ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

National and Kapodistrian University of Athens

Department of History and Philosophy of Science
&
Department of Informatics and Telecommunications

Interdepartmental Graduate Program:
Science, Technology, Society—Science and Technology Studies

MSc Thesis

"Chess engines: from their historical emergence to contemporary experiences"

Andrianna Anastasopoulou

Thesis Advisory Committee:
Stathes Hadjiefthymiades, Professor (advisor)
Dimitris Varoutas, Associate Professor (member)
Manolis Simos, Adjunct Faculty (member)

Athens, 2023

Chess engines: from their historical emergence to contemporary experiences

Abstract

In the vast expanse of intellectual games, chess stands as a timeless testament to strategic thinking and prowess. This research delves into the transformative journey of chess in the digital age, primarily focusing on the emergence and evolution of chess engines. Drawing from historical precedents set by luminaries like Alan Turing and Claude Shannon to the modern-day marvels of artificial intelligence, the study provides a comprehensive look into the technical and algorithmic underpinnings of these engines. Furthermore, through expert opinions and a meticulously conducted survey, the research elucidates the profound impact of chess engines on the game, teaching methodologies, and player strategies. This work serves as a confluence of historical narratives, technical expositions, and sociological insights, offering a panoramic view of how technology has reshaped the realm of chess.

1) Introduction

Chess, originating back to the 6th century, has witnessed a profound transformation over the last few decades. This change is primarily driven by the rise of digital technology and the advent of chess engines. These engines, essentially computer programs designed to analyze, evaluate, and play chess, have revolutionized the way the game is approached, learned, and mastered.

To understand the significance of chess engines, one needs to consider the broader context. Chess has always been more than just a game. It's a reflection of strategic thinking, foresight, and the ability to outwit an opponent. For centuries, mastering chess required years of practice, studying under mentors, and playing countless games. However, the digital age has added a new dimension to this learning curve.

In this thesis, the focus is squarely on these powerful chess engines and their many-sided impact. How are they designed? What algorithms drive their decision-making processes? How have they evolved from their early incarnations to the sophisticated versions we see today? These are some of the technical aspects we'll delve into.

But the influence of chess engines is not confined to the digital realm. They've had tangible effects on the human side of the game. New players now often start their chess journey with a computer program as their primary sparring partner. Even seasoned players and grandmasters use these engines for practice, analysis, and to refine their strategies. The engines provide insights into moves and strategies that even the most experienced human players might overlook.

Furthermore, we'll explore the broader cultural and social implications of this shift. How has the rise of chess engines affected the way chess is played? Has it democratized the learning process, making high-quality chess education accessible to more people? Or has it created a dependency, where the human element of intuition is overshadowed by machine-calculated precision?

Drawing from expert opinions, firsthand accounts, and survey data, this thesis aims to paint a comprehensive picture of the current chess landscape. We'll look into perspectives of players, from amateurs to professionals, sharing their experiences on the fusion of traditional chess with modern technology.

In essence, this exploration is a journey through the intersections of technology, tradition, and talent. Throughout this narrative, an invitation will be extended to ponder the evolving dynamics of chess in the digital age, reflecting on the past, understanding the present, and speculating on the future of this age-old game in a rapidly changing world.

2) Methodology

1. Research Objective:

The primary aim of this study is to explore the development, functioning, and impact of chess engines in the realm of chess. This encompasses a technical examination of the engines, a historical overview of their progress, and a sociological analysis of their influence on the game and its players.

The Technical Chapter serves as the foundation, introducing and explaining the core terminologies and functionalities of chess engines. This technical grounding is crucial for comprehending the subsequent historical chapter, ensuring that readers are well-equipped with the necessary terminology.

The History Chapter delves into the progression of chess engines over time, tracing their origins, milestones, and advancements. This historical perspective not only offers insights into their technical evolution but it also sets the stage for understanding their broader societal impact.

The final chapter on the Impact of Chess Engines on Chess is enriched by the prior chapters. With a grasp on the technical nuances and historical trajectory of chess engines, readers can better appreciate their profound influence on the game, its players, and the overarching chess community.

2. Data Collection:

Literature Review: A comprehensive review of existing literature was conducted to understand the historical and technical aspects of chess engines. This included academic papers, articles, books, documentaries, videos and interviews. Key figures, milestones, and advancements were identified and compiled to form a cohesive narrative.

Survey: An online survey was distributed to a diverse group of chess players of all ages, ranging from novices to experts. The survey aimed to gauge players' experiences, preferences, and opinions regarding the use of chess engines in their practice and gameplay.

3. Data Analysis:

Software and Libraries: The analysis was primarily conducted using Python in a Jupyter Notebook environment. The following Python libraries were employed for data processing, analysis, and visualization:

Data Processing and Analysis: pandas, numpy, and statsmodels.api.

Visualization: matplotlib.pyplot and seaborn.

Statistical Testing: stats from scipy and statsmodels.formula.api.

Quantitative Analysis:

Survey responses were collated and subjected to statistical analysis using software tools. Patterns, correlations, and significant findings were extracted and visualized using graphs and tables.

Qualitative Analysis:

Insights from expert interviews were transcribed, coded, and analyzed thematically. Common themes, opinions, and perspectives were identified and integrated into the broader narrative of the study.

4. Limitations:

The rapid advancement of technology means that some technical aspects might evolve post this research.

5. Editing and Refinement:
For the improvement of sentence structure and correction of syntax errors within the thesis, the language model ChatGPT 3.5 was utilized. This tool assisted in refining the language for clarity and coherence, ensuring the content is presented in the most understandable and polished manner.

6. Ethical Considerations:
All survey participants were informed of the research's purpose, and their consent was obtained.

3) Technical and Algorithmic Aspects of Chess Engines

The development of chess engines dates back to the early days of computing. Early pioneers like Alan Turing and Claude Shannon laid the groundwork for artificial intelligence and its application to chess. Through the years, the development of chess engines has been marked by various milestones, such as the legendary match between IBM's Deep Blue and Garry Kasparov in 1997. This historic event signaled a new era in the world of chess, with machines proving that they were capable of defeating even the most formidable human opponents. The achievement of this outcome was the culmination of a prolonged and intricate process in advancing key components of the chess engines such as the board representation, the move generation and the evaluation function.

A. Board representation

There are three main representation techniques.

1. Bitboards

Bitboards, also known as "bitmaps," are a popular and efficient board representation technique used in modern chess engines. A bitboard is a 64-bit binary number, with each bit corresponding to a square on the chessboard. A set bit (1) represents the presence of a particular piece on a square, while an unset bit (0) indicates the absence of the piece.
In a typical chess engine, there are 12 distinct bitboards for each type of piece for both players (e.g., white pawns, black pawns, white knights, black knights, etc.). By performing bitwise operations (AND, OR, NOT, XOR, and bit shifts), an engine can quickly generate legal moves, evaluate board positions, and detect captures and other special moves (castling, en passant, and promotions) (Frayn, 2005).
Advantages of bitboards include:

- Speed: Bitwise operations are extremely fast, allowing for quicker move generation and position evaluation.
- Compactness: Bitboards use minimal memory, which is an essential factor for efficient chess engines.
- Parallelism: Modern processors can perform bitwise operations on multiple bitboards simultaneously, further improving the engine's speed.

2. Array-based representations

Before bitboards, array-based representations were the standard approach for chess engines. In this technique, the chessboard is represented by an 8x8 or a 10x12 array, with each element storing a piece code or a special value for empty squares or squares outside the board. (Hyatt, 2004)
Two commonly used array-based representations are:

- 8x8 array: A simple, straightforward representation that closely mimics the actual chessboard. Each element contains the piece code or a value for an empty square.
- 10x12 array (mailbox): This representation extends the 8x8 array by adding extra rows and columns around the board to simplify move generation and boundary checking. The extra rows and columns are filled with "off-board" values to indicate illegal moves.

Advantages of array-based representations include:

- Clarity: Array-based representations are more intuitive and easier to understand than bitboards.
- Move generation: Generating moves is relatively simple, as the engine can directly access the piece codes in the array.
- Compatibility: Engines using array-based representations can easily work with external tools, as the board representation is more human-readable.

3. Hybrid approaches

Some engines combine bitboard and array-based representations to leverage the advantages of both techniques. An engine might use bitboards for move generation and attack detection while maintaining an array-based representation for position evaluation and compatibility with external tools. Tom's Simple Chess Program (TSCP) is a representative example of the hybrid approach since it uses bitboards to achieve computational efficiency, while the array-based representation ensures compatibility with other tools and provides a more accessible and understandable system for new developers. [https://www.chessprogramming.org/TSCP] [http://www.tckerrigan.com/Chess/TSCP/]

B. Move Generation

The ability to generate legal moves quickly and efficiently is a crucial aspect of any chess engine. Move generation is the first step in the engine's search process, which is responsible for selecting the best move to play.

1. Pseudolegal move generation

The most straightforward approach to move generation is generating all possible moves for a given position without considering whether they expose the king to check. These moves are called "pseudolegal" moves. Pseudolegal move generation involves iterating through all pieces on the board and determining their possible moves based on their type, location, and the current position. (Ellis Jones, 2017),(Pettersson, 2006)

Advantages of pseudolegal move generation:

- Simplicity: The process is straightforward and easy to understand, making it suitable for beginners or developers who want to focus on other aspects of the engine.
- Speed: Pseudolegal move generation can be fast, especially when combined with efficient board representation techniques like bitboards.

Disadvantages of pseudolegal move generation:

- Inaccuracy: The generated move list may include illegal moves that expose the king to check. These moves must be removed later in the search process, adding extra overhead.

2. Legal move generation

Legal move generation involves generating only moves that do not expose the king to check, ensuring that the resulting move list is accurate and free of illegal moves. This approach requires more complex algorithms and often involves checking for pins, discovered attacks, and other special cases.

Advantages of legal move generation:

- Accuracy: The generated move list only contains legal moves, eliminating the need for later filtering.
- Efficiency: Legal move generation can lead to a more efficient search process, as illegal moves are not considered during the search.

Disadvantages of legal move generation:

- Complexity: The algorithms required for legal move generation are more complex and challenging to implement.
- Speed: Legal move generation can be slower than pseudolegal move generation, as it requires additional checks and calculations.

3. Move generation optimizations

To enhance move generation efficiency, chess engines employ various optimizations, including:

- Move ordering: By generating and examining moves in a specific order, engines can improve search efficiency. For example, captures and promotions are typically more promising moves and should be examined first.
- Piece list: Instead of iterating through all squares on the board, engines can maintain a list of active pieces, speeding up the move generation process.
- Precomputed data: Engines can use precomputed tables or databases for specific piece movements, particularly for sliding pieces (rooks, bishops, and queens), to reduce computation time.
- Magic bitboards: A technique used in conjunction with bitboard representation, magic bitboards enable fast and memory-efficient move generation for sliding pieces using a combination of multiplication, bitwise operations, and lookup tables.

C. Evaluation Function

A chess engine's performance depends significantly on its ability to evaluate positions accurately and efficiently. The evaluation function serves as a heuristic to estimate the advantage of one side over the other, allowing the engine to differentiate between favorable and unfavorable positions. In turn, it informs the search algorithm, which selects the best move to play. This paper will provide an in-depth analysis of various factors and techniques that contribute to an effective evaluation function for chess engines (Frayn, 2005).

