the selection and explanation of bad moves replicable by a computer, Ikeda et al. asked Go coaches to annotate game records of intermediate-level players. They requested that for each move, the coach decide "whether to point it out as a bad move" and, if so, "to select the reason from ten categories." Using the collected training data, they applied supervised learning to develop a model for detecting bad moves and a model for reasoning behind them. Although these models perform slightly below the level of a human expert coach, professional players have judged them to be sufficiently practical.

In this study, following the work of Ikeda et al., we undertook a similar approach using Xiangqi (Chinese Chess) as our focus. Xiangqi is a very popular game in China. While its nature shares similarities with Chess and Shogi, it is a fairly different game from Go.

Therefore, to build a model, it was necessary to carry out tasks such as game record collection, reason label selection, annotation, and feature engineering.

We first conducted interviews with several Xiangqi coaches and observed their actual coaching sessions. As a result, we found that "pointing out bad moves and explaining the reasons" is one of the main tasks, similar to Go. However, the nature of the reason explanations was found to be completely different from Go. We first accumulated much natural-language explanations from the coaches regarding their reasons and, through discussions with the coaches, grouped them into about 16 categories. We then requested annotations on 100 game records from intermediate-level players. The collected training data consisted of 4,666 annotations, of which only 798 were marked as bad moves. If this were to be treated as a 16-class problem as-is, some groups would have too few samples, making high-accuracy predictions unlikely. Therefore, we further grouped these into five reason groups for supervised learning.

In supervised learning, when there is a significant imbalance in the number of samples across classes, it often limits the learning performance. In fact, the number of moves identified as bad moves is only about one-fifth of those identified as good moves. Therefore, we decided to augment the training data for bad moves through oversampling.

One critical factor that affects the performance of supervised learning is what input to use. In cases where new board states can be generated indefinitely through self-play, as with AlphaZero, or where an extremely large database is available, as with Maia, high performance can be expected even if the board state itself is used as input. However, the training data we used consists of only 4,666 samples for the detection model and 798 for the reasoning model. For this reason, rather than directly inputting the board state and moves, we need to provide (as input) the "meaning and features in Xiangqi" that they have.

We held discussions with Xiangqi coaches to clarify which aspects of the board state and moves are considered important when detecting bad moves and providing reasoning. These aspects were then quantified in a definable way. Next, we conducted preliminary experiments to investigate how combining these aspects could improve performance—or, alternatively, decrease it due to overfitting.

Additionally, many different models have been proposed for supervised learning, and it is essential to choose the appropriate model based on the characteristics of the input space, output space, and the volume and distribution of the data. After comparing and evaluating numerous models available in the supervised learning tool Weka, we selected RandomForest for the bad move detection model and AdaboostM1 for the reasoning model.

Based on these preparations, the training yielded an accuracy for the bad move detection model (F1-score for bad moves) of 0.825. Although this is slightly lower than the level of agreement among the three coaches (average F1-score of 0.837), it is quite close in performance. The reasoning model achieved an accuracy of 0.501, which slightly exceeded the human average of 0.497. While this result is not statistically significant due to the limited number of samples, it can be considered promising.

Finally, a coach with a professional Xiangqi coaching license evaluated the bad move detection and reasoning results for 10 games. The results from the three coaches involved in training, as well as those from the proposed RandomForest/AdaboostM1 method, were presented in a blind manner and rated on a five-point scale for suitability. The evaluation for bad move detection showed scores of 3.988 for the human coaches and 4.074 for the proposed method. For reasoning, the human coaches scored 4.074, while the proposed method scored 4.048. These results indicate that, in addition to statistical agreement, the quality of our approach is at a level comparable to human coaches.

Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisor, Professor Ikeda Kokolo from the School of information Science at Japan Advanced Institute of Science and Technology. Under his guidance, I received ample support, encouragement, and valuable advice. Even though I encountered many difficulties and setbacks, making numerous mistakes along the way, he always listened patiently and used his extensive experience and profound knowledge to guide me through challenges and help me overcome them.

I would like to thank my parents for always supporting and helping me. I also want to thank my girlfriend, Han Laiyi, who has always trusted me unconditionally and provided me with great help, enabling me to persevere through difficult times. I love you! Finally, I would like to express my gratitude to my lab mates and friends at school for their support, which made my life here very happy and fulfilling. Thank you so much!

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.Introduction

In recent years, Artificial Intelligence (AI) has made tremendous progress and is now widely applied across various industries. From healthcare to finance, AI technologies are fundamentally changing how we address problems and make decisions. Algorithms have become increasingly complex, enabling them to handle complex tasks with higher accuracy and efficiency. Board games are also an important application direction for artificial intelligence. Chinese chess, also known as Xiangqi, ranks among the world's most played board games, boasting a player base in the hundreds of millions [1]. As a strategic two-player game with zero-sum dynamics, its complexity in terms of state-space and game-tree is substantial, estimated at approximately 10^{48} and 10^{150} respectively. These figures position Chinese chess in complexity between Shogi and international chess, indicating a significant depth and challenge in gameplay [2][3][4].

Games have proven to be excellent testbeds for developing and honing AI technologies. They provide controlled and rule-based environments where AI algorithms can be tested rigorously [5]. Games like chess, Go, and even more complex multiplayer online games challenge AI with intricate strategies and decision-making scenarios that mimic real-world complexity.

Most top-performing Chinese chess programs, like QITIANDASHENG and Inspur TS, use alpha-beta pruning techniques similar to those in computer chess. This method evaluates the advantage of board positions by considering factors such as piece characteristics, placement, mobility, threats, protection, and king safety, with specific weights assigned to these factors. It is important to note that the state evaluation function is used to assess the position on the board by analyzing various factors and assigning weights to calculate a numerical value representing the position's advantage. Alpha-beta pruning, on the other hand, aggregates the evaluation values of leaf nodes (the deepest evaluated positions) during the search process to select the best move, and improves search efficiency by pruning suboptimal branches. Typically, the advantage of a position is determined by assigning weights to these specific features [6]. Historically, these evaluation factors were carefully selected, and their weights were manually adjusted by experts. However, as the number of features increased, this manual adjustment process became increasingly complex and time-consuming. Modern chess programs have turned to machine learning techniques, such as neural networks, to automatically adjust and optimize these weights. This has significantly improved the efficiency and effectiveness of position evaluation, enabling the programs to engage in more complex and competitive gameplay.

In recent years, Chinese chess programs have advanced by using neural networks for feature selection and methods like AlphaZero's self-play, improving their performance to rival and even surpass top human players. AIs like "Fine Art" and "Xiangqi Tianji" have consistently outperformed professional players [8], leading to a greater reliance on AI-assisted analysis in training and significantly impacting traditional chess competitions, while still offering potential for innovation in entertainment and education.

Board game software using AlphaGo technology has defeated top professionals, benefitted skilled players but there is little research on programs for ordinary players or beginners. It is important to note that these players also need to enjoy the game and improve their skills by playing against computers. Therefore, programs for entertainment and training of players have great potential and are a promising research direction. To achieve the goal of entertaining or guiding players, we first need to understand how human coaches do this. The following is a brief description of the teaching process. We will explain these processes in detail in Chapter 4.

1) Maintaining Balance: Coaches maintain balance in instructional matches by making gentle yet reasonable moves, preventing quick wins and giving learners more opportunities to think and learn.

2) Appropriate Thinking Time: Coaches adjust thinking time for each move according to the learner's level, ensuring the pace is appropriate.

3) Error Detection and Explanation: Coaches promptly identify learners' incorrect moves and provide explanations using diagrams, text, or verbal communication to help them understand and correct their mistakes.

4) Dynamic Difficulty Level Adjustment: The coach dynamically adjusts the level of game skills taught based on the learner's progress, ensuring that the learner always learns skills appropriate for their current level.

In this research, we focus on how to find bad moves as human coaches. Taking the Chinese chess teaching process as an example, human players often play against stronger opponents and seek feedback on their erroneous moves. Therefore, an ideal coaching computer program should be able to identify incorrect moves, highlight the types of errors, and provide detailed explanations. Additionally, showing professional recommendations on better moves and their expected results would be very helpful. In this research, we specifically address the detection and marking of incorrect moves.

In this research, we initially collected a large amount of user data from Chinese chess beginners, gathering game moves and decisions to analyze typical patterns and common errors made by beginners. After the data collection was completed, we conducted thorough data cleansing to remove any outliers and irrelevant information that might distort the analysis results, thus ensuring the accuracy and relevance of the data. Subsequently, we performed feature calculation, extracting key attributes from the data that significantly impact model performance, such as piece positions, timing of moves, and defensive and offensive strategies employed by the players.

Using the cleaned and structured data, we trained two models under a supervised learning framework: the first model was used to classify each move as "good" or "bad", these "good" or "bad" labels are annotated by experts we invited, providing feedback on the quality of moves to beginners; the second model focused on diagnosing the reasons for bad moves. These analyses were based on the review and data annotation by Chinese chess experts, who provided insights into why certain moves were detrimental to the player's position or strategy.

To validate the effectiveness of our research and the accuracy of the models, we invited several Chinese chess experts to evaluate the results produced by the models. These experts reviewed the models' predictions and their related reasoning to assess their correctness and practical relevance. Their feedback was crucial for refining the models, ensuring that they could effectively help beginners improve their chess skills. This comprehensive evaluation not only confirmed the effectiveness of our approach but also demonstrated the practical utility of applying machine learning techniques to Chinese chess training for beginners.

This thesis is structured as follows. Chapter 2 introduces the game of Chinese chess, including some of its unique rules. Additionally, this chapter discusses the differing perceptions of bad moves between beginners and expert players in Chinese chess. Chapter 3 reviews relevant literature and academic research, defining key terms and contrasting these sources with the methodologies employed in this thesis.