1. Material balance:

Material balance is a fundamental aspect of position evaluation. Chess engines assign point values to each piece type, reflecting their relative strength and importance. These values are typically derived from centuries of human chess knowledge and are generally agreed upon as follows: pawn (1), knight (3), bishop (3), rook (5), and queen (9). The evaluation function calculates the material balance by summing the point values for each side and subtracting one side's total from the other.

2. Piece-square tables:

Piece-square tables are a widely used technique to enhance the evaluation function by considering piece placement. Each piece type is associated with a table containing position-specific values that reflect the relative desirability of placing that piece on a particular square. The engine adds these values to the material balance, favoring positions with better piece placement. Piece-square tables can be fine-tuned based on opening, middlegame, and endgame knowledge to optimize piece coordination and activity throughout the game.

3. Mobility:

Mobility, or the number of legal moves available to a player, is another crucial factor in position evaluation. Greater mobility often translates to more opportunities for creating threats or responding to the opponent's moves. The evaluation function considers the mobility of both sides and assigns a higher score to positions with greater mobility.

4. King safety:

The safety of the king is a paramount concern in chess. A well-protected king reduces the likelihood of checkmate, while an exposed king may become vulnerable to threats and tactics. The evaluation function takes into account factors such as the presence of enemy pieces near the king, pawn shielding, and the king's escape routes. By penalizing positions with unsafe kings, the engine promotes moves that prioritize king safety.

5. Pawn structure:

Pawn structure plays a vital role in determining the overall strength of a position. The evaluation function considers aspects of pawn structure such as doubled pawns, isolated pawns, and passed pawns. Healthy pawn structures are rewarded with higher scores, while weak structures are penalized. This encourages the engine to make moves that maintain or improve its pawn structure.

6. Machine learning in evaluation functions:

Recent advancements in machine learning, particularly in neural networks, have facilitated the development of more robust evaluation functions. Engines like Leela Chess Zero (LCZero) use deep neural networks to evaluate positions, learning from a vast dataset of high-quality games. These machine learning-based evaluation functions can capture subtle positional nuances that are difficult to quantify with traditional hand-crafted evaluation features, resulting in stronger and more human-like play. (Frayn, 2005)

D. Algorithmic approaches

Minimax

The minimax algorithm is a decision-making process in game theory and artificial intelligence that seeks to minimize the worst-case scenario outcome. In the context of a chess engine, it is used to find the best move by assuming that the opponent will always respond with the best countermove. Here's a brief summary of how the minimax algorithm works in a chess engine:

1) **Generating the game tree:** For a given state of the chess game, the algorithm generates a game tree where the nodes represent game states and edges represent moves. The game tree extends down to a certain depth, which depends on the computational power available. Each layer of the tree represents one move, with the root being the current state of the game.
2) **Evaluation:** At the bottom of the tree (the leaf nodes), the algorithm evaluates the state of the game from the perspective of the AI. This evaluation as mentioned in the previous section could be based on many factors, such as the material balance, control of the center, king safety, etc.
3) **Backpropagation:** The algorithm then backpropagates these evaluation scores up the tree using the minimax principle: At each level of the tree, the player whose turn it is seeks to either minimize or maximize the evaluation score. If it's the AI's turn, it aims to maximize the score (choose the move that leads to the best outcome for the AI), and if it's the opponent's turn, it aims to minimize the score (choose the move that leads to the worst outcome for the AI).
4) **Choosing the best move:** At the top of the tree, the AI chooses the move that leads to the maximum evaluation score.

It's worth noting that the basic minimax algorithm is not practical for a game like chess due to the huge number of possible game states (estimated to be more than 10^120). So, in practice, chess engines use an optimized version of the algorithm called alpha-beta pruning, which cuts off branches in the game tree that it has determined will not affect the final decision, thus significantly reducing the computational cost. (GeeksforGeeks, 2022)

Alpha-beta pruning

Alpha-beta pruning is an optimization technique that significantly reduces the number of nodes that need to be evaluated in the game tree. The main idea behind it is that evaluating branches (or subtrees) can be skipped in the game tree if a better move is already found elsewhere. By doing this, the tree can be “pruned” and, thus, save computational resources.

How it works:

1) Two values, alpha and beta: In alpha-beta pruning, we maintain two values while traversing the tree, named alpha and beta. Alpha represents the minimum score that the maximizing player is assured of, while beta represents the maximum score that the minimizing player is assured of. Initially, alpha is negative infinity and beta is positive infinity, meaning that the players' scores can range anywhere.

2) Updating alpha and beta: As the nodes in the game tree are evaluated, the values of alpha and beta are updated. If a node is reached where the AI is supposed to play (a maximizing node), and a move with a higher score than the current alpha is found, alpha is updated to that score. Similarly, if a node is reached where the opponent is supposed to play (a minimizing node), and a move with a lower score than the current beta is found, beta is updated to that score.
3) Pruning the tree: The key insight of alpha-beta pruning is that, if alpha is greater than or equal to beta at any point, the evaluation of the remaining siblings of the current node can be stopped (i.e., the tree can be pruned). The reason is that a move has been found that guarantees the maximizing player a score of at least alpha (which is better than beta), or a move that guarantees the minimizing player a score of at most beta (which is worse than alpha). Since the other player will not choose this branch, there is no need to evaluate the remaining nodes.

The final result of the minimax algorithm is not affected by alpha-beta pruning; it only improves the efficiency of the algorithm by pruning unnecessary branches. However, it is important to note that the effectiveness of alpha-beta pruning relies on the order in which nodes are evaluated. When the best moves are considered first, alpha-beta pruning becomes most effective, leading to a much faster execution of the algorithm. Therefore, employing good move ordering heuristics is crucial to maximize the performance of a chess engine utilizing alpha-beta pruning (Frayn, 2005).

Move ordering heuristics

Move ordering heuristics are used to improve the efficiency of the minimax algorithm with alpha-beta pruning. The idea is to consider the most promising moves first, so that the algorithm can prune as many nodes as possible. Good move ordering allows alpha-beta pruning to reach its maximum efficiency, thus allowing the engine to search deeper in the same amount of time.

In chess engines, several heuristics can be used to order moves:

1. Capture moves: Moves that capture an opponent's piece are often high-impact moves that can drastically alter the game's state. Therefore, they're usually examined early in the move ordering, especially if they capture a higher-valued piece without losing a piece in return (a "winning" capture).

   The 'Most Valuable Victim/Least Valuable Aggressor' (MVV/LVA) is a commonly used heuristic for ordering capture moves. The idea is to prioritize captures that could potentially result in a large net gain in material. For instance, it's more beneficial to lose a pawn (least valuable) in order to take a queen (most valuable) than the other way around. This heuristic works because such trades often force the opponent to recapture, leading to a significant pruning of the search tree.

   However, MVV/LVA does not consider the safety of the attacking piece after the capture. Hence, extensions like SEE (Static Exchange Evaluation) are used. SEE looks one step ahead and considers whether the attacking piece can be recaptured. It continues this process until no more captures are possible or beneficial. The sequence of captures and recaptures is evaluated statically, hence the name. (Boulé, 2002)

2. Killer heuristic: If a certain move caused a beta cutoff (forced the opponent into a worse position) in another branch at the same depth, this move is tried early on in other branches at the same depth (Akl and Newborn, 1977).
3. History heuristic: Moves that have caused cutoffs in the past are more likely to cause cutoffs in the future, so they're considered earlier (Schaeffer, 1989).
4. Transposition tables: These are a kind of database that stores previously evaluated positions of the game. If the same position is reached by a different sequence of moves, the engine can simply look up the evaluation in the transposition table rather than recalculating it (Frayn, 2005).
5. Principal variation: The sequence of moves that was considered best during the last iteration of the search (assuming iterative deepening is used) often provides a good move ordering for the next iteration (Frayn, 2005).

Iterative Deepening

This is a strategy where the AI starts by only considering moves one step ahead, then two steps ahead, then three, and so on, incrementing the depth of the search until it runs out of time. This strategy is useful because it provides a rough evaluation quickly, which can be refined as time allows.

The advantage of iterative deepening is that it's an anytime algorithm, meaning it can return a valid result given any amount of time. This makes it suitable for situations where the AI has a strict time limit, such as in a timed chess game.

It might seem that iterative deepening would waste a lot of time because it re-computes the same nodes multiple times. However, because of the exponential nature of the game tree, the nodes on the "surface" (i.e., at the current maximum depth) vastly outnumber the nodes "inside" (at lesser depths), so the extra computation is relatively small.

Iterative deepening can be combined with alpha-beta pruning and good move ordering to make it even more efficient. The principal variation (the best move sequence) found in each iteration can be used as a guide for move ordering in the next iteration, which can lead to more effective pruning (Frayn, 2005).

Quiescence Search

It is a technique used in chess engines to address the horizon effect, a limitation where a computer might prematurely stop analyzing a position because it reaches its maximum search depth, potentially missing a crucial sequence just beyond.

The Quiescence Search is invoked when the regular search reaches its maximum depth. Instead of stopping entirely, the search continues only for 'quiet' or non-volatile positions, which are typically captures and checks in chess. The idea is to avoid stopping the search in the middle of a battle where pieces are being exchanged, as this could lead to greatly misleading evaluations.

For instance, if the search stopped right before an opponent's piece could be captured, the evaluation might erroneously consider the player's position favorable. By extending the search to include these 'quiet' moves, the Quiescence Search helps ensure that the evaluation function gets invoked at a relatively stable position, leading to a more accurate evaluation of the game state (Frayn, 2005).

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm widely used in game playing AI, including in some chess engines. It is especially famous for its use in the game of Go, where it was a key component of Google's AlphaGo, the first AI to beat a world champion Go player.

MCTS differs from the minimax-based approach in that it does not require an explicit evaluation function to assess the game positions. Instead, it uses statistical analysis of simulated games to inform the decision-making process.

Here's a brief overview of how MCTS works:

1. Selection: Starting from the root node (the current game state), the algorithm selects optimal child nodes until it reaches a leaf node. The selection of nodes is based on a balance between exploring nodes that have not been visited much and exploiting nodes that have high average rewards. This balance is usually achieved through a formula called the Upper Confidence Bound applied to Trees (UCT).
2. Expansion: If the leaf node is a non-terminal state (i.e., the game is not over at this state), one or more child nodes are added to the tree.
3. Simulation: The algorithm simulates a game from the new node state to a terminal state (an end of the game), using a policy of random or semi-random moves.
4. Backpropagation: Based on the result of the simulation (win, loss, or draw), the algorithm goes back up the tree to the root, updating the visit count and the total reward of each node along the way. Nodes that lead to wins get higher rewards.

This process (selection, expansion, simulation, and backpropagation) is repeated many times. When it's time to make a move, the algorithm chooses the move that leads to the child node with the most visits (Wang, 2021).