Chapter 4 is dedicated to the methodology of this research. It elaborates on the methods used for data collection, processing, and computation in this study, as well as the approaches and outcomes of feature selection. This chapter also details the results obtained from running the model, specifically the identification of bad moves.

Chapter 5 involves the evaluation and discussion, which includes the selection of experts and the analysis of the results. Finally, Chapter 6 summarizes the findings of the paper.

Chapter 2 Chinese chess (Xiangqi)

Chinese chess (Xiangqi) is one of the most popular board games in the world, with approximately one billion players. The modern form of Chinese chess has a long history, becoming widely popular during the Southern Song Dynasty (1127-1279 AD). The earliest recorded games and theoretical writings on Xiangqi originated from this era.[9]

Chinese chess is a two-player game where both players have access to all information. Approximately 800 years ago, experts began developing knowledge and strategies for the game. Today, there are many outstanding human players worldwide. Despite the game's complexity, the most advanced Chinese chess programs have achieved a level comparable to top human players. Table 1 illustrates the state-space complexity and game-tree complexity of international chess, Chinese chess, shogi, and Go [10],.

Table 1: Computational complexity of different Boardgames

Game	State-space compl.	Game-tree compl.
Go(19 x 19)	10^{172}	10^{360}
Shogi	10^{71}	10^{226}
Chinese Chess	10^{48}	10^{150}
Chess	10^{46}	10^{123}

In this paper, our introduction to Chinese chess will be divided into sections 2.1 to 2.4:

2.1 The Board

The Chinese chess (Xiangqi) board consists of 9 vertical lines and 10 horizontal lines intersecting each other. The pieces are placed on the intersections of these lines and move along the lines. The middle row of the board, which lacks a vertical line, is called the "river boundary".

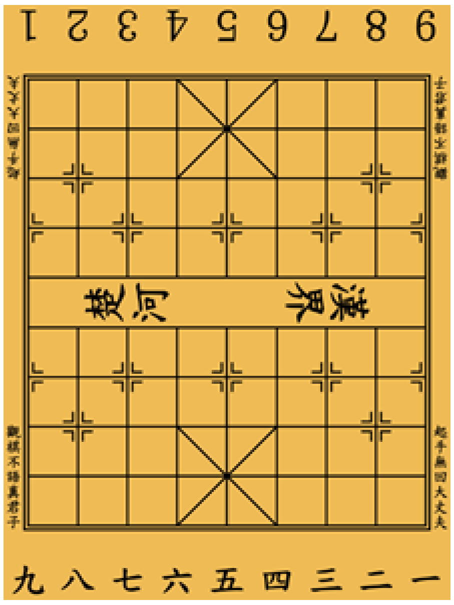
Figure 1: Chinese chess board

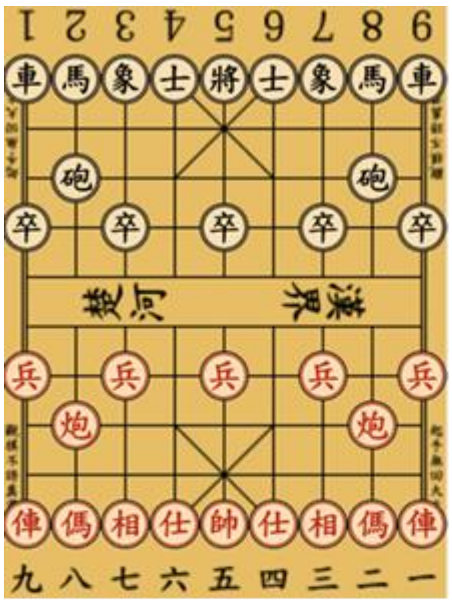
Figure 2: Chinese chess initial settings

In Figure 1, it can be observed that at the bottom of each side of the board, there is an X-shaped area. This area represents the palace, within which the general (or king) moves. The pieces are placed on the intersections of the horizontal and vertical lines and move along these intersections according to specific rules detailed in section 2.2. In Figure 2, we can see Chinese chess initial settings. The pieces for each side are distinguished by color, usually red and black, to indicate the respective players. The objective of the game is similar to that of international chess: to capture the opponent's general. When a player captures the opponent's general, they win the game [11].

2.2 Pieces and Movement Rules

Chinese chess features seven distinct types of pieces. Each player controls a King, two Advisors, two Elephants , two Rooks, two Horses, two Cannons, and five Pawns (Figure x), abbreviated as K, A, B, R, H, C, and P, respectively. The king can move horizontally or vertically, covering one unit distance at a time, represented on the board as moving one intersection at a time. The king is restricted to moving within the palace and cannot leave this area. Additionally, the two kings cannot face each other directly, which means they cannot be on the same vertical line without any intervening pieces. If this occurs, the player making the move will lose the game. Advisors are also confined to the palace, but unlike the king, they can move diagonally within the X-shaped palace. They move one unit at a time, meaning advisors cannot move in horizontally, vertically, they only can move diagonally, which is different from the king's movement. Advisors are also confined to the Palace and move one space diagonally at a time, without the ability to move horizontally or vertically. Elephants are limited to their side of the board and cannot cross

the river. They move
two spaces diagonally per turn, effectively shifting two spaces up or down, and two

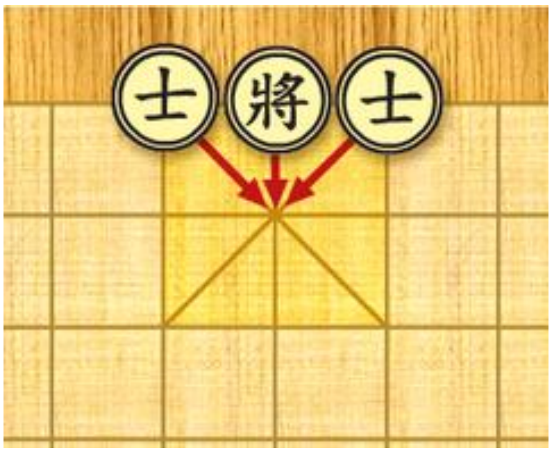

Figure 3: General and advisors

spaces left or right. Elephants can be obstructed; if a piece occupies the space next to an Elephant’s diagonal path, it blocks the Elephant’s movement in that direction.

Elephants cannot move horizontally or vertically. Instead, they move diagonally by two units, as long as they do not cross the river in the middle of the board. If there is a piece occupying the space one unit diagonally from the elephant's current position in the direction it wants to move, the elephant's movement is blocked, and it cannot proceed.

The rook can move horizontally or vertically across any distance, as long as there are no pieces (on either side) obstructing the path between the starting and target points. The knight cannot move horizontally or vertically but can move in an L-shaped pattern, similar to the elephant. This means it moves to a point that is two intersections away in one direction and then one intersection perpendicular, covering a diagonal distance of

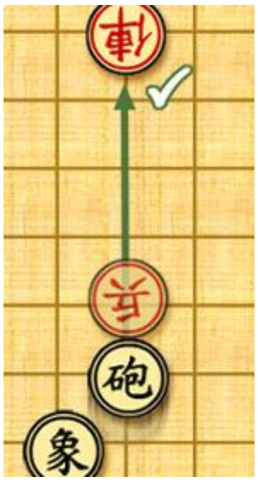

Figure 4: The long-range threat of the cannon

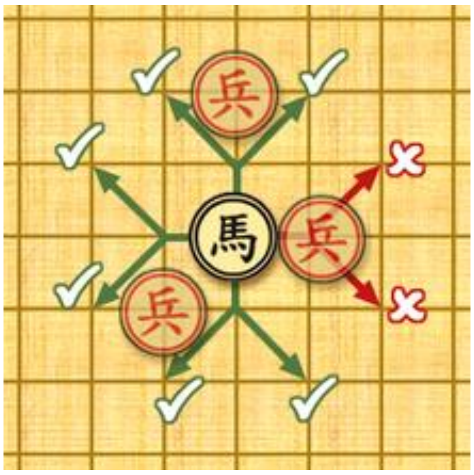

Figure 5: Special movement rules

approximately 2.24 units. If there is a piece in the path of the knight's intended move, it cannot proceed.

Cannons move in a manner similar to rooks, capable of horizontal and vertical movements over any distance. However, capturing an opponent's piece follows a unique rule: there must be exactly one intervening piece between the cannon and the target. This intervening piece can belong to either side. Lastly, soldiers move similarly to the king, advancing one unit at a time either vertically or horizontally. Before crossing the river, soldiers can only move vertically. Once they cross the river, they can move both vertically and horizontally but can only advance forward and not retreat or move backward.

2.3 Recording the Moves of Pieces

In Xiangqi (Chinese Chess), recording a move simply means indicating which piece moves from which position to which position. There are two common methods for this: the "Vertical Line Method" and the "Coordinate Method." Here is a brief explanation of each:

2.3.1 Coordinate Method

This method, commonly used in international chess, assigns a coordinate to each square on the board. A move is determined by the starting square and the destination square, making it more convenient and logical. This method can also be applied to other board

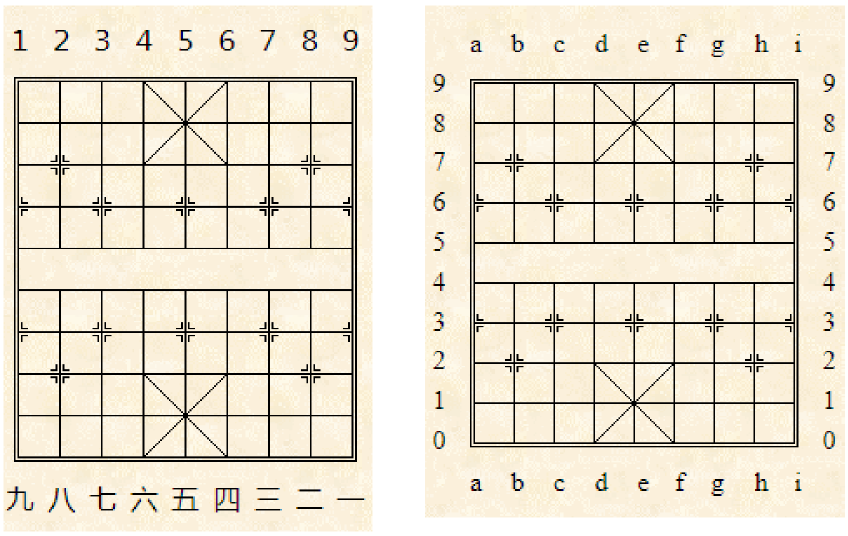

Figure 6: Chinese chess board under different representation methods

games. In Xiangqi, following international chess conventions, the vertical lines (for the red side) are labeled from left to right as a, b, c, d, e, f, g, h, i, and the horizontal lines are labeled from bottom to top as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (as shown in Figure).

2.3.2 Forsyth-Edwards Notation (FEN) File Format

FEN (Forsyth-Edwards Notation) is a specialized standard for recording positions in Xiangqi. Under this standard, a game position can be simply represented with a single line of "FEN format string" rather than laboriously describing the position in words, such as "the red king is on the fourth file from the bottom, and the black king is on the fifth file from the bottom...".

Since it is a text format (involving only a limited set of letters, numbers, and symbols), it is very convenient to transmit game positions over the internet. There's no need to draw

the board on paper or awkwardly describe it in text. Instead, one can set up the position using game notation software, automatically generate the FEN string, and then post it to text-capable online platforms such as web pages, BBS, and electronic forums.
When readers encounter a FEN string, they can use game notation software to "copy and paste" the string and fully reproduce the position, making the exchange of game records quick and efficient.

Representation of Historical Positions

This method is used exclusively in the "Universal Chinese Chess Interface" (UCCI) protocol to represent historical positions. In the UCCI protocol, the position is transmitted using position fen <fen_string>. Since the FEN format string cannot record historical positions, the moves option must be used to specify the moves leading to the current position.

For example, consider the following position transmitted to the engine:

position fen 9/9/3k5/9/9/9/4R4/3A5/8r/4K4 b - - 0 1

In this case, it is acceptable for the engine to make the move R9+1. However, if this position is part of a perpetual check sequence, the interface must provide the historical moves as well:

position fen 9/9/3k5/9/9/9/4R4/3A5/4K4/8r b - - 0 1 moves i0i1 e1e0 i1i0 e0e1 i0i1 e1e0 i1i0 e0e1 i0i1 e1e0

Although this is the same board position, the context has changed. In this scenario, the engine cannot play R9+1 because it would result in a third repetition of the position after K5+1, causing the black side to lose by perpetual check.

2.3.3 Portable Game Notation (PGN) File Format

PGN (Portable Game Notation) is a file format used to record chess games. Given that international chess has adopted this standard for recording game moves, PGN is an excellent choice for Chinese chess, which currently lacks a unified standard. Table 2 is the PGN tags about Chinese chess.PGN files have the following characteristics:

Text Format: PGN files are in text format, which can be created and modified using any text editor.

Structured Sections: PGN files are divided into two main sections: the "Tag Pair Section" and the "Move Text Section". Both sections are well-structured, making them easy to read and convenient for chess software to generate and interpret.

Multiple Games: A single PGN file can contain multiple games, forming a simple chess database.

Open Format: The format of PGN files is open and has been widely accepted by most chess software.

Table 2: PGN Tags for Chinese Chess

PGN Tags for Chinese Chess	
Game	This tag specifies the game type. It is not used in international chess but must be the first tag in a Chinese chess PGN file, and its value must be ″Chinese Chess″

Event	The name of the tournament or event.
Site	The location of the tournament.
Date	The date of the game in the format ″yyyy.mm.dd″
Round	The round number of the game
Red	The player with the red pieces, differing from international chess's ″White″.
Black	The round number of the game.
Result	The result of the game. ″1-0″ indicates a win for Red, ″0-1″ indicates a win for Black, ″1/2-1/2″ indicates a draw, and ″*″ indicates an unknown result.
RedTeam and BlackTeam	The teams (club, chess association, province, or country) of the players, typically written before the Red and Black tags.
Opening	These tags provide information about the opening, variation, and ECCO (Encyclopedia of Chinese Chess Openings) number
FEN	Opening positions, middlegame, endgame, and composed positions

2.4 Other ways to play Chinese chess

Chinese chess has several interesting variants that add new dimensions to the traditional game. One popular variant is "Dark Chess" (also known as "Blind Chess" or "Hidden Chess"). In this variant, the pieces are placed face down on the board at the beginning of the game, hiding their identities from both players. The game starts with players flipping over pieces to reveal their identities, introducing an element of chance and surprise. Players must then strategize based on the newly revealed pieces, making for an unpredictable and exciting game.

Another variant is "Banqi" or "Half Chess," which uses the same pieces as traditional Chinese chess but follows different rules and objectives. The pieces are again placed face down randomly on the board, and players take turns revealing and moving them. Capturing and winning are based on the relative strengths of the pieces as they are revealed.

These variants add layers of complexity and excitement to the classic game of Chinese chess, attracting players who enjoy both the traditional strategy and the added elements of luck and discovery.

2.5 Chinese chess culture

For those looking to improve their Chinese chess skills, there are many ways to enhance their level of play. For children who want to learn Chinese chess, the first step is to understand and master the basic rules. Joining a Chinese chess training class is an effective way to systematically learn chess skills. Practicing with opponents of different skill levels is an important way to test and improve their abilities. Watching expert games and reading classic chess manuals can help them understand tactics and strategies. Additionally, many places offer Chinese chess training classes taught by experienced players or coaches, covering basic rules, tactics, practice games, and competition techniques.

Adults are well-suited for learning Chinese chess, which offers both mental exercise and enjoyment. Many cities offer adult training classes led by experienced coaches, covering theory, demonstration games, practice matches, and game analysis. Classes may adjust based on skill level; for example, stronger players might face restrictions, or beginners may get the first move advantage. Coaches guide learners through rules and tactics, helping them analyze their games to identify mistakes and areas for improvement.

Chinese chess can be played in diverse settings: parks often have dedicated areas, community centers host regular activities, and schools may offer courses or competitions. Online platforms like Xiangqi Online, Chinese Chess Online, and TianTian Xiangqi provide convenient access, offering online matches, AI games, chess puzzles, and live broadcasts. Players can learn through game analysis, real-time matches, and training classes. Apps and software like Xiangqi Wizard and Xiangqi Master support all skill levels, making it easy for beginners to experts to improve skills and enjoy the game.

Chapter 3 Related work

Scientists began researching the field of artificial intelligence in the mid-20th century. In 1950, a paper on the Turing Test had already introduced the concept of artificial intelligence. As society developed, the performance of computers and the capabilities of machine learning algorithms also increased significantly. People moved from endowing computers with logical reasoning abilities to enabling them to acquire knowledge to solve more general problems. This transition was not smooth. The initial algorithms required people to collect vast amounts of knowledge summarized by experts and input this knowledge into computers[12]. The computers would then search for answers in the database when solving problems. This method not only could not guarantee correctness but also was highly inefficient. For game problems like Go, which have a vast search space, there is no complete expert database. Therefore, to solve related problems, machine learning methods emerged, leading to the development of the Monte Carlo Tree Search reinforcement learning method. Automated game programs has also become a core research direction in the field of artificial intelligence. The development of computer gaming technology has gone through three important stages: the initial stage, the growth stage, and the rapid development stage. Over more than a century of growth, the field of computer gaming has developed a variety of machine learning and search algorithms.

Initial Stage: In 1989[13], IBM developed "Deep Thought," which played against the world champion Garry Kasparov and was defeated by a score of 0:2. In 1992, Tesauro and others applied reinforcement learning to backgammon and created the program TD-Gammon. After 1.5 million self-play training sessions, TD-Gammon reached the level of a human master and defeated the human champion with a score of 39:1. In 1993, Professor Chen Zhixing from Sun Yat-sen University developed the Go program "Handtalk," which won the International Computer Go Championship. In 1994, Chinook used a rich database of checkers positions to defeat the human world champion in international checkers, marking the first time a computer program won an official world championship in a game. In 1997, an enhanced version of "Deep Blue" could calculate 200 million moves per second and defeated Garry Kasparov, the world's top-ranked chess player. The program used minimax and pruning algorithms, allowing Deep Blue to search and evaluate up to 12 future moves, whereas human chess players could predict about 10 moves ahead. This advantage in predicting future positions, owing to its immense computing power, led to the computer's victory. This match brought Deep Blue to fame and caused a worldwide sensation, as it represented the first time a computer had defeated the world's best human player in such a complex board game.

Development Stage: From the late 20th century to the early 21st century, through extensive and in-depth research by scientists, the field of computer gaming gradually developed several excellent algorithms, including parallel search algorithms(1998), and genetic algorithms(2000)[13]. These excellent algorithms were also applied by scholars at the time in the development of Chinese chess software. In 2005, Professor Xu Xinhe from Northeastern University in China and his team developed the Chinese chess program "Qitian Dasheng," which possessed the skills of a top-level Chinese chess master and won the championship at the 2006 World Computer Chinese Chess Olympiad. In 2006, Hinton and his students published "Reducing the Dimensionality of Data with Neural Networks" in Science. The paper proposed the method of reducing high-dimensional data to lowerdimensional data using neural networks, sparking a wave of deep learning research in academia. In the same year, the Upper Confidence Trees (UCT) algorithm proposed by Kocsis and others revolutionized the application of reinforcement learning in Go. In 2007, Chinook perfected the checkers database for the best move in every position, making it an unbeatable presence in the field of international checkers[14].