While MCTS can be used on its own, it can also be combined with other techniques, such as neural networks. A relatively new type of modern chess engines do not use traditional chess programming techniques like handcrafted evaluation functions or explicit alpha-beta pruning. Let’s take for example LCZero which instead uses a combination of MCTS and two types of neural networks:

1. The Policy Network: This network is used to guide the selection of moves during the MCTS. Instead of exploring all possible moves equally, LCZero uses the policy network to favor moves that are more likely to lead to good outcomes. This allows it to focus the search on the most promising moves, which makes the MCTS more efficient.
2. The Value Network: This network is used to evaluate game positions during the MCTS. Instead of simulating games all the way to the end, LCZero can use the value network to estimate the outcome of the game from the current position. This allows it to simulate games faster and more efficiently (Silver et al, 2016).

Neural networks

Neural networks have become an integral part of modern chess engines, revolutionizing the way artificial intelligence understands and plays chess. A neural network is a type of machine learning model inspired by the human brain's structure. It consists of

interconnected layers of nodes, or "neurons," which can learn to recognize patterns and make decisions.

Basics of Neural Networks

Neural networks are made up of several layers, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to all neurons in the next layer, with each connection having a weight that determines the influence of one neuron on another.

The model learns by adjusting these weights based on the error of its predictions, a process known as backpropagation. The goal is to minimize the difference between the network's output and the expected outcome. The learning process can involve millions, or even billions, of these adjustments.

Neural Networks in Chess Engines

Neural networks have proven highly effective for chess engines. They can learn to recognize complex patterns on the chessboard, evaluate positions, and even suggest moves. This is a significant departure from traditional chess engines, which rely on handcrafted evaluation functions and brute-force search techniques (Silver et al., 2017).

AlphaZero and Monte Carlo Tree Search

The real game-changer was AlphaZero. AlphaZero, developed by DeepMind, uses a type of neural network known as a deep convolutional neural network along with Monte Carlo Tree Search (MCTS). The network is trained through self-play, starting from random play and improving over time.

AlphaZero's neural network serves two main purposes. First, it evaluates the position on the board, outputting a score that estimates the probability of winning. Second, it provides a policy vector, suggesting the likely best moves in the current position.

This information is then used by MCTS to decide which positions to explore. Unlike traditional engines that try to analyze all possible moves, MCTS uses the neural network's suggestions to focus on the most promising ones, significantly reducing the search space.

AlphaZero's approach differs significantly from earlier engines like Stockfish, which use handcrafted evaluation functions and alpha-beta search algorithms. Despite evaluating far fewer positions per second, AlphaZero outperformed Stockfish in matches, demonstrating the effectiveness of its neural-network-based approach.

Neural Network Architectures in Chess

The architecture of the neural network plays a crucial role in its performance. AlphaZero uses a deep convolutional neural network, designed to process grid-like data, which is ideal for a chessboard.

Another popular architecture in chess is the transformer network, used by DeepMind's MuZero, an extension of AlphaZero. Transformers use self-attention mechanisms that allow them to consider the entire input sequence simultaneously, making them effective for understanding the relative importance of different pieces and their positions on the board.

Training Neural Networks for Chess

Training a neural network for chess involves reinforcement learning, a type of machine learning where an agent learns to make decisions by interacting with its environment. In the case of AlphaZero, the environment is the game of chess. AlphaZero begins by playing games against itself, initially making moves randomly. After each game, the neural network is adjusted based on the outcome. Over time, the engine becomes better at evaluating positions and suggesting moves.

This self-play reinforcement learning allows AlphaZero to learn from a much broader range of positions than would be possible from human games alone. It also enables the engine to discover new strategies and ideas, leading to a highly creative and unconventional style of play.

Challenges and Limitations

While neural networks have brought significant advancements, they also come with challenges. Training a large, deep, neural network requires considerable computational resources and time. AlphaZero's training, for instance, was performed on Google's high-powered hardware over several hours. This kind of resource availability might not be feasible for most individual developers or researchers.

Moreover, a trained neural network is often described as a "black box," as it's challenging to understand why it makes a particular decision. While this opacity is not a problem for gameplay, it may hinder the learning experience for human players who want to understand the reasoning behind a recommended move.

Another limitation is overfitting, where the network becomes so well-tuned to the training data that it performs poorly on unseen data. In the context of chess, overfitting could mean that the engine plays incredibly well in positions it has encountered during training but struggles in unfamiliar situations. (Silver et al, 2017)

4) History of Chess Engines

After the thorough examination of the intricacies of the algorithmic approaches employed by modern chess engines, a comprehensive understanding of their inner workings has been established. With this solid foundation and knowledge of terminology in place, it is now possible to delve into the journey of chess engine evolution and trace the path that led to their remarkable capabilities today. Chess engine development did not occur in isolation; rather, it has been an iterative process fueled by ingenious minds and relentless pursuit of improvement. Understanding the historical context and significant milestones achieved along the way is crucial for appreciating the remarkable advancements witnessed in recent years.

Before the Beginning

The Mechanical Turk and the Prelude to Chess Automation

The concept of a machine playing chess predates the advent of computers and artificial intelligence, and it can be traced back to the creation of "The Mechanical Turk", a famous chess-playing machine built in the late 18th century.

Created by Wolfgang von Kempelen in 1770, The Turk was a life-sized automaton figure resembling an Ottoman chess master seated at a cabinet. It was dressed in traditional Turkish clothing and had a chessboard placed in front of it. The machine gained significant attention and toured Europe and the United States, competing against famous opponents and astounding audiences with its chess-playing abilities.

The Turk appeared to possess remarkable intelligence and skill, defeating many human challengers, including prominent figures such as Napoleon Bonaparte and Benjamin Franklin. However, what made The Turk truly fascinating was the fact that it was not an autonomous machine capable of playing chess on its own. Instead, it relied on the hidden presence of a human chess master who operated the machine from inside.

The illusion of The Turk's autonomy was achieved through clever mechanical and engineering techniques, utilizing a series of hidden compartments, levers, and magnets to manipulate the chess pieces on the board. The operator, concealed within the cabinet, observed the game through a system of mirrors and employed their chess expertise to make the moves (Walter, 2019).

While The Turk itself was not a true machine capable of playing chess independently, it laid the foundation for the idea of automating the game. Its existence sparked discussions and debates about the possibility of creating a machine that could rival human chess players. The Turk captured the imagination of the public and demonstrated the fascination with the concept of a mechanical device emulating human intellect and skill in a game as complex as chess (Schaffer, 1999)].

The legacy of The Turk lives on in the development of chess-playing machines and eventually the field of artificial intelligence. The quest to create a true autonomous chess-playing machine became a significant challenge for inventors and scientists in the years that followed. It was not until the advancement of computers and the emergence of artificial intelligence that the dream of a machine playing chess without human intervention became a reality.

Therefore, The Turk stands as a testament to the early attempts at creating a chess-playing machine and serves as a historical precursor to the ongoing advancements in the field of artificial intelligence and machine learning, which eventually led to the development of powerful chess-playing programs and chess-playing AIs (Geoghegan, 2020).

El Ajedrecista: The first real chess automation

El Ajedrecista, Spanish for "The Chess Player", holds a crucial position in the timeline of artificial intelligence (AI) in gaming, nestled between the era of The Mechanical Turk and the advent of Alan Turing's chess algorithm. While The Mechanical Turk was a masterful illusion, El Ajedrecista bridged these eras, and it was the first real instance of an automatic machine playing a game of chess (Montfort, 2003).

Invented by Leonardo Torres y Quevedo, a Spanish civil engineer and mathematician, El Ajedrecista was revealed to the public at the Paris World Fair in 1914. Although the machine only played a specific endgame scenario - a rook and king against a lone king - it was a marvel of technology for its time. The machine would respond to a human's move by physically moving its own pieces, giving it an aura of autonomy that was virtually unheard of at the time. El Ajedrecista's impact lies not in its complexity of play, but in its ability to mechanize decision-making. Torres y Quevedo implemented a series of electromechanical circuits and actuators, effectively creating a rudimentary form of AI. This marked the first instance of a machine interacting with its environment and making decisions ("Automates la nature torres," n.d.) based on its current state. Unlike The Mechanical Turk, which concealed a human operator, El Ajedrecista was completely autonomous and deterministic. Every move it made was calculated through its embedded circuits, with no human intervention involved during gameplay (Velasco, 2011).

This marked a paradigm shift in the conceptualization of machines, from mere tools to autonomous entities capable of decision-making. This contraption represented the first steps towards a world where machines would not just perform manual labor but also engage in intellectual pursuits.

Early Beginnings

Turochamp

Alan Turing's and David Champernowne's chess program, affectionately dubbed "Turochamp", is a fascinating artifact of computer history, representing one of the earliest attempts at programming a computer to play a strategic board game (Giannini & Bowen, 2017). The ingenuity of Turing's approach is reflected not only in his vision of what machines could accomplish but also in his practical solutions to the limitations of his era.

Turochamp was conceptualized in the late 1940s to early 1950s, a time when computer technology was in its infancy. No computer existed then that was capable of running such a program. However, this did not dissuade Turing. Instead, he proposed the program as a theoretical exercise and even carried out a game of chess by manually computing each move according to the program's rules, an arduous process that took several hours (Kasparov & Friedel, 2017).

Turochamp used a purely numerical and rather straightforward system to evaluate the chess board. It did not look ahead to consider chains of moves and counter-moves as modern chess engines do, largely because the computational power to do so was unimaginable at the time. Instead, it evaluated the board based on the current position alone. The program would explore all possible moves and assign each a score based on a series of mathematical considerations. The move with the highest score was the one the program would make. Scoring in Turochamp was based on a unit that Turing called a "point". Different pieces were assigned different values, much like in modern chess: a pawn was worth 1 point, a knight or bishop 3 points, a rook 5 points, and a queen 10 points. The king was not given a point value because its capture ends the game (Copeland et al., 2017).

In addition to these basic piece values, Turing included a few more advanced concepts in his scoring system. For example, he recognized the importance of pawn structure and added a bonus of 0.5 points for a pawn that was part of a contiguous line of friendly pawns, signifying a strong pawn formation. He also included a concept of "mobility", or the ability to control a large number of squares, and gave a bonus of 0.1 points for each legal move that a piece could make.

However, Turochamp was not just about capturing and counting pieces; it also considered strategic elements of the game. For instance, it understood the concept of a 'fork', a tactic where a piece threatens two or more opponent pieces at the same time. The program also knew to secure its own king and to make offensive plays for the opponent's king. It was also designed to be adaptable. If the program was losing (that is, if it had fewer points than the opponent), it would play more aggressively, assigning a higher value to moves that threatened the opponent's pieces. If it was winning, it would play more defensively, prioritizing the safety of its own pieces (Turing & Copeland, 2004).

Despite these advanced features, Turochamp had its limitations. It lacked the ability to plan ahead, which is a critical part of human chess strategy. Turing's program could be easily duped by a strategy that involves a sacrifice leading to a longer-term advantage, since it would simply see the sacrifice as a gain in points. This limitation reflects the computational limitations of the time, as looking ahead to future moves would require an exponential increase in computational power.

Turing tested the program against human opponents and it is said that it won against Champernowne's wife, who was a beginner at chess, but the game is not recorded. Although, a later game, played in 1952, versus a colleague of Turing's, Alick Glennie, is saved (chessgames.com, n.d.). Turochamp was performing relatively well until the mid-game. At this point, at move 28, it had a slight advantage. However, move 29 was a blunder that led to Turochamp losing its queen and being in a completely lost position, so the game stopped there.