Application Stage: The development of gaming technology has reached a point where there has been a qualitative leap in computer hardware, alongside breakthrough progress in the field of machine learning. The successful application of neural networks in machine gaming has solved many practical problems. In 2012, Google trained a machine learning model, "Deep Neural Networks (DNN)," on parallel computing devices. In 2013, Baidu announced the establishment of the "Institute of Deep Learning" (IDL). In March 2016, the artificial intelligence program AlphaGo defeated the world Go champion Lee Sedol with a total score of 4:1 in a man-machine match. AlphaGo again defeated the world champion Ke Jie with a score of 3:0 in May 2017. The AlphaGo program caused a sensation in the academic world, marking a significant success in computer gaming technology. In March 2020, DeepMind announced that Agent57 had surpassed human players on all 57 Atari games for the first time. Agent57 uses reinforcement learning algorithms, and its scores increase with more computing power. Scientists are now applying these models to other areas, such as electronic sports games that combine multiple technologies, including image recognition. However, these methods have yet to reach the heights achieved in gaming in other fields.

From these milestone events in computer gaming, it is evident that some board game systems have reached or even surpassed the level of top human players.

Historically, the research focus of international board game projects has been mainly on Go and chess. In recent years, as gaming technology has matured, computer gaming in Chinese chess has gradually attracted the attention of many scholars. Its spatial complexity and search complexity are higher than those of chess, making the development of Chinese chess machine gaming more challenging. This challenge has attracted many researchers to this field, resulting in numerous achievements.

In conclusion, the development of Chinese chess gaming technology has made significant progress in recent years. However, due to the late start of computer Chinese chess research, with relevant literature only emerging in the late 1970s, it lags behind international chess research by 20 years and is relatively less mature compared to computer chess technology. Despite the remarkable achievements in the level of Chinese chess software due to the tireless efforts of many chess programming enthusiasts and development teams, chinese chess software capable of defeating top human players has already been designed. Chinese chess is the most widespread and representative board game, holding a crucial position with a large base of players who enjoy human-computer gaming. Therefore, there is a promising future for Chinese chess machine gaming. The research on Chinese chess machine gaming is significant and urgent[15].

In the process of researching and developing Chinese chess software, learning from and drawing inspiration from the development of Go software is crucial and meaningful. For this reason, Chinese chess program designers created the UCCI protocol, establishing a framework for the future development of Chinese chess computer programs. In go, similar to the UCCI standard in Chinese chess software, Go software also has its own protocols and frameworks. For instance, platforms like GNU Go and CGOS (Computer Go Server) provide open and standardized interfaces, enabling different Go engines and interfaces to be compatible and interact with each other. This standardization has facilitated the development of Go AI, allowing researchers to focus on their areas of expertise and driving continuous improvements in Go software.

The current research results are enough to create a Chinese Chess program that is far stronger than human players, still has limitations in terms of entertainment and educational applications for players. While Chinese Chess AI can compete at the highest levels, it often lacks the ability to provide engaging and instructive experiences for casual players and learners. Playing games that are too difficult or against programs that are too strong can be frustrating, while playing games that are too easy or against programs that are too weak can be boring; both scenarios can cause players to lose motivation to continue playing. Some researchers have conducted studies to address these issues, aiming to develop Chinese Chess software that not only excels in competition but also enhances the learning experience and enjoyment for a wider audience.

In the early days, game developers used a technique called Dynamic Difficulty Adjustment (DDA) to achieve dynamic game difficulty adjustments. Specifically, this technique was used in the game Zanac, compiled in 1986, where the game automatically adjusted its difficulty level based on the player's skill, shooting speed, and the current defense level of the spaceship. In other research, some researchers intentionally weakened and restricted powerful programs in order to create less dominant programs. Specifically, Sephton et al. [16] studied several methods for selecting moves in MCTS, one of which is based on the SoftMax strategy. However, a problem with this method is that very poor moves have a chance to be selected as long as their visit count is not zero. To address this issue, Liu et al. [17] introduced a visit count threshold to solve this problem.

In addition to having proper strength, some researchers further utilized human games in their programs to achieve human likeness. For example, In 2013[18], Ikeda and colleagues proposed a computer Go program designed to entertain players by employing various strategies and manipulating the board position. Nakamichi and Ito[19] used shogi amateur games to train evaluation functions. They then replaced the evaluation functions in a strong program with the trained ones to create weaker but human-like programs. As another way to use human players' games to create human-like programs，McIlroy Young et al [20] trained neural networks to predict chess amateur moves.

Based on these, the research field focused on making board games more entertaining and educational for players is continuously developing. In 2015, Kameko and colleagues applied machine learning to generate natural language commentary on Shogi positions. That same year, Ikeda and colleagues used machine learning to learn the natural language terms commonly used by humans to describe moves in the game of Go. In 2016, Ikeda[21] and colleagues further used machine learning to implement the marking and classification of bad moves in Go. Let the program's judgment of good and bad moves be very close to the accuracy of the human coach's judgment of the good and bad moves pointed out by the beginner player game records.

In summary, although Chinese chess and Go differ in their rules and complexity, their trajectories in the field of artificial intelligence development are similar. Both have evolved from basic algorithms to advanced machine learning techniques and shifted the goal from strong players to entertaining or educating programs.

Chapter 4 Methodology

In this chapter we will describe in detail the methods and main experimental procedures used in this research, as well as the results obtained.

4.1 Method overview

In this section, we will give an overview to all the research methods used in this paper, such as observation and interview, data collection and annotation, feature engineering, and learning two models.

Initially, we employed user research methods to determine the ideal teaching process that our program aims to emulate. In the user research process, we used observation and interview methods to investigate our target subjects. Specifically, we interviewed several professional coaches and observed their teaching methods, then took time to discuss with them the typical mistakes[22] made by beginners.

This dataset was newly created specifically for our research, consisting of game records from numerous beginner and intermediate players. After obtaining this dataset, we invited a professional coach to spend time annotating every piece move in each game. The annotations classify moves as either good or bad, including explanations for why each bad move was considered suboptimal. Following this, to enable summarization and generalization of these reasons, we discussed with experts to translate their abstract explanations of poor moves into specific bad move labels. This preparatory work will greatly support our subsequent research.

We also leveraged expert knowledge for feature engineering, as the design and selection of features are crucial for the accuracy of supervised learning. On the one hand, we studied the foundational and complex features selected in prior research and learned methods for calculating them. Additionally, we referenced findings from studies in other board games, such as Go, to observe commonly chosen features in those contexts. Based on feedback and discussions with experts, we extracted and calculated features that can effectively distinguish good moves from poor ones, such as board control scores, positional safety among pieces, and the threat level posed by the opponent’s pieces toward one’s own. Furthermore, we filtered and optimized these features to enhance classification accuracy. This feature engineering approach enables the model to more efficiently evaluate good and bad moves in game scenarios, providing robust support for further classification of the causes behind suboptimal moves.

Finally, we developed two models for detecting and labeling poor moves. The first model is a classification model that identifies whether a move is good or bad based on the extracted features. It uses supervised learning to understand patterns in labeled data and provides a binary output, helping to streamline the initial identification of suboptimal moves.The second model, a cause identification model, goes a step further by categorizing the reasons behind each poor move. This model analyzes the specific characteristics of a move and labels the underlying cause according to predefined categories (e.g., weak board control, lack of piece safety, or high opponent threat). Together, The combination of these two models not only enables automatic detection of poor moves but also helps players understand specific weaknesses in their moves, providing targeted guidance for improving strategy and skills.

4.2 Interviews with human coaches

User research is crucial in Chinese chess studies, offering insights into the experiences and needs of players and coaches during teaching and learning, and providing a scientific basis for research and program development. This study specifically examines how coaches identify bad moves, highlighting the importance of coaching beginners and methods used by coaches to guide them in practice. In this process, we employed both observational and interview methods to clarify the coaching process. Through observation, we directly observed chess classrooms and matches, obtaining real data on teaching methods and student reactions, which helped us understand the dynamics and details of teaching. The interview method involved in-depth discussions with coaches and students to gather their insights and experiences for understanding common teaching issues and the specific needs of students.

Therefore, in this experiment, we interviewed a coach from a Chinese chess school in Shanghai that specializes in teaching beginners. The insights gained from this interview are quite significant and provide valuable reference points for our research.

Based on the interview, there are many ways to improve one's Chinese chess skills, such as watching professional matches, replaying games between professional players, solving Chinese chess endgame puzzles that have been developed over centuries, or reading books on common Chinese chess techniques, openings, and tactics. However, it is generally believed that one of the best ways to improve is to play games with stronger players and review the games with them.By engaging with and learning from experts, players can gain a deeper understanding of strategies and tactics within Chinese chess, rapidly enhancing their skills.Chinese chess coaching is a somewhat effective method and is also popular among Chinese chess players. However, this method is also the most expensive way for players to improve their skills. Coaches typically charge by the hour, and many players cannot afford these fees.

Additionally, beginner and intermediate players might avoid using this method due to the fear that their skills are too low to understand the coach's teachings. If a computer program could provide a similar coaching experience tailored to the player's skill level, it would be extremely valuable and helpful, especially for players at the beginner to intermediate levels.

In observing the teaching process of Chinese chess coach, we learned that the teaching process of human coaches can be summarized into the following four steps, this observation was almost the same to the case of coaching Go [23].first(1) Detect a bad move, then(2) explain why it is bad, referred to as a "label" in this paper,(3)Explain the caused result, at last(4) Show the best move and expected variation. Below, I will use image to explain this process in more detail.