During the commemoration of Turing's centennial in June 2012, held in Manchester, Garry Kasparov and Frederic Friedel presented a lecture on the reconstruction of Turing's programmed chess engine. Friedel, co-founder of Chessbase, emphasized that Turing, using only paper and pencil, relied on his intuition to generate the moves and unbeknownst to him, Turing employed Alpha-Beta pruning, a technique that was developed several years later when machines became capable of executing code. Kasparov, the former world chess champion, reviewed Turochamp's game against Alick Glennie and remarked, "Blundering a queen in one move, that is not a great accomplishment. But still the game was played and there were 29 moves, not just legitimate moves but you may call them decent moves" (Friedel, 2017). Following the lecture, the reconstructed Turochamp faced Kasparov and unsurprisingly lost in just 16 moves. Nevertheless, after his victory, the former world champion acknowledged that, considering its era, Turochamp served as a significant precursor to today's chess engines and could provide an adequate challenge for an amateur player.

Claude Shannon's Approach

Renowned as a significant figure in mathematics and electrical engineering, Claude Shannon and his groundbreaking work during the 1940s and 1950s drastically altered the landscape of computer chess and contributed significantly to the nascent field of artificial intelligence. Shannon's approach to digital chess began with his landmark paper in 1949, "Programming a Computer for Playing Chess" (standford.edu, 2013).

This work served as a blueprint for the early development of computer chess. He proposed two distinct strategies for chess-playing algorithms: a full-width strategy, termed 'Type A', and a selective strategy, termed 'Type B' (Shannon, 1949). The former involved exhaustive calculations of all possible moves up to a certain depth, while the latter emphasized discerning and pursuing only the most promising moves. These strategies laid the foundation for more sophisticated techniques in future computer chess algorithms, marking a significant leap forward in computerized decision-making.

A notable aspect of Shannon's contribution was his introduction of an evaluation function to digital chess. Unveiled in the late 1940s, this innovative concept was designed to assign numerical values to different chess positions. The function considered multiple factors contributing to a player's position, including material balance, control of the center, pawn structure, piece mobility, and king safety. This advancement presented a novel way of digitizing the traditionally human exercise of strategic assessment in chess, paving the way for computers to engage in deeper, more nuanced gameplay (Shannon, 1950).

Moving from theory to practice, Shannon actualized his concepts with the construction of a chess-playing machine in the early 1950s. This machine, while limited in its

ability to play a full game of chess, demonstrated the practical implementation of Shannon's theories (Chessprogramming.org, n.d). The device could try to calculate and execute the optimal move based on the current board configuration, integrating digital computation and algorithmic strategy in a way that had rarely been seen before.

Claude Shannon envisioned a future in which artificial intelligence could be applied to a wide range of tasks, such as routing phone calls, translating text, or composing melodies (Shannon, 1950). Despite the diversity of these applications, they shared an important characteristic: they did not operate according to a "strict, unalterable computing process." Instead, the solutions to these problems had a continuous range of quality, allowing for a more nuanced and flexible approach to decision-making (Soni, 2017).

Given this context, chess emerged as a valuable test case for the emerging generation of artificial intelligence. Chess was a game that people believed required "thinking," making it an ideal candidate for testing whether computers could be made to think. Additionally, chess had well-defined rules, yet it was a complex game with many possible moves, providing a challenging problem for computer programming.

Dietrich Prinz: Machine solves mate in two.

Dietrich Prinz, a German-born computer scientist, made significant contributions to the field of computer chess programming, standing alongside the likes of Alan Turing and Claude Shannon. His work during the early 1950s is widely acknowledged as one of the first instances of creating a program to play a full game of chess, albeit in a rudimentary form. Prinz was a protégé of Alan Turing’s, and he joined Ferranti in Manchester, England, in 1951. There, he worked on programming the Ferranti Mark I, the world's first commercially available general-purpose electronic computer. During this period, he developed a chess program that could effectively navigate the game's complexities, albeit limited to solving mate-in-two problems.

Prinz's program, like those of his contemporaries, used a form of brute-force search to evaluate possible moves. However, due to the limited computational power of the Ferranti Mark I, his program could analyze a small number of positions per second. As a result, the algorithm was tailored to solve 'mate in two' problems, rather than full games of chess.
Prinz's algorithm involved generating all legal moves for the given position and then evaluating the resulting positions. If any of these positions was a checkmate for the opponent, the corresponding move was the solution. If none of the positions resulted in a checkmate, the algorithm would generate all legal responses for each move, checking again for a checkmate. This cycle would repeat until a mate-in-two solution was found or all possibilities were exhausted (Prinz, 1988).

Prinz's work marked a significant milestone in computer chess history, primarily due to his ability to implement the concepts put forth by Turing and Shannon into a practical application. His program was the first to demonstrate a computer making non-trivial decisions about chess gameplay based on a fixed set of rules and evaluation methods, a substantial leap from El Ajedrecista and a meaningful stepping stone towards modern computer chess engines.

Early Attempts at Computer Chess: MANIAC I and Beyond

In the mid-20th century, the Mathematical Analyzer, Numerical Integrator, and Computer, better known as MANIAC I, represented a milestone in the world of computers. Developed at the Los Alamos Scientific Laboratory in the early 1950s, it was among the earliest electronic digital computers. Beyond its primary mission—conducting calculations related to the hydrogen bomb—MANIAC I found a unique secondary application in the realm of chess. In 1956, under the direction of the researchers, Stanislaw Ulam, Paul Stein, Mark Wells, James Kister, William Walden and John Pasta, there was implemented a program which stood as one of the earliest chess applications to run on a digital computer. However, due to the memory and processing constraints of MANIAC I, the program was forced to navigate a modified chess landscape which later would be named Los Alamos Chess. Specifically, it operated on a 6x6 chessboard with a pared-down number of pawns and no bishops (Anderson, 1986). There was no pawn double-step move, nor the en passant capture. Also, castling was not allowed.

This abbreviated version of chess represented a practical compromise given the technological constraints of the period. Despite these limitations, it was not simplistic for its era. The chess program designed for MANIAC I relied on a brute-force Shannon Type A strategy algorithm, which generated all possible moves, followed by all potential responses. The resulting positions were then assessed, with the program selecting what it deemed the most promising move. With 11,000 ops./sec, each move took about 12 minutes to be selected (Isenberg, 2020).

The computer participated in a series of three games, each with distinct characteristics and outcomes. The first game involved the computer competing against itself, testing its computational capabilities. In the second game, the computer faced a skilled human player who played without a queen, introducing a handicap. Despite the disadvantage, the human player emerged victorious. The third and final game witnessed the computer challenging a laboratory assistant who had received specific instruction in chess rules during the preceding week, solely for this match. In a significant breakthrough, the computer emerged triumphant, marking the first recorded instance of a computer defeating a human player in a chess-like game (Lewis, 2022).

A few hundred miles to the east, in New York City, another landmark development in computer chess was taking shape. At IBM, Alex Bernstein was working on a project that would soon achieve a significant leap in computer chess programming. In 1957, Bernstein completed his work on the first comprehensive chess program. Running on an IBM 704, a computer far more powerful than the Ferranti Mark I or the MANIAC I, it was the first program capable of playing a complete game of chess on a standard 8x8 board, including all the traditional rules and piece types and taking into account the special moves castle and en passant. While Bernstein's program, like its predecessors, utilized a brute-force search algorithm and an evaluation function, it employed Shannon's Type B strategy (Lasar, 2011) and integrated additional features such as move ordering and pruning. Move ordering involved sorting moves in order of their perceived quality to examine the most promising moves first, while pruning was the practice of ignoring moves that were deemed unpromising, thus increasing the program's efficiency. It searched four plies minimax in around 8 minutes, considering seven most plausible moves from each position and evaluated material, mobility, area control and king defense (Bernstein & Roberts1988).

Despite these algorithmic advancements, the program had a relatively shallow search depth, primarily due to the limitations of computer processing power at the time.

On the hardware side, the IBM 704 was the first mass-produced computer with floating-point arithmetic hardware. It had a memory of 4,096 words (each 36 bits long) and could perform about 40,000 operations per second (Anonymous, 2014). The memory and processing power of the IBM 704 far exceeded that of earlier computers, making it possible to run Bernstein's sophisticated program.

The traditional rules of chess were explicitly coded into the program; however, the program lacked strategic concepts like opening theory, endgame knowledge, or tactical themes, such as forks, skewers, and discovered attacks. In terms of playing strength, Bernstein's program was rather weak by human standards. It played at roughly the level of a beginner, frequently missing relatively simple tactical opportunities and making strategic errors. However, it represented a major technical achievement in that it could play a complete game of chess against a human opponent, operating autonomously and making decisions based on a rigorous analysis of the current position.

Bernstein's program demonstrated the feasibility of computer chess as a field of research, and the methods it used became standard components of many later chess programs. It was a milestone in the development of computer chess and played a crucial role in the evolution of artificial intelligence.

Similar advances were being made in the Soviet Union. Georgy Adelson-Velsky, an accomplished mathematician and computer scientist, made significant contributions to the field of computer chess programming during the early 1960s. He was a member of the Institute of Theoretical and Experimental Physics (ITEP) in Moscow, where he worked on one of the first Soviet chess programs that operated on a standard 8x8 board and run on the M-20 computer which had an average performance of 20 thousand instructions per second (Proydakov, 1997).

Adelson-Velsky's program also applied a brute force algorithm that generated all possible legal moves for a given position and used a minimax algorithm to evaluate the potential outcomes of these moves. The scoring system rated each position based on several factors, such as material balance, piece mobility, king safety, and control of the center. The program would then select the move that led to the highest-scoring position.

To manage the computational challenge posed by the game's complexity, Adelson-Velsky implemented various optimization techniques. One of these was an early form of alpha-beta pruning, which helped to reduce the number of branches in the minimax tree that the program had to evaluate. This technique enabled the program to discard branches that would not likely lead to the best move, thereby making the search process more efficient.

While working on this chess program, Adelson-Velsky encountered a significant challenge in efficiently storing and retrieving game positions. The use of a simple array or linked list data structure proved inadequate for managing the rapidly expanding number of game positions. This led him to a crucial insight and a lasting contribution to computer science: the development of the AVL tree in 1962, which he co-invented with his student, Evgenii Landis (Adelson-Velsky & Landis, 1962).

An AVL tree (named after the initials of Adelson-Velsky and Landis) is a type of binary search tree with a self-balancing property. For every node in the tree, the heights of the left and right subtrees differ by no more than one. This balancing property ensures that the depth of the tree is logarithmic in the number of nodes, which guarantees efficient search, insert, and delete operations. The AVL tree was a major breakthrough whose

development was directly linked to the challenges that Adelson-Velsky faced while working on his chess program. In seeking a solution to the problem of efficient storage and retrieval of game positions, he ended up inventing a new data structure that would prove valuable in a wide array of computer science applications (Isenberg, 2018).