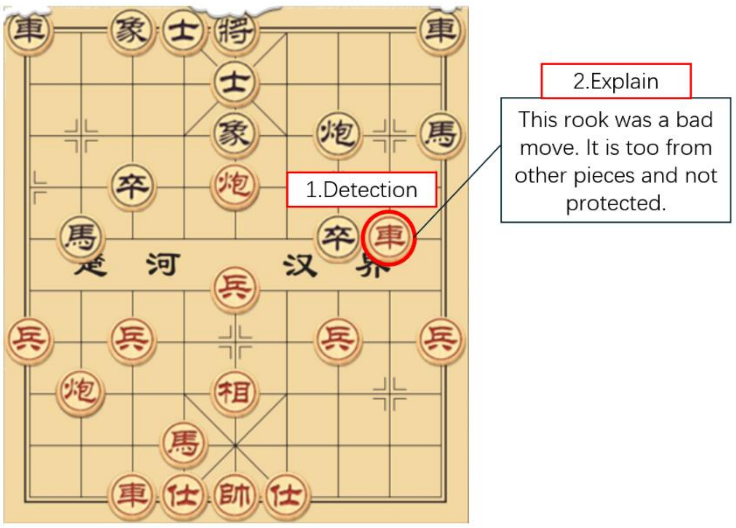
2.Explain
This rook was a bad move. It is too from other pieces and not protected.
1.Detection
楚 河 汉 界

Figure 7:Detect a bad move and explain why it is bad

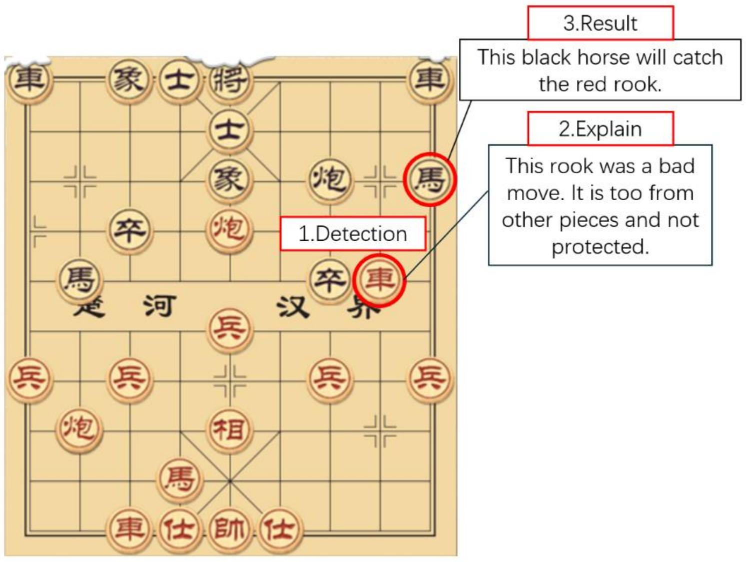
3.Result
This black horse will catch the red rook.
2.Explain
This rook was a bad move. It is too from other pieces and not protected.
1.Detection
楚 河 汉 界

Figure 8: Explain the caused result

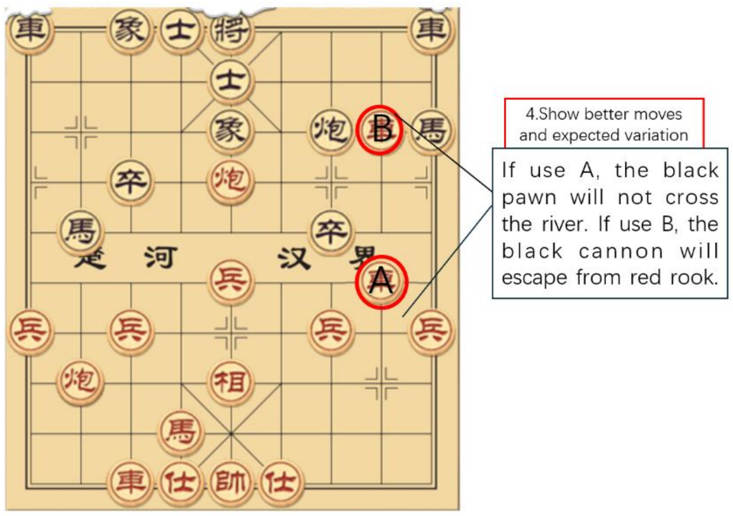
4.Show better moves and expected variation
If use A, the black pawn will not cross the river. If use B, the black cannon will escape from red rook.
B
A
楚 河 汉 界

Figure 9: Show the best move and expected variation

In this paper, we address only the first two steps: the detection of bad moves and labeling with some "reason." In future studies and work, if given the opportunity, I hope to develop a program that can fully execute all four steps.

4.3 Expert knowledge acquisition and annotation

In the field of Chinese chess (Xiangqi) artificial intelligence and research, Expert Knowledge Acquisition and Annotation play critical roles. Expert knowledge acquisition involves extracting strategic insights and decision-making patterns from the games of Xiangqi masters. By analyzing a vast number of historical games, experts can derive key concepts such as classical openings, offensive and defensive strategies, and tactical maneuvers. This expert knowledge is then systematized and converted into rules or models that can be used by Xiangqi AI systems.

For annotation, researchers often label different moves in a game based on its phase—opening, middlegame, and endgame. For instance, a common opening like the "Central Cannon vs. Screen Horse" can be annotated with labels such as "central control" or "king safety" to capture the intent behind specific moves. These detailed annotations enable machine learning models to better grasp the underlying strategies in Xiangqi, enhancing the AI's decision-making capabilities. Expert knowledge acquisition and annotation are essential steps in advancing Xiangqi AI, ensuring it can replicate high-level human thinking in complex games.

Moreover, expert knowledge acquisition in Chinese chess goes beyond simple move replication; it involves understanding the deeper layers of strategy and anticipating an opponent's actions several moves ahead. For this, experts provide not only tactical insights but also broader strategic frameworks, such as positional evaluation, long-term planning, and risk management. These high-level concepts are harder to quantify but are crucial for developing AI systems capable of sophisticated play. By capturing these nuanced elements, the AI can make decisions that mirror a human expert's reasoning rather than just following pre-set algorithms[24].

Annotation, on the other hand, ensures that these expert-level insights are systematically documented for machine learning purposes. This process often includes labeling key turning points in a game, where a specific move significantly shifts the balance of power, as well as documenting mistakes or suboptimal moves for AI training. By combining both expert knowledge acquisition and detailed annotation, the system is capable of learning not only from optimal games but also from flawed ones, improving its ability to handle a wider range of scenarios and opponents. This dual approach is vital for creating a robust and adaptable Xiangqi AI.

4.4 Feature selection

Feature selection is a critical step in machine learning and AI model development, aimed at improving performance by identifying the most relevant features while reducing noise and complexity. One effective approach for feature selection is ablation experiments. Ablation involves systematically removing or "ablating" one or more features from the model to observe how their absence affects performance. By doing so, researchers can assess the importance of each feature in contributing to the overall effectiveness of the model.

In ablation experiments, the process typically begins by training a baseline model with all features included. Then, individual features (or groups of related features) are selectively removed in subsequent experiments. If removing a feature significantly degrades model performance, it indicates that the feature is essential. Conversely, if the model’s performance remains unchanged or improves, that feature might be redundant or even detrimental, suggesting it could be removed to simplify the model and potentially avoid overfitting.

Ablation experiments provide a structured way to identify and eliminate irrelevant or weak features, leading to models that are not only more efficient but also generalize better to unseen data. This method is particularly useful in complex datasets where feature interactions are not always obvious, and traditional feature selection techniques may struggle to distinguish the most informative features.

Building on the foundation of ablation experiments, feature selection becomes a more robust and insightful process, especially when dealing with complex, high-dimensional datasets. By iteratively removing or masking specific features, ablation experiments provide a direct way to assess the contribution of each feature or group of features to the model's performance. This method stands out because it allows for a deeper understanding of how features interact with each other and how crucial they are in the model’s decision-making process.

When conducting ablation experiments, researchers may follow several strategies. One common approach is single-feature ablation, where each feature is removed individually, and the impact on model accuracy, precision, recall, or other performance metrics is evaluated. This provides a straightforward view of which features are most important. However, features often have interactions with one another, meaning that the importance of a feature might depend on the presence of others. To account for this, group ablation can be used, where sets of related features (such as those that represent similar types of data or are highly correlated) are removed together[25]. By doing this, researchers can detect whether certain combinations of features are collectively important.

Another aspect to consider in ablation experiments is the potential trade-off between model complexity and performance. In many machine learning tasks, having too many features can lead to overfitting, where the model becomes too tailored to the training data and performs poorly on new, unseen data. By systematically removing less relevant or redundant features, ablation helps streamline the model, making it more generalizable. This also has practical benefits, such as reducing computational costs, improving model interpretability, and speeding up inference times.

Ablation experiments vs. traditional feature selection methods: While ablation provides a direct, experimental approach to feature selection, traditional methods such as filter methods (e.g., correlation coefficients, chi-square tests) or wrapper methods (e.g., recursive feature elimination) rely more on statistical techniques and heuristics[26]. These traditional methods can be quicker, especially for very large datasets, but they might not capture the intricate dependencies between features as effectively as ablation experiments do. For instance, filter methods typically evaluate features independently, without considering their interactions with other features, whereas ablation experiments directly test the impact of removing features in the context of the model's performance.

Furthermore, ablation experiments can be extended to include hybrid approaches. For example, initial feature selection can be done using a filter or wrapper method to narrow down the number of features, followed by ablation to refine the final feature set. This combination allows for faster feature selection while still maintaining the depth of insight provided by ablation experiments.