Despite these advancements, Adelson-Velsky's program had several limitations. It is not straightforward to compare it with Bernstein's program as they were developed in different contexts and did not play against each other. However we can conclude that Adelson-Velsky's also lacked knowledge of many strategic and tactical elements of chess and had no understanding of opening theory or endgame technique. The scoring function was also relatively simplistic and did not take into account many important positional factors. The playing strength of the program was not very high, but it was not the primary objective. The main purpose of Adelson-Velsky's work was to explore the capabilities of artificial intelligence and to push the boundaries of what computers could achieve.

Another important development during this period came from Richard Bellman, an American mathematician renowned for his work in decision-making processes. He also made substantial contributions to computer chess during the 1950s and 60s. Bellman was best known for his development of dynamic programming, a method used in mathematics and computer science for solving complex problems by breaking them down into simpler subproblems. He applied this technique to his computer chess program with a unique approach, utilizing his method to examine game states.

Bellman's chess program broke away from the traditional brute force method of exploring all possible moves. Instead, he employed dynamic programming to evaluate the game of chess. Dynamic programming relies on storing the results of subproblems to avoid recomputation, making it well-suited to the task of analyzing the vast number of possible chess positions (Bellman, 1965).

In Bellman's method, each game state was evaluated based on a position evaluation function. This function considered factors such as material balance, control of the center, king safety, and pawn structure to assign a numerical score to the position. The system then selected the move that led to the most favorable position. To efficiently explore future game states, the algorithm used stored evaluations of game positions, a technique called "memoization." This process reduces redundant calculations, saving computational resources, and increasing the depth of the game tree that can be explored within a reasonable time.

However, the memoization approach in Bellman's chess program had its own limitations. The most notable was the memory requirement: given the enormous number of possible chess positions, it was not feasible to store the evaluation of all of them with the limited memory of computers in the 1950s and 60s. To overcome this, Bellman's program used an approximate method, storing evaluations for a selected subset of game positions that were considered important.

This technique had its origins in Bellman's work on the "curse of dimensionality," a term he coined to describe the exponential growth in computational requirements when dealing with high-dimensional problems (Bellman, 1961). Chess, with its many potential game states, is an example of such a problem. Through dynamic programming, Bellman sought to alleviate this curse, crafting a balance between computational feasibility and decision-making effectiveness.

Parallel to working on his chess program, Bellman also developed the concept of "Bellman's Principle of Optimality." This principle states that an optimal strategy has the property that whatever the initial state and initial decision are, the remaining decisions must

constitute an optimal strategy with regard to the state resulting from the first decision (Bellman, 1954). Although it was not fully applied in the chess program due to the computational limitations of the time, the principle was foundational for many subsequent developments in the field of decision-making and optimization.

The First Chess Engine World Tournaments

In the previous sections we saw that until the mid-20th century the design and implementation of chess algorithms were primarily theoretical exercises, limited by the constraints of the contemporary computing machinery. However, this landscape began to dramatically shift during the 1960s and early 1970s, as researchers like Richard Bellman pushed the boundaries of what was possible, developing more efficient algorithms to evaluate chess positions.

With those principles in place, the trajectory of chess engine development was set to take a leap. Advancements in computer hardware, a direct consequence of the broader digital revolution, were pivotal in propelling the field into its next stage. The arrival of more robust and faster processing units, coupled with increased memory capacities, enabled chess algorithms to be implemented and run effectively, giving birth to a variety of chess engines. These engines were no longer merely theoretical constructs but functional tools that could analyze and strategize in real-time, bringing us to a significant milestone in the history of chess engines: the first ever chess engine tournament.

The First World Computer Chess Championship took place from August 5 to August 8, 1974, in Hotel Birger-Jarl, Stockholm, Sweden under the auspices of IFIP. It was a four-round Swiss tournament with 13 participants. Kaissa emerged as the sole winner with a perfect score of 4 out of 4 (Isenberg, 2020). Kaissa, a Shannon type A program, employed the alpha-beta technique with a "window" and searched 7 ply full width, with extensions for captures, checks, and forcing moves. It also introduced a feature called "best move service," which stored a table of the 10 best moves to improve move ordering for the alpha-beta method. Another notable feature of Kaissa was the use of a "dummy move," where one side does nothing at its turn to discover threats (Anonymous, 2014).

During the tournament, Kaissa won all its games convincingly. In round 1, it mated Frantz in 34 moves. In round 2, it defeated Tech II in 33 moves (missing a mate in 1). In round 3, it mated Chaos in 36 moves. And finally, in round 4, it mated Ostrich in 67 moves. Kaissa defeated three American entries and one Canadian to become the world champion (ICGA, n.d.).

Kaissa's victory in the First World Computer Chess Championship was attributed to its strong positional understanding, efficient search algorithms, and innovative features such as move ordering and dummy moves. Kaissa's success in the tournament was not just a result of its performance during those four days. It was the culmination of years of development and improvement by a team of programmers in the Soviet Union. Kaissa was originally developed in the 1960s and went through several iterations and enhancements. The program made use of a pruning technique called "the method of analogies," which allowed it to attribute the same score to positions that were similar. This technique, along with tree searching methods for reductions of computational load, made Kaissa a formidable opponent.

Some of the games are saved till today (chessgames.com, n.d), and their level of play, when analyzed with the use of modern machines, appears to be equivalent to a good club player. However, in one game[1] between the engines CHAOS and Chess 4.0, CHAOS, which came second in the final standings, managed an accuracy of 96%. This number is

[1] https://lichess.org/5WvWoLCg#34

exceptional, especially considering that the particular game was as long as 79 moves (the longer a game is, the harder to keep a high accuracy[2]).

The First World Computer Chess Championship marked the beginning of a series of annual events known as the World Computer Chess Championship (WCCC). The WCCC is organized by the International Computer Games Association (ICGA) and features computer chess engines competing against each other. The championship is open to all types of computers, including microprocessors, supercomputers, clusters, and dedicated chess hardware. Over the years, the WCCC has evolved, and rules have been modified to ensure fair competition and accommodate advancements in computer hardware and software.

The impact of the First World Computer Chess Championship goes beyond the individual performance of Kaissa. It paved the way for further advancements in computer chess and inspired the development of new programs and techniques. The tournament served as a platform for researchers and programmers to showcase their innovations and push the boundaries of what was possible in computer chess. Subsequent editions of the World Computer Chess Championship have witnessed the emergence of new champions and the continuous evolution of computer chess engines. For example, Peter Jennings, who developed MicroChess – the first commercially sold microcomputer game – noted significant improvements in the programs at the subsequent 1977 championship in Toronto. He made a special mention of the engine BELLE's pioneering use of a microprogrammed hardware move generator added to the PDP-11, a computer weighing 730 kg (Weik, 1961). This enhancement allowed a much faster generation of positions for analysis and is thought by many to be the route to better chess playing computers. He also highlights a notable game of the tournament between engines DUCHESS and KAISSA in which Kaissa was seemingly ahead, but after being checked it gave away a rook and lost afterwards. After examining the obvious alternative, Kaissa explained why it did not play it. DUCHESS had a queen sacrifice that led to a checkmate in 5, a sequence that none of the present human spectators saw. This was considered a breakthrough and was published in many chess magazines of that time.

Jennings Peter also notes the need for incorporating long-range plans into chess algorithms, an aspect essential to human strategy but absent in computer chess programs of the time. He recognizes the inherent challenge of translating the human ability to plan and strategize into computational algorithms, a critical aspect of artificial intelligence development. Then he identifies a need for improvements in endgame play, referring to the "horizon effect." This term describes a situation where a computer program fails to recognize threats or opportunities beyond a certain number of moves it has been programmed to consider. The endgame, often requiring significant accuracy and long-term planning, was particularly challenging for early chess algorithms.

Jennings was optimistic about future developments and notes the advances in artificial intelligence, pattern recognition, and multiprocessor systems. He astutely predicts these advancements could lead to significant improvements in computer chess. Despite this optimism, he acknowledges that achieving grandmaster level play might not be possible within the century (Jennings, 1978, pp. 108-118). This last statement was later proven wrong.

[2] https://lichess.org/page/accuracy

Chess engines vs Humans

The competition between the chess engines' programmers along with the aforementioned advancements in technology brought the engines in a position where they could face skilled humans.

In 1980, Ken Thompson's chess engine, "Belle," achieved a remarkable victory over Grandmaster Dr. Helmut Pfleger in a simultaneous exhibition. What made this victory particularly out of the ordinary was the fact that Dr. Pfleger had no idea he was playing against a computer program, adding an intriguing twist to the encounter. During the simultaneous exhibition organized by Ken Thompson, Belle was programmed into some of the chessboards, disguising its true identity from both Dr. Pfleger and the other participants. Throughout the games, Belle's programming, developed by Ken Thompson and Joe Condon, proved to be a challenging adversary for Dr. Pfleger. Calculating moves and responding strategically, the engine showcased its understanding of positional play and demonstrated impressive tactical acumen. One of Dr. Pfleger's matches against Belle concluded in favor of the engine, and it was only after the game that Ken Thompson revealed the true identity of Dr. Pfleger's opponent, leaving the grandmaster astounded by the capabilities of the chess engine. However, as he commented, Dr. Pfleger was still not convinced that the engines were able to ever reach grandmaster level of play (Silver, 2022).

A year later, in the 1981 Mississippi State Closed Chess Championship the chess engine CRAY BLITZ got the first place with a perfect score of 5 wins and 0 losses, making it the first computer to ever become a state chess champion. During that tournament CRAY BLITZ achieved a rating of 2258 placing it in the Guinness Book for being the first computer chess Master (Hyatt, 1981).

In 1985, a 22-year-old Garry Kasparov, already the reigning world chess champion, embarked on an unusual exhibition match where the opponents were not individuals but computers, 32 of them, simultaneously. This historic match showcased not just Kasparov's prowess but also presented a fascinating window into the state of computer chess programming at the time.

The simultaneous match took place in Hamburg, Germany, and was organized by the German magazine "Der Spiegel" (Friedel, 2015). The computers in play were running a different commercial chess program. The range of chess software provided a variety of styles and levels of play, albeit limited compared to the human grandmaster.

While the match was in progress, Kasparov would move from one computer to the next, considering and executing his moves. The computers, in turn, had to execute a move when he appeared at their board. Kasparov was able to leverage the machines' inherent weaknesses, most notably their lack of strategic understanding and the inability to think abstractly about the game. He could maneuver in a way that appeared suboptimal to the machines, drawing them into positions where he had significant advantage. This was particularly evident in positions that required a deep positional understanding of chess rather than raw calculating power. Despite being able to calculate multiple moves ahead, the chess engines lacked the human-like strategic awareness, unable to recognize long-term positional advantages or comprehend complex tactical sequences that spanned many moves in a short time. However, on one of the boards he was facing a problem, which later he himself described: “At some point I realized that I was drifting into trouble in a game against one of the "Kasparov" brand models. If this machine scored a win or even a draw, people would be quick to say that I had thrown the game to get PR for the company, so I had to intensify my

efforts. Eventually I found a way to “bluff” the machine with a dubious sacrifice that any modern chess computer would refute in a split-second. But in the good old days of computer chess (to me!) and in my spry youth I could keep coming back to the board fast enough to terrorize the machine with a mating attack” (Kasparov, 2010).