In more complex models, such as deep learning architectures, ablation experiments can be applied to specific layers or components of the model. For instance, in convolutional neural networks (CNNs), researchers may perform layer-wise ablation, where entire layers or even specific filters are removed to assess their significance in feature extraction and decision-making. This can provide valuable insights into the internal workings of the model and highlight which parts are crucial for performance and which are redundant or underperforming.

In summary, ablation experiments are a powerful tool for feature selection, providing a practical and insightful approach to understanding which features (or combinations of features) drive model performance. By iteratively removing features and observing the impact on results, researchers can fine-tune their models, improve generalization, reduce overfitting, and simplify model complexity. When used in conjunction with traditional methods, ablation experiments offer a comprehensive strategy for optimizing feature sets in machine learning applications.

In addition to their role in identifying critical features, ablation experiments can also serve as a diagnostic tool to detect hidden biases or weaknesses within a model. When certain features are removed and the model's performance drops disproportionately, it can indicate that the model is overly reliant on those specific features, which might not generalize well to new data. This is particularly important in real-world applications where datasets often contain noise, outliers, or imbalances that can mislead the model during training. By using ablation experiments to systematically assess the importance of each feature, researchers can uncover dependencies that could result in biased or unstable predictions and take corrective actions, such as rebalancing the data or refining feature engineering processes.

Moreover, ablation experiments can be employed to evaluate the robustness and resilience of machine learning models. In many practical applications, models are deployed in environments where data can be incomplete, noisy, or subject to unforeseen changes. Ablation allows researchers to simulate such scenarios by artificially removing or corrupting parts of the data and observing how well the model adapts. A robust model should be able to maintain reasonable performance even when key features are missing or degraded. If ablation reveals that the model's performance collapses in the absence of certain features, it may indicate that the model is too fragile and needs further refinement, such as through regularization, improved data preprocessing, or additional training with more diverse datasets.

Lastly, ablation experiments can be invaluable in interpreting black-box models, such as deep learning networks, which are often criticized for their lack of transparency. By systematically removing inputs or altering the architecture in ablation studies, researchers can gain insights into how these complex models make decisions. For instance, in computer vision tasks, ablation can be used to determine which specific parts of an image are most influential in the model's prediction by masking different regions and analyzing the resulting changes in output. Similarly, in natural language processing (NLP), ablation experiments can help pinpoint which words, phrases, or linguistic features are most critical to the model's understanding and prediction. This not only enhances the interpretability of the model but also aids in debugging and optimizing the learning process by revealing which parts of the data or model architecture are underperforming.

In conclusion, feature selection through ablation experiments provides a versatile and robust method for refining machine learning models. Beyond simply identifying important features, ablation allows for a deeper understanding of feature interactions, model dependencies, and potential weaknesses. By iteratively experimenting with feature removal, researchers can make informed decisions about which features to keep, which to discard, and how to balance model complexity with performance. This method also enhances the interpretability and robustness of models, making it an essential tool in the development of both traditional machine learning systems and more advanced deep learning architectures. When applied thoughtfully, ablation experiments offer a clear path toward creating more efficient, reliable, and interpretable models capable of handling real-world challenges.

Chapter 5 Experimentation and Evaluation

In this section, we show four series of experiments. The brief content is as follows:

1) Baseline Experiment provides criteria for the results obtained by subsequent evaluation procedures.
2) Learning of bad move detection system, and comparison with huma's decision.
3) Learning of bad move labeling system, and comparison with human's decision.
4) Invite human experts to evaluate the results.

5.1 Baseline Experiment

In this experiment, we will invite three strong human players, compile their annotation data, and compare and evaluate the matching rate between the annotation results of each pair of players. The specific procedure involves each coach independently analyzing one or more chess games and marking the moves they consider to be bad. Afterward, the other two coaches evaluate the same games independently, determining whether they agree with the bad moves identified by the first coach. In this research, Human Player A labeling all 100 game records, while Human Players B and C labeling 20 out of the 100 games for comparison with Human Player A.

The core of the experiment lies in comparing the degree of agreement between the three coaches. By counting the bad moves identified by each coach and the number of those moves agreed upon by the other two coaches, the level of consistency between each pair of coaches is calculated as a percentage. For example, if Coach A identifies 10 bad moves, and 8 are agreed upon by Coach B while 7 are agreed upon by Coach C, then the agreement between Coach A and Coach B would be 80%, and between Coach A and Coach C, it would be 70%. Similarly, Coach B and Coach C′s agreement can be calculated in the same manner. This mutual evaluation among the three coaches allows for a comprehensive understanding of their consistency in evaluating bad moves. By comparing the agreement levels between each pair of coaches, the experiment can further analyze which types of errors are more likely to reach consensus and which are prone to varying judgments[27].

This multi-coach evaluation method reveals more details, especially regarding differences in assessment standards among the coaches. Some bad moves might be universally recognized by all three coaches, while others may be agreed upon by only some, highlighting the complexity or controversy of these moves.

Table 3: F-measures (good/bad/total) among three coaches

Players	F-measures
Player A for B	0.908/0.411/0.839
Player B for C	0.863/0.353/0.805
Player C for A	0.920/0.455/0.866

Average	0.897/0.406/0.837

Table 3 shows the F-measures (good/bad/total) among three coaches. We can see that only around 40% of bad moves pointed by one player are also pointed by another player.

This result shows how challenging this task is and how diverse policies coaches have.

It also provides a specific goal for our subsequent experimental results. These two points are very important to our research.

5.2 Training Preparation

Two models for detection and labeling are trained through supervised learning using annotated data.

To accomplish this, the data was divided for validation, adjustments were made to correct the imbalance between positive and negative samples, after get the new dataset, a learning method is selected, and feature selection is performed..

5.2.1. Training Data

The original data we collected consists of 100 matches between novice players, with a total of 4666 moves. Due to the small sample size, directly splitting the dataset may result in insufficient data in the training or validation sets, which could affect the model's generalization ability. To address this issue, we adopted 10-fold cross-validation[28]. In this method, the dataset is evenly divided into 10 subsets. In each iteration, one subset is used as the validation set, and the remaining nine subsets are used for training. This process is repeated 10 times, ensuring that each subset is used once for validation and once for training. The overall model performance is evaluated by averaging the results of all.

5.2.2. Correction of Positive and Negative Sample Imbalance

During the process of data set segmentation and sample balancing, we found that only 798 of the 4666 samples were negative samples, resulting in an uneven distribution of categories. To alleviate the bias problem caused by this imbalance, we adopt an oversampling method to increase the number of negative samples to the same level as the positive samples. Specifically, the proportion of positive and negative samples in the data set is balanced by copying existing negative samples or generating synthetic samples so that the number of negative samples reaches the same number as positive samples.

To confirm the effectiveness of over-sampling, we conducted a preliminary experiment. The employed learning method is Multilayer Perceptron in a free machine learning platform, Weka version 3.8.6[29].

To evaluate the performance of the model, we use F-measure, a comprehensive metric that combines precision and recall. When we do not do any resampling of the model, we can get the confusion matrix in Table 4 :

Table 4: Confusion matrix of original dataset

good by coach	bad by coach	
3181	249	estimated as good
492	106	estimated as bad

The F-measure is 0.912/0.222/0.823 (good moves, bad moves, weighted average). We can see that the performance is very poor when judging bad moves to be bad. Because the number of positive samples in the data set is much higher than the negative samples, this result is reasonable.

When we resample the data set, the results we get are as the confusion matrix in Table 5:

Table 5: Confusion matrix of resampled dataset

good by coach	bad by coach	
2944	1123	estimated as good
538	3529	estimated as bad

After observing Table 5, we can know that when the multi-layer perceptron is also used for testing, the result of F-measure is 0.780/0.809/0.795. The result shows that the ability to identify wrong moves has been improved, which proves that we are targeting Preliminary processing of the data set makes sense.

5.2.3. Model selection

In this section, we describe the process of selecting several promising models for classification. and comparing their performance using the Weka software. Initially, we begin by exploring Weka's extensive classifier library, which includes options like decision trees, support vector machines (SVM), and neural networks. The choice of models was selected based on their performance on our dataset.

Next, we proceed to fine-tune the parameters of each selected model to enhance its performance. For instance, we adjust the depth of decision trees, the kernel type for SVMs, and the number of layers in neural networks. This parameter optimization step is crucial for improving each model’s accuracy and adaptability to our dataset[30].

Following model training, we employ various evaluation metrics in Weka to compare the model’s effectiveness. Metrics such as accuracy, recall, and F1-score are used to gauge how well each model performs on our classification task. So, we tried a variety of classifiers with the goal of narrowing down the choices, selecting a few classifiers with better performance and using them in subsequent experiments.

We conducted a preliminary experiment for this selection. Its setting is different from what we have explained. Specifically, only one supervised learning trial was done for each model, using fixed training data (70%) and test data (30%). In other words, 10-fold cross validation was not employed. Further, we didn't employ oversampling. To better approximate real-world application scenarios, we manually divided the dataset into training and test sets. The split was performed using Python [X_train, X_test, y_train, y_test = split_data(size=0.3)]function. The classifiers were then trained and tested on imbalanced samples to ensure that the selected models perform well in the context of this project.

The experimental results in the following table are obtained. In this experiment a positive example is a bad move The evaluation index is weight F-measure results of bad moves. We will select the top 10 classifiers for subsequent experiments. The results are summarized in Table 6.