A few years later in 1988, the chess engine HITECH won the Pennsylvania State Chess Championship, after defeating International Master Ed Formanek, crossed the 2400 rating and became the first engine to get the Senior Master title in the U.S. Chess Federation (Berliner, 1988). Later that year, in the 24th American Open, held in Santa Monica, Grandmaster Bent Larsen became the first GM to lose to a computer in a major tournament (chessgames.com, 2004).

From September 22 to September 25 of 1989, the AGS Challenge Match at the New School for
Social Research in New York City, funded by AGS Computers, Inc., was held with the purpose of testing one of the best chess programs against a titled player. The participants in this highly anticipated event were Hitech, the world's highest-rated computer program with an impressive rating of 2407, and International Grandmaster Arnold S. Denker, a former U.S. champion.

Denker boasted a comparable rating of 2410, reflecting his expertise. On paper, the match promised a closely contested battle. However, the outcome surprised everyone, as Hitech secured a remarkable victory with a score of 3.5 to 0.5. However, it is important to note that, at the time of the match, Denker was not as actively involved in chess as he had been during his peak years when he earned the esteemed grandmaster title.

In the first game, Denker opted for a cautious approach, resulting in a hard-fought draw. The second game showcased Hitech's dominance, maintaining control throughout. Despite Denker's valiant efforts and a challenging drawn position, a critical blunder led to his defeat. Another blunder in the third game sealed Denker's loss in the match. In the final game, Hitech delivered a display of brilliance, capitalizing on a minor error by Denker and executing an impressive series of moves, ultimately securing a well-deserved victory. The audience's response was divided, reflecting the intensity of the spectacle. Denker, showing sportsmanship, graciously praised Hitech, acknowledging its exceptional performance, stating, "The machine gave me a real trimming; I am truly impressed" (Berliner, 1989).

In 1992, World Champion Garry Kasparov played 35 blitz games against a beta version of Fritz 3, a chess engine launched by ChessBase, winning 31 games while Fritz won four. This marked a historic occasion as it was the first time Kasparov had lost a game against a computer. In 1994, Fritz tied for first place with Kasparov in the strongest blitz tournament of all time, which included 17 Grandmasters and had an Elo average of 2625. Fritz managed to beat the grandmasters Chernin, Anand, Cvitan, Gelfand, Wojtkiewicz, Hjartarson, Kasparov, Kramnik and Short (in that order) to finish equal first with Kasparov. However, Kasparov won the playoff. Fritz's performance in the tournament was calculated at over 2800 points, demonstrating its growing prowess (Friedel, 2021).

Kasparov vs Deep Blue

The historic chess matches between Garry Kasparov and Deep Blue, a supercomputer developed by IBM, in 1996 and 1997 were a watershed moment in the field of artificial intelligence (AI). These matches were a high-stakes contest between human intellect and machine computation, and they garnered significant media attention worldwide.
The first match took place in Philadelphia in February 1996. Deep Blue, capable of evaluating 200 million positions per second, was a formidable adversary. The world was taken aback when the computer won the first game, making it the first time a reigning world champion had lost to a computer under tournament conditions. However, Kasparov rebounded, employing strategies that pushed Deep Blue to its computational limits, leading to complex positions. His approach was effective, and he won the match 4-2.
Following this defeat, the IBM team enhanced Deep Blue's capabilities. The upgraded version, colloquially known as "Deeper Blue," was twice faster than its predecessor.
The rematch occurred in New York City in May 1997 and Deep Blue emerged victorious with a score of 3.5-2.5. As Kasparov later said “AI had climbed its mount Everest” (Kasparov, 2017).

The creators of DeepBlue, Feng Hsu, Murray Campbell and A. Joseph Hoane Jr. split Carnegie Mellon University's Fredkin Prize of $100,000 which was established in 1980 to be given the first time a computer beat a world chess champion (Loviglio, 1997).The media played a crucial role in these events. The Newsweekly described the 1997 match as the "brain's last stand," encapsulating the essence of the contest between human intellect and machine computation. The Guardian, a British newspaper, referred to the matches as a "historic encounter" that had "captured the public imagination" (Computer History Museum, 2005). Major television networks like CNN provided live updates and analysis, describing the matches as a "showdown" between the world's best chess player and the most advanced chess-playing computer.

The matches were not devoid of controversy. Kasparov accused the IBM team of cheating, suggesting human intervention during the games. He cited a move by Deep Blue in the 1997 match as being too sophisticated for a computer. IBM refuted these allegations, attributing Deep Blue's performance to its programming and computational prowess. Twenty years later the former champion had changed his mind: “I have been asked “Did Deep Blue cheat?” more times than I could possibly count, and my honest answer has always been “I don’t know.” After twenty years of soul-searching, revelations, and analysis, my answer is now “no.” As for IBM, I believe the lengths they went to win were a betrayal of fair competition, but that the real victim of this betrayal was science” (Kasparov & Greengard, 2017 p. 188].

The impact of these matches extended beyond the chessboard. They ignited a debate about AI's capabilities and limitations and spurred interest in computer chess and AI. Chess strategies and tactics were reevaluated, with players incorporating computers into their training for game analysis and preparation. This trend has continued, with top players today heavily relying on computers. Beyond chess, the matches underscored AI's potential. They demonstrated that machines could equal, and even surpass, human intellect in specific areas. This realization has propelled AI advancements, leading to the creation of more sophisticated algorithms and applications.

In summary, the Kasparov-Deep Blue matches were pivotal in AI history. They signified the moment when machines demonstrated their potential to outwit humans in one

of the most intellectually demanding games. The influence of these matches persists, shaping AI development and altering the way chess is played. The media coverage amplified their significance, sparking global debates about the capabilities and future of AI (IBM, n.d.).

In the next decade, many matches were played between top human chess players and chess engines (Chessbase, 2003A), (Chessbase, 2003B), with the last official one being the match between Grandmaster Vladimir Kramnik who was at the time the world champion and Chessbase's Deep Fritz in 2006. Kramnik did not manage to win a single game in the 6-game match and eventually lost 4-2 (Chessbase, 2006).

It is worth mentioning that four years prior Kramnik had tied 4-4 against Deep Fritz in an 8-game match called Brains in Bahrain (Hallsworth, 2003).

Neural Networks rise

One of the strongest engines, if not the strongest, is Stockfish. Stockfish is written in C++ and was developed by Tord Romstad, Marco Costalba, Joona Kiiski and Gary Linscott. It is top contender of the prestigious Top Chess Engines Competition (TCEC), reaching the super finals since season 4 in 2013 (chess.com, n.d.).

One serious factor of its success is that it is open source. The engine's code is freely available to the public, allowing anyone to study, modify, and distribute, fostering a large and dedicated community of developers and researchers who have contributed to the engine's continuous improvement (Smatovic et al., 2023).

Stockfish had firmly established itself as the strongest chess engine in the world before 2017, which is why the chess world was shaken when a neural network computer program called AlphaZero beat Stockfish 8 with a score of 64-36 without a single loss. Super grandmasters stated to chess.com that they were pleasantly surprised...."We normally work with Stockfish and it looks like it's a good program but if we have a program which beats Stockfish so easily it might be a new generation for computers and maybe it's a historical day for chess. We'll see how it will get stronger!", said GM Sergey Karjakin. Others were more skeptical, for instance GM Hikaru Nakamura states: "I think the research is certainly very interesting; the concept of trying to learn from the start without any prior knowledge, so certainly it's a new approach and it worked quite well obviously with go. It's definitely interesting. That being said, having looked at the games and understand[ing] what the playing strength was I don't necessarily put a lot of credibility in the results simply because my understanding is that AlphaZero is basically using the Google supercomputer and Stockfish doesn't run on that hardware; Stockfish was basically running on what would be my laptop. If you wanna have a match that's comparable you have to have Stockfish running on a super computer as well" (Doggers, 2018).

The programmers of AlphaZero, housed within the DeepMind division of Google, did not teach Alphazero the game in the traditional sense. That means no opening book, no endgame tables, and no complicated algorithms evaluating minute differences between moves (McGrath et al., 2022). AlphaZero starts out knowing only the rules of chess, with no embedded human strategies. In just a few hours, it plays more games against itself than have been recorded in human chess history (Kasparov, 2018).

It teaches itself the best way to play, reevaluating such fundamental concepts as the relative values of the pieces. AlphaZero uses a different search strategy compared to traditional chess engines like Stockfish. It searches just 80 thousand positions per second in chess, compared to 70 million for Stockfish. AlphaZero compensates for the lower number of

evaluations by using its deep neural network to focus much more selectively on the most promising variations – arguably a more “human-like” approach to search (Somers, 2018). AlphaZero's Monte Carlo Tree Search (MCTS) scaled more effectively with thinking time than either Stockfish or Elmo, calling into question the widely held belief that alpha-beta search (used by Stockfish) is inherently superior in these domains (Hubert et al, 2017).

According to the words of Matthew Sandler and Natasha Regan, “AlphaZero is about more than just chess. It is a proof of concept, demonstrating AI’s capacity to crack complex problems without the use of human knowledge of strategy. In other words, chess is the testing ground” (Sandler & Regan, 2019, p. 645).

Stockfish’s loss to AlphaZero led to the development of other neural network projects (most notably Leela Chess Zero, Leelenstein, and Alliestein). Leela Chess Zero (commonly abbreviated as Lc0 or Leela) is an open-source chess engine that utilizes neural networks for learning. It is based on the research findings of AlphaGo Zero and similarly starts with a basic understanding of the chess rules and improves its gameplay through self-play and reinforcement learning.While the code of AlphaZero, which seems to be discontinued, remains undisclosed, the dedicated efforts of developers, led by Gary Linscott, resulted in the emergence of Leela. Determining Leela's exact Elo is challenging due to continuous improvement with each game played against itself. However, by 2022, it reached a level comparable to leading chess programs like Stockfish 15 and Komodo 3.

Stockfish has integrated neural networks (Efficiently Updatable Neural Network - NNUE) since version 12 which was released in 2020 (Stockfish, 2020), enhancing its performance. Presently, Stockfish 15 stands among the top chess engines globally. Leela and Stockfish regularly engage in matches, with Stockfish generally prevailing. Nonetheless, Leela's playing strength has been steadily advancing, and by 2021, it secured the position of the second-best chess program, trailing only behind Stockfish 14 (Alexander Thamm, 2023).

5) Impact of Chess Engines

Nowadays, chess engines have become an integral part of a chess player's life in numerous ways, shaping the way both amateur and professional players practice the game. In the past, a player would mostly rely on books, human coaches, or personal experiences to learn and grow in the game. Now, chess engines provide immediate access to highly advanced analysis and understanding of positions, enabling players to quickly evaluate their play.

These engines can play at levels far surpassing the best human players, reaching ratings of above 3500[3] while the highest human rating in history (till August 2023) has been 2882, reached by former world champion Magnus Carlsen on May of 2014[4]. Players can set the difficulty level to match or exceed their skill level, providing an unparalleled opportunity to learn from both victories and defeats. Furthermore, chess engines help in analyzing and dissecting games, making it easier to understand where mistakes were made or where improvements can occur. This functionality is especially beneficial for professional players who use these insights to prepare for tournaments. They can study the games of their future opponents, simulate their styles, and find weaknesses to exploit.