Table 6: All available classifier results on weka 3.8.6

classifiers	TP	FP	FN	TN	Accuracy	Precision	weight F-Measure
RandomForest	22	161	56	1176	84.66%	12.02%	0.819
RandomCommittee	20	163	59	1173	84.31%	10.93%	0.815
RandomSubSpace	28	155	109	1123	81.34%	15.30%	0.802
RandomizableFilteredClassifier	37	146	132	1100	80.35%	20.22%	0.800
RandomTree	37	146	144	1088	79.51%	20.22%	0.795
AdaboostM1	38	145	155	1077	78.80%	20.77%	0.790
LMT	46	137	183	1049	77.39%	25.14%	0.784

Baggiing	54	129	201	1031	76.68%	29.51%	0.782
PART	41	142	189	1043	76.61%	22.40%	0.777
J48	48	135	203	1029	76.11%	26.23%	0.776

5.2.4. Employed Features

In this section, we employ the Incremental Feature Enhancement approach to systematically improve model performance by gradually adding new features. This method allows us to evaluate the impact of each additional feature set on the model's accuracy, ensuring that each feature contributes meaningfully to the detection of bad moves.

Initially, we train the model using distance feature group as a baseline, obtaining a preliminary performance measurement. From this baseline, we then incrementally introduce additional feature groups, such as piece relationships, move evaluations, and spatial positions. With each iteration, we assess the model’s performance metrics, including accuracy, F1 score, and recall, to gauge the effectiveness of the added features[31].

In this experiment, we first made the following settings. Based on past researchers’ references to Chinese chess research and other chess studies, we selected 29 features from features that can be obtained directly in the program and features that are calculated. Conduct experiments, in which we divide these 29 features into 6 types for experiments.

Table 7: Employed features result

features	F-measures
A. distance feature group	0.853/0.269/0.779
B. A+Chess piece relationship evaluation score	0.873/0.312/0.802
C. B+ Traditional program assessment scores	0.894/0.306/0.819
D. C+Winning/loss rate	0.897/0.332/0.825
E. D+ Piece weight assignment	0.918/0.235/0.831
F. E+ Neural network evaluation score	0.934/0.247/0.847

The results, as shown in Table 7, indicate that certain features, such as those derived from neural network evaluations and piece interactions, lead to notable improvements in performance. For instance, adding features related to piece distance and positional evaluations significantly boosted the F-measure, particularly in the detection of bad moves. This demonstrates the importance of incorporating domain-specific insights, as these features capture the unique dynamics of Chinese chess that simpler features may overlook.

Here, we take the random forest as an example to conduct ablation experiments on this classifier, dividing the 29 features into seven categories: A, B, C, D, E, F, and G. Detailed information will be provided in the appendix.

5.3 Training result

5.3.1. Machine-Learning for Detection

In the previous section, we observed that several additional features effectively improved detection accuracy using 10-fold cross-validation. In this section, due to the limitation of learning and analysis time, the training and test sets are manually separated, and the performance on the test data is compared with that of strong human players(result like Table 2).

According to the preliminary experiment, we employed Random Forest as the learner. Not only is its robust and accurate estimation, but also its output of "feature importance" is valuable to analysis.

For our experiment, we used train_1 as the training set, containing 2844 good moves and 519 bad moves, and test_1 as the test set, with 1223 good moves and 177 bad moves. Observing an imbalance in the number of good and bad moves in the training set, we applied a resampling method to increase the number of bad moves to equalize them with the good moves at 2844 each. According to Ikeda et al.'s study[21], this approach proves more effective than removing good moves, ensuring a balanced dataset that enhances model training.

Table 8: F-measures of good/bad detection

Players	F-measures
Player A for B	0.908/0.411/0.839
Player B for C	0.863/0.353/0.805
Player C for A	0.920/0.455/0.866
Average	0.897/0.406/0.837
Random forest	0.872/0.332/0.825

The resulting F-measure values are 0.872/0.332/0.825. It can be seen that the model's performance is slightly below the average level of strong human players but outperforms Player B's evaluation of Player C. This suggests that, compared to skilled human players, the bad moves detected by the newly trained model exhibit a certain degree of rationality.

5.3.2. Machine-Learning for Labeling

The second model assigns a type label to each detected bad move. We have a total of 598 instances of bad moves. As with the training process for the first model mentioned earlier, we conducted some preliminary experiments to select a classification method and reference features. After comparing several methods available in Weka, such as J4.8, LADTree, SMO, and Multilayer Perceptron, we selected "AdaboostM1" as the classification method. When using all 29 features, the overall F-measure (average of 10-fold cross-validation) is 0.413. In contrast to the incremental feature enhancement shown in 5.2.4, we employed an ablation optimization of features.First, the baseline model was learned with using all 29 features.

Then, we removed one (group of) feature and retrain the model.

To further analyze and optimize this model, we applied ablation experiments. By systematically removing specific groups of features, we could assess their individual contributions to the model’ s performance. This process allows us to identify which features are most impactful for accurately labeling types of bad moves, thus refining the model and potentially improving its classification accuracy without unnecessary complexity. Then, we remove one set features and retrain the model. The specific details of the experiment are as follows:

The experimental process is as follows: first, a baseline test is conducted using the full feature set to establish the model’ s optimal performance. Then, in each subsequent experiment, we remove one feature or group of features and retrain the model. By comparing changes in accuracy, F1 score, and other performance metrics, we can evaluate the contribution of each feature. This approach enables us to identify the features that most significantly enhance model performance, optimizing feature selection strategies and improving model efficiency.

Through the experiments, we observed that removing certain features led to an improvement in the F-measure of the second model. After removing some features from the six types of feature, the F-measure showed a slight increase, and after further adjustments, the F-measure reached 0.437.

Next we need to divide the data set into a training set and a test set as we did before, and compare the obtained F-measure with the F-measure value of a powerful human player.

For our experiment, we used train_2 as the training set, containing 1617 good moves and 217 bad moves, and test_2 as the test set, with 869 good moves and 109 bad moves. As a comparison baseline, we selected two games with a total of 127 moves, including 96 good moves and 31 bad moves, and had strong human players A, B, and C label them. After obtaining the F-measure, the model then labeled the baseline game records as well. The F-measure results from the model were subsequently compared with those from the human players.

Table 9: F-measures for AdaboostM1

Players	F-measures
Player A for B	0.487
Player B for C	0.574
Player C for A	0.432
Average	0.497
AdaboostM1	0.501

The overall F-measure for human players is shown in Table 9, with an average F-measure of 0.497, indicating that even among strong human players, there was no consistent agreement on the labels for errors. The F-measure obtained by AdaboostM1 is 0.501, which is better than the human average. However, since the number of moves in the baseline test is relatively small, with only 127 moves, this result may not be universally applicable.

For the labeling system, it is inherently more challenging than the detection system, as the output is not binary but multi-class. The subcategories include more than ten different error reasons, and when summarized, there are five main categories. Additionally, the number of training samples remained unchanged, which may also impact the accuracy of the labeling system compared to the detection system. If we were able to obtain more training data, the results of the second system might improve.

5.4 Result evaluation

In Sections 5.3.1 and 5.3.2 the F-measure is primarily used to evaluate the results, comparing them with the average F-measure of strong human players. However, the F-measure cannot fully capture whether a bad move has occurred accurately. To accurately evaluate move quality, it’ s essential to recognize varying levels of error severity. Some moves may seem reasonable for beginners but get misclassified as poor, while strong players might overlook certain subtle errors. Similarly, errors differ in impact: just as mistaking a monkey for a gorilla is less critical than mistaking a human for a monkey, distinguishing move mistakes by severity can prevent inappropriate generalization and improve labeling accuracy.

Therefore, we decided to invite a strong human player to provide a final evaluation of the experimental results. The specific procedure is as follows: we first invited the three strong human players mentioned previously to label ten new game, Gamerecord_1. After obtaining Gamerecord_1, we used the system from Section 5.3.1 to detect bad moves, followed by the system from Section 5.3,2 to label the bad moves. The labeling results were then recorded. Then we will get Gamerecord_1 through Gamerecord_10.

After obtaining these game records, we invited another professional coach, D, who holds a certified Chinese chess coaching license, to evaluate these labels. The evaluation was conducted in the form of a questionnaire, with each move rated on a scale of 1 to 5, representing the following:

(1) [Completely incorrect label]to (5)[Completely correct, highly valuable for beginner players].

Coach D scored the judgments and labels made by the three strong human players and the two models based on his assessment. The final results are as follows:

Table 10: Evaluation scores by a professional, for bad move detection

Players	Professional evaluation
Player A	4.162
Player B	3.832
Player C	3.96
Average	3.988
Our method	4.074

Table 11: Evaluation scores by a professional, for type labeling

Players	Professional evaluation
Player A	4.122
Player B	4.014
Player C	4.086
Average	4.074
Our method	4.048

Table 10 shows the scores for (1) bad move detection, and Table 10 presents the scores for (2) type labeling. We found that the score differences among the strong human players A, B, and C were minimal, which indirectly confirms that there is a small skill gap between the human players participating in our experiment.. Our model scored higher than the average, indicating that it is usable.

In the Table 11, we observe that due to the reduced sample size, Table 11 shows the scores for type labeling. We can see that our method perform as good as human coaches, and achieve acceptable level. Although our model's results were slightly below the average human score, they were better than the lowest-performing human player, which we consider an acceptable outcome.

Based on these evaluation results, we believe that our method is promising and potentially usable to some extent. For bad move labeling, if we can obtain more data(this means more game records and annotators.), we anticipate that the results would improve further.