The involvement of chess engines also transforms the way spectators engage with the sport. With real-time evaluation and prediction, viewers can gain a deeper understanding of the game's intricacies, turning even a complex grandmaster battle into a more accessible and engaging experience.

Chess engines also foster a more competitive environment at all levels, from casual play to professional tournaments. They assist in developing new opening theories and evolving the game's meta, forcing players to continually adapt and grow. They have revolutionized the game of chess and it is believed by many that their advantages lead to a positive outcome.

However, the ubiquity of chess engines has led to a debate. Chess enthusiasts have expressed their concerns on over-reliance on engine analysis, potentially stifling creativity or original thought in players. Some argue that it might lead to a homogenization of playing styles, with players often choosing the engine-approved best moves. Additionally, there's the ongoing challenge of ensuring fair play, as the availability of such powerful tools can tempt players into cheating by using them during competitive games.

The first part of this chapter presents the viewpoints of highly-regarded chess players on the influence of chess engines on the game. As individuals who have dedicated significant portions of their lives to chess, their perspectives provide valuable insights into the evolving relationship between the game and technology.

The viewpoints presented in this chapter were collected through a combination of interviews, public statements, and published works.

---

[3] https://ccrl.chessdom.com/ccrl/4040/

[4] https://ratings.fide.com/profile/1503014/chart

Expert Opinions on Chess Engines

“The chess I learnt in the 80s, we no longer play chess like that. The introduction of computers has changed the approach, the way you study completely. Only the two players in front of the board has not changed,” stated former world champion Viswanathan Anand (PTI, 2020), and this seems to represent the consensus of most veteran chess players who were active in both pre and after chess engines era. Grandmaster Judith Polgar says in an interview: “When I was a kid, I was preparing on paper-based material. Later on, in the early 2000s, it was already very clear that if you don’t use the engine for help or advice, you’re going to be falling behind. It was a struggle for me because I’m a very creative player, and with the computer, many times it points out that maybe I have creative ideas, but not necessarily good ones” (Wilkenfeld, 2019). One of the most profound impacts computers have had on contemporary chess lies in the realm of opening preparation. Many openings that were once thought of as unplayable, for the elite levels, have been shown to be palpable to a computer. Many openings that were thought of as playable have been concretely refuted by computers, all the way to checkmate. Chess literature based on such analysis has proliferated in the 21st century, with many books being published each week. The opening preparation involves anticipating an opponent's opening moves well before the game begins, using specialized chess databases. Such foresight typically enables players to make optimal moves early on, steering their adversaries into positions where those well-prepared feel more at ease and likely hold an objective edge. This advantage can be both practical and theoretical. While there's nothing inherently wrong with this—after all, preparing for matches is a natural outcome of the game's competitive nature—the heavy reliance on computers has led to an unintended consequence. In some matches, particularly those involving intricate openings like the 'Sicilian Defense' or the 'Benko Gambit', not a single novel move is played, underscoring the dominance of computer-aided preparation and raising questions about the lack of creative games(Radley College, 2020).

One result of excessive preparation, which computers aid, is that draws have become more frequent, a trend that seems to have been on the rise long before chess engines. According to an analysis by data scientist Randy Olson, draws have become up to three times more frequent since 1850 (Olson, 2014).

Although, there has been a decline in the draw rate since the 1990 which is inversely proportional to the rise of the chess engines. Gary Kasparov, in his book Deep Thinking, answers these questions by supporting that originality in games is still apparent: “Grandmasters have used the ability to prepare with engines and databases to play riskier, more experimental opening variations. Many members of the chess community were afraid that super-strong machines would damage professional chess irreparably by reducing Grandmasters to the role of puppets doing little more than relaying the moves their engines told them were the best. And, to be fair, there is an element of this at a level below elite, as there has always been a class of imitators below the innovators. But at the top level, the effect has been the opposite, with a few glaring exceptions. With the safety net of using an engine at home during preparation, many GMs are more willing to play sharp variations over the board in tournaments. The lure of catching your opponent in a deadly piece of preparation is stronger than the chance of it backfiring. Human recall isn’t perfect, and your opponent might also be well prepared, or come up with something you didn’t think of at home. Either way, there are still plenty of exciting variations and games being played.”

A notable change chess engines brought to the chess world is accessibility. Grandmaster Maurice Ashley, chess professional and commentator, says “The engines have benefited us in many ways. I think the primary one is that it has given us access. Growing up in Brooklyn I had no access, but now you have a device in your pocket that you could say ‘analyze this’ and it will tell you ‘you know what, your move is daft’. There is a teacher right at our disposal” (The Guardian, 2020).

Garry Kasparov also supports this argument as he came to realize that the rise of powerful chess programs could democratize the game on a global scale. His own achievements in chess were not just a result of innate talent and a supportive mother but also due to the geographical advantage of growing up in the Soviet Union. This region had a rich chess tradition, providing budding players with resources like books, coaches, and skilled opponents that were hard to find elsewhere. However, the advent of affordable personal computers with Grandmaster-level chess programs began to level the playing field.

It's interesting to observe the diverse regions from which young chess talents are emerging today. While traditional chess strongholds like the former Soviet regions remain prominent, countries like India, Norway, China, Peru, and Vietnam are also making their mark. In the U.S., the epicenter of chess has expanded beyond New York City to include states like California, Wisconsin, Utah, Florida, Alabama, and Texas. Kasparov notes that for the past twenty years, there's been a focus on how technology can empower individuals globally to pursue their ambitions, irrespective of their location. Chess, in this context, serves as a precursor, demonstrating that with the right tools, talent can flourish anywhere. Grandmaster Anish Giri, ranked number 7 in the world as of July 2023, when asked to comment about the possibility of chess becoming part of the Olympics he underlined that he does not believe that this would be essential for the popularization of the game. He continued by saying that there is already a chess boom, and the fact that you can now follow games online, with commentators and chess engines enabling anyone to evaluate the position without needing to understand the finesses, has helped bring chess to a bigger audience (Cox, 2019).

However, chess engines’ use as an educational tool has some intricacies. Even the strongest chess programs in the world do not have the ability to explain the reasoning behind their brilliant moves beyond elementary tactical sequences. They play a strong move simply because it was evaluated to be better than everything else, not by using the type of applied reasoning a human would understand. “It’s still very useful to have a super-strong machine to play against and to analyze with, of course, but for a nonexpert it can be a bit like asking a calculator to be your algebra tutor,” says Garry Kasparov. The former champion explains the problems of overreliance on a machine with more general concepts. He explains that the acquisition of knowledge should not be limited to merely serving immediate tasks or answering questions, especially if our aim is to attain true wisdom. The convenience of our phones, with access to Google and Wikipedia, allows us to become instant experts on various subjects, which is undeniably beneficial and does not diminish our intelligence any more than encyclopedias, phone books, or librarians did in the past. It is simply a natural progression in how our technology enables us to access and interact with vast amounts of information rapidly, and this evolution is far from reaching its conclusion [Kasparov, 2017, p. 192].

While the danger does not lie in intellectual stagnation or an addiction to seeking instant facts, there is a genuine risk of replacing deeper understanding and insight with superficial knowledge. To truly innovate and create new things, we must guard against substituting profound comprehension with surface-level information. Is there a cognitive

trade-off when integrating technology like chess engines into our learning processes? Observing this integration in chess strongly suggests there is, but it does not always translate to a negative outcome if we remain cognizant of its influence. Kasparov challenges the idea that every intellectual advantage is offset by a loss elsewhere. Major shifts in our cognitive approach can yield overwhelmingly positive outcomes. Historically, young chess players often adopted the style of their earliest coaches, but the question is what happens when the primary influence becomes a computer, which lacks bias, operates free from traditional dogma, and evaluates moves based purely on algorithms. The rise of computers in training and analysis has spawned a generation of players who, like the machines they work with, challenge conventional chess wisdom. Grandmaster Fabiano Caruana, currently (July 2023) ranked number 3 in the world, agrees: “They (young players) have a more concrete approach to chess without biases, just pure calculation, they also have better defensive skills. Chess has become a lot tougher, everyone is extremely well prepared” He elaborates that there is a difference on how young players play in comparison to his generation (those born in early 90s). There is not that much difference in strength, at least not yet, but there is a stylistic difference and much of it could be attributed to how strong chess engines are (Johnson, 2022).

. Part of elite players training involves deciding when to employ computer analysis and when to stick to your own ideas. “The thing is, you cannot rely too much on it. If you’re used to looking at computer lines, your brain will not switch on when it’s time to play a game,” says Grandmaster Maxime Vachier-Lagrave, currently the second highest rated player in France. This approach matches the one applied by Grandmaster Magnus Carlsen, former five-time World Champion and currently highest rated player in the world. His coach, Peter Heine Nielsen reveals: “Magnus is proud of saying that he’s probably the top player who works the least with the computer and is the least influenced by the computer. He wants to trust his own evaluation, his human touch and to keep that” (Robinson, 2021).

Furthermore, the wide accessibility of chess engines has led to a concern on how easily one can cheat. Cheating occurrences are more common in online chess but it also happens on over the board tournaments.”The big problem with hosting online tournaments was that nobody trusted them”, says Woman Fide Master Alexandra Botez. “But the pandemic forced people to do it regardless. So, a lot more people started relying on the anti-cheating algorithm. And the algorithm has gotten very good at detecting cheaters”.
The easiest it became to use an engine to cheat, the more anti-cheating methods were developed. Andy Howie, FIDE international arbiter and organizer, when asked if there
really are irrefutable measures to prevent players from cheating he responded positively: “We most certainly do. At the Candidates Tournament 2020-21 the players’ toilets had 12 cubicles. I had to go round them and check them multiple times per round. To get to the toilets, the player had to go through an airport-style metal scanner. I also have means to detect electronically and by using metal detectors. Someone trying to beat the cistern is going to be caught and flushed out very quickly” (Rivlin, n.d.).

Howie also underlines that top players who rely on their chess careers are less likely to cheat with more to lose on the line. “If they were to be caught cheating, it would be devastating for them, for their careers. But as you come down to the lower ranks, that’s when you’re more likely to find people cheating, your weaker players. People who see it as, it’s not important for them, they get banned for a couple of years. ‘So what? I’ll come and play in a couple of years’ time. I’m not too fussed about it.’ It doesn’t have the impact on them as it does on the top players. It’s not their livelihood.” (Morse,2022a). Another possible

reason that explains the highest ratio of weaker players cheating is that no matter how good the anti-cheating system has become, it is rarely applied in non-elite tournaments.

In addition to the manifest injustice introduced by cheating, which inherently skews the fairness of games, there's a subtler yet deeply corrosive side effect: heightened suspicion within the community. This pervasive mistrust can often result in innocent players being wrongfully accused. The inherent challenge here lies in the fact that, in many cases, it's nearly impossible for players to conclusively demonstrate their innocence. Such allegations, even if baseless, can tarnish a player's experience in tournaments and, more significantly, inflict lasting damage to their reputation in the chess community. A vivid illustration of this issue happened in 2007 when Anna Rudolf, a Hungarian International Master, faced a challenging situation during the Vandoeuvre Open. Despite her stellar performance, she was accused of cheating based on her frequent use of lip balm, with some suggesting it was a coded way to receive moves. This baseless claim overshadowed her achievements at the tournament. While Rudolf was eventually cleared, the incident underscores the ripple effects of a distrustful environment in chess. Cheating does not just disrupt the game; it creates a backdrop where even innocent actions can be misinterpreted (chessdom.com, 2008).