Chapter 6 Conclusion and Future Work

As powerful computer players are capable of conducting numerous games, their potential in entertainment and guidance has emerged as a new topic in artificial intelligence. This paper presents the design and development of a system capable of detecting bad moves which are pointed by human coaches, along with a labeling system for annotating these moves. The interviews with three Xiangqi coaches and the analysis of the annotation results revealed that even among human coaches, the level of agreement on points to highlight is not very high. It also became clear that, to enable accurate judgments by the computer, it is necessary to design features that fully represent the characteristics of the Xiangqi board state and moves.. We collected 4,666 moves labeled by skilled players, extracted 29 features, and applied a two-step supervised learning approach.

The quality of detection and labeling in this system was assessed by professional coaches. Results show that, while both systems perform slightly below the level of top human players, they have reached a usable level.

In future work, we plan to expand the training dataset, as supervised learning requires extensive features and data to reduce overfitting. Additionally, we aim to explore other aspects of game coaching. Developing an explanatory system to complement or replace the current guidance based on optimal moves would also be valuable. After detecting and labeling bad moves, further analysis to recommend the best alternative moves would greatly enhance the system. Guiding human players to grasp abstract concepts remains a complex and meaningful area for research.

Bibliography

[1] Shi-Jim Yen,Jr-Chang Chen,Tai-Ning Yang, Shun-Chin Hsu. COMPUTER CHINESE CHESS. (2003)

[2] Zou Lei, Zhang Xianfeng. Artificial intelligence and its application. Information Network Security, 11-13 (2012).

[3] Ji Hui, Ding Zejun. An improved Monte Carlo tree search algorithm for two-player game problems. Computer Science, 140-143.(2018).

[4] Shang Yuhong. Research on the prehistory of game theory. Northwestern University, (2003).

[5] Tian Yuandong. A brief analysis of Alphago system. Acta Automatica Sinica, 671-675 (2016).

[6] Chen Dong-Bin, Shao Kun, Zhu Yuan-Heng, Li Dong, Chen Ya-ran, Wang Hai-Tao, LIU De-Rong, Zhou Tong, Wang Cheng-Hong. Overview of deep reinforcement learning: On the development of Computer Go. Control Theory and Applications, 2(2016).

[7] Pei Yangfang.Design and implementation of China Xiangqi self-playing chess and reinforcement learning system(2019).

[8] https://sports.sina.cn/others/qipai/2016-05-10/detail-ifxryhhh1833447.d.html (2016).

[9] Banaschak: A story well told is not necessarily true - being a critical assessment of David H. Li's "The Genealogy of Chess"(1998) .

[10] Xu Xinhe.Wang Jiao,Luo YanHong and Wang Tao. Implementation of Adaptive Genetic Algorithm for the Evaluation Function in Chinese Chess Computer Game Systems. Journal of Northeastern University, (2005).

[11] Xu Xin and Wang Jiao. Analysis of Key Technologies in Computer Game Playing . Mini-micro Systems, (2006) .

[12] Liu Zhiqing, Wu Xiuzhu. Interpretation of Artificial Intelligence Technology Behind AlphaGo. Control Theory and Applications, (2016).

[13] Xu XinHe, Deng ZhiLi, Wang Jiao, et al. Various Challenges Facing Machine Game Research [J]. CAAI Transactions on Intelligent Systems, (2008).

[14] Xu Xinhe, Zheng Xinying. Board Games and Event Countermeasures [J]. Control and Decision(2007).

[15] Wang Jiao, Xu Xinhe. Computer Game Playing: A Frontier Field of Artificial Intelligence – National College Computer Game Competition [J]. Computer Education, (2012).

[16] Sephton, T., et al. "Exploring Move Selection in MCTS with SoftMax Strategy." Proceedings of the NeurIPS Conference,2020).

[17] Liu, Z., et al. "Implementing a Visit Count Threshold in MCTS for Enhanced Move Selection." Journal of Artificial Intelligence Research, (2021).

[18] Kokolo Ikeda, Simon Viennot, Production of Various Strategies and Position Control for Monte-Carlo Go - Entertaining human players, IEEE Conference on Computational Intelligence and Games (CIG2013), (2013).

[19] Takafumi Nakamichi, Takeshi Ito. Adjusting the evaluation function for weakening the competency level of a computer shogi program. ICGA Journal,(2018).

[20] McIlroy-Young, R., et al. Aligning Superhuman AI with Human Behavior: Chess as

a Model System.KDD2020, (2020).
[21]Kokolo Ikeda, Simon Viennot and Naoyuki Sato, Detection and Labeling of Bad Moves for Coaching Go, IEEE Conference on Computational Intelligence and Games (CIG2016),(2016).
[22]Cai Shen. An Improved Pruning Strategy for Chinese Chess Machine Game Playing. Foreign Electronic Measurement Technology, (2016).
[23]Hinton, G. E., Osindero, S., & Teh, Y.-W. A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, (2006).
[24]Guo Xiaoxiao, Li Cheng, & Mei Qiaozhu. Application of Deep Learning in Games . Acta Automatica Sinica, (2016).
[25]Liu Wei. Research on Imperfect Information Machine Game Based on CNN and MCTS [D]. Xidian University,(2019).
[26]Gao Qiang. Research on the Complexity, Theoretical Solution, and Related Search Algorithms of Computer Game Playing. Northeastern University, (2016).
[27]Xie Yanru. Research and Implementation of Data Structure and Evaluation Function in Chinese Chess Computer Game Playing. Xi′an University of Technology, (2008).
[28]Li, F., & Du, Y. "From AlphaGo to Power System AI: What Engineers Can Learn from Solving the Most Complex Board Game," IEEE Power and Energy Magazine,(2018).
[29]Weka(software).In Wikipedia. Retrieved [Date], from https://en.wikipedia.org/wiki/Weka_(software)
[30]Gelly, S., Wang, Y., Teytaud, O., et al. Modification of UCT with Patterns in Monte-Carlo Go, (2006).
[31]Wang Yifei. Research and Implementation of a Self-learning Function in Computer Chess Game Playing System. Harbin Engineering University, (2007).
[32]Iida, H., Sakuta, M., & Rollason, Computer Shogi. Artificial Intelligence(2002).

Appendix A

In Chapters 4 and 5 of this article, the 29 features used in this experiment are mentioned. Here is a brief introduction to all the features. In this experiment, all features are divided into 6 types, All the characteristic values mentioned below can be obtained from the console of powerful Chinese chess software such as Pikafish.

Type 1: Distance feature group

Distance1: Distance from enemy king.

Distance2: Distance from ourselves king.

Distance3: Distance from best move from Chinese chess program.

Distance4: Distance from best move from Chinese chess program.

Distance5: The distance from the previous move of the chess piece.

Type 2: Chess piece relationship evaluation score

Protection score: The value score of pieces that protect another piece

Capture score: The value score of the piece that captures the opponent's piece

Black moving score: Score after black's move from Chinese chess program.

Red moving score: Score after red's move from Chinese chess program.

Black block moving score: Black's value points for preventing other pieces from moving.

Red block moving score: Red's value points for preventing other pieces from moving.

Type 3: Traditional program evaluation scores

Round: length of Chinese chess gamerecord.

Type: There are seven types of chess pieces in total

Value: Basic value points of chess pieces

Depth: The number of moves (ply) the engine looks ahead in its analysis to predict potential outcomes.

Seldepth: The selective depth, or the maximum depth reached within specific lines the engine prioritizes as more relevant.

Multipv: Refers to "Multi Principal Variation," indicating multiple best move options explored by the engine simultaneously.

Score Mode: The mode or setting defining how scores are represented (e.g., centipawns, win probability) in evaluation outputs.

Score: The numerical evaluation value indicating which side has an advantage and by how much, typically measured in centipawns.

Color: color of chess piece

Type 4:

Winning/loss rate: Changes in winning percentage when piece move.

Type 5: Piece weight value

Cumulative Move Value: In Chinese chess, if a piece remains in play, it will accumulate move value over time.

Mobile efficiency: Move Efficiency: Due to the differences between types of pieces, each piece has a distinct move efficiency.

Cumulative Capture Value: The change in a piece's value after capturing an opponent's piece.

Type 6:Neural network evaluation score

NNUE Evaluation: A neural network-based evaluation method that assesses positions using pre-trained parameters, enhancing chess engines' accuracy in evaluating complex positions.

Bucket: A grouping mechanism that organizes similar positions or evaluations into categories for streamlined processing.

Material: An evaluation component that measures the value of all remaining pieces on the board.

Positional: An evaluation that considers a piece's placement and influence on the board, beyond just material value.

Final Evaluation: The concluding assessment of a position after all calculations, used to determine the overall advantage or disadvantage.

Appendix B

After discussions with powerful Chinese chess human players, we summarized the following labels (for humans) for labeling bad moves, which are classified into six types in total. The details are as follows:

Type 1: Multiple Moves for One Piece

a. Moving a piece back to its original position

b. Could complete the move in one step but spent extra turns

c. Did not account for cumulative movement value (multiple moves for the same piece)

Type 2: Violation of Opening Principles

a. Using low-value pieces in the opening (not moving major pieces)

b. Failing to balance the board left and right

c. Not occupying effective terrain (missed opportunity to position pieces in high-value locations)

d. Leaving a favorable position

Type 3: Unfavorable Exchange or Accepting Opponent's Exchange Invitation

a. Exchanging a high-value piece for a low-value piece

b. Losing position (resulting in a poor shape/reduced attack speed)

Type 4: Mistakes in Attack

a. Incorrect piece choice (attacking with low-mobility or low-threat pieces or using such pieces for tactical tasks)

b. Correct piece choice but targeting the wrong location

Type 5: Piece Loss (Beginner Mistake, Direct Loss of Piece)

a. Losing a piece

b. Losing the general (checkmate blunder)

Type 6: Greedy Attack or Over-capturing

a. Capturing a piece but losing initiative

b. Capturing a piece but losing position (resulting in a poor shape/reduced attack speed)

c. Capturing a piece but getting captured in return