A more recent example took place during a Titled Tuesday tournament. Grandmaster Fabiano Caruana implied that his opponent, a Greek FIDE Master, was perhaps playing 'a bit too well,' insinuating potential cheating. This sort of public accusation from a figure of Caruana's stature could be potentially devastating for a lesser-known player's reputation. Fortunately, in this instance, the Greek FM was live-streaming his games, offering clear evidence of his honest gameplay (C-Squared, 2022).

Community Perspectives through Survey Analysis

After the dive into the insights of elite players in the previous chapter, a natural question arises: How do the wider ranks of the chess community feel about the same issues? While the thoughts of the chess elite offer a perspective enriched by years of top-level competition, the view of the broader chess-playing population is equally vital. This chapter transitions from the summit of the chess world to its vast base, bringing the voices of players of all stripes to the fore. To this end, a comprehensive survey was designed and disseminated, targeting an array of participants varying in age, experience, and expertise. Through a series of calibrated questions, the intent was to gauge the overarching sentiment towards chess engines, their implications on gameplay accuracy, creativity, and the potential challenges they introduce, such as cheating.

Survey Conduction:

Our survey was disseminated digitally across several popular chess forums, and social media platforms and local chess clubs in Athens. Inclusion criteria ensured that

participants had a basic familiarity with chess, but no restrictions were placed on age, rating, or years of involvement, ensuring a wide and diverse set of responses.
The survey ran from 14/05/2023 to 17/06/2023 and received 217 submissions. Three of them were discarded due to them being deemed non-serious or inconsistent with the survey's intent. Thus the final analysis includes 214 answers.

During the course of the survey distribution, a potential geographical bias was noted. Approximately 19% of the responses originated from local chess clubs in Greece, which could influence the overall representation and outcomes of the survey. However, upon analyzing the data, the responses from Greek partitioners were found to be consistent with the overall trends and sentiments of the remaining 81%. Consequently, initial concerns regarding a potential geographical bias proved to be unfounded.

Survey characteristics:

The survey consists of three parts.

The first part aims to gather demographic information of the participant by including the following fields:

1) Rating
2) Name
3) Age
4) Years involved in chess
5) Currently receiving any training or coaching for chess (YES/NO question)

All the above questions of this part were optional; however, the majority of the participants chose to answer them.

The second part aimed to a quantitative assessment of matters that were most mentioned during the research of the previous chapters. The participants were presented with four statements and had to rate them from 1 to 5, where 1 means that they strongly disagree with the statement and 5 means that they strongly agree with the statement.
The statements were the following:

1) The availability of chess engines has made chess more accessible to a wider audience.
2) Chess engines have made the game of chess more accurate.
3) Chess engines have slowed down the creativity of chess players.
4) Chess engines made cheating a serious problem in tournaments.

The rating of all the statements in the second part was mandatory.
(The above statements will be represented as Accessibility, Accuracy, Creativity Loss, Cheating respectively.)

The third and last part of the survey involved open-ended questions for qualitative insights on the participants' nuanced views on the multifarious effects of chess engines on the game. It provided a space for players to add anything that might have been missed, encouraging a rich tapestry of personal opinions and stories.
The questions were the following:

1) In your opinion, what positive changes chess engines brought to the game?
2) In your opinion, what negative changes chess engines brought to the game?
3) Anything else you would like to add?

All the questions in this part were optional.

Analysis:

Participant demographics:

The participants of the survey that filled the rating field had a mean rating of 1540, ranging from unrated people to 2500.

The mean age of the participants is 30 years, ranging from 8 to 80.

The mean years of involvement with the game of the participants is 12 years ranging from 0.2 years to 67 years.

19.63% of the participants are currently training for chess while 78.97% are not.

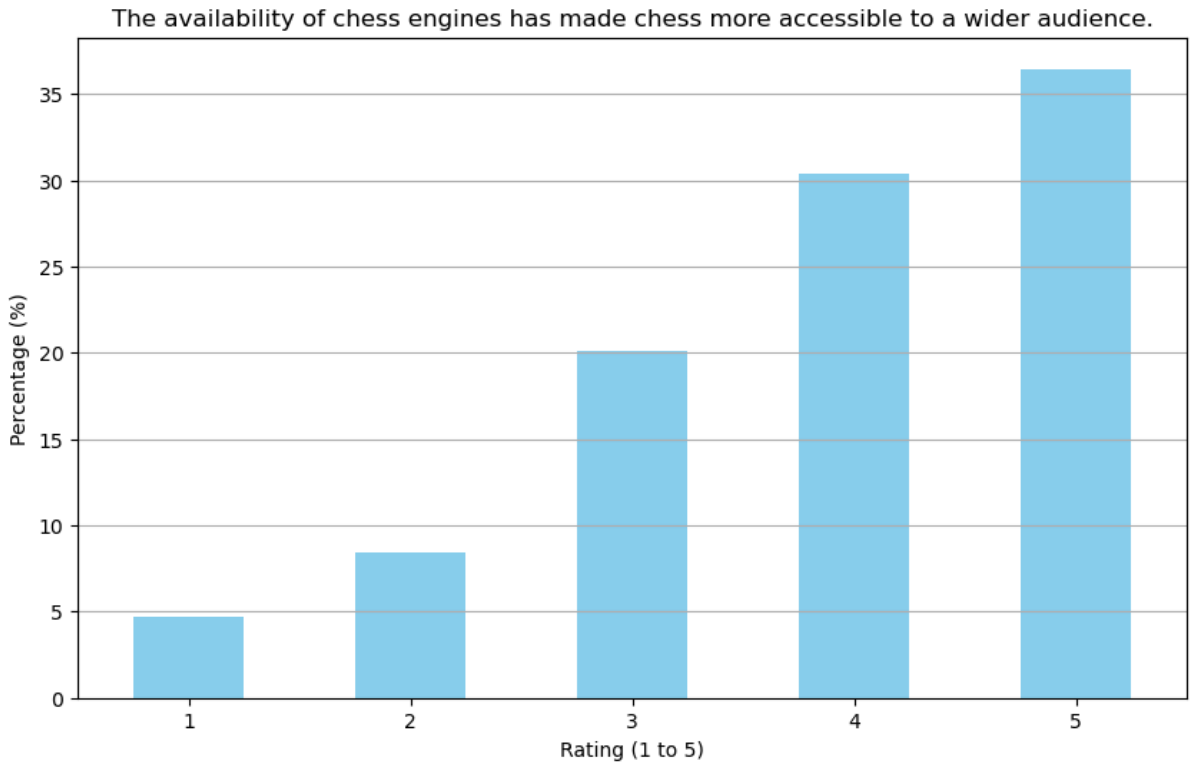
Linear scale questions:

1) The availability of chess engines has made chess more accessible to a wider audience.

Overall Insight: The respondents generally felt that chess engines have contributed to making the game more accessible. This is reflected in the mean score of 3.86, which leans towards agreement with the statement.

Distribution: A substantial portion of the participants, 66.82%, chose either 4 or 5, further accentuating the sentiment that engines enhance accessibility. Notably, the mode – the most frequent response – was a strong agreement with a score of 5.

Variability: With a standard deviation of 1.14, the responses exhibit a moderate spread around the mean. However, the predominant sentiment remains positive about the role of engines in popularizing chess.

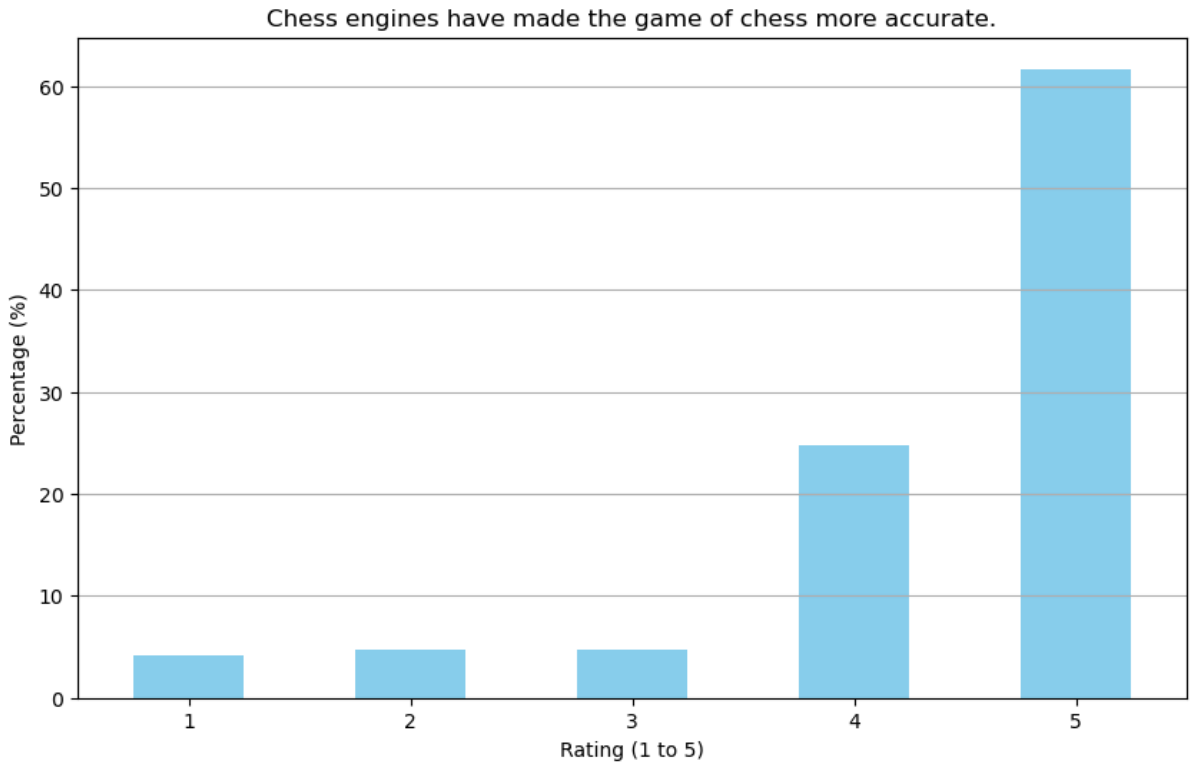


2) Chess engines have made the game of chess more accurate.

Overall Insight: Respondents overwhelmingly feel that chess engines have enhanced the accuracy of the game. This sentiment is strongly supported by an impressive mean score of 4.35, clearly indicating an agreement with the assertion.

Distribution: A notable 86.45% of participants chose either 4 or 5, underscoring the prevailing belief in the positive impact of engines on game accuracy. The mode stands out distinctly with a score of 5, signaling a strong consensus around the notion that engines significantly bolster game precision.

Variability: The responses presented a standard deviation of 1.05, indicating a moderate dispersion in answers. However, the prevailing sentiment is decisively in favor of the belief in the bolstered accuracy due to chess engines.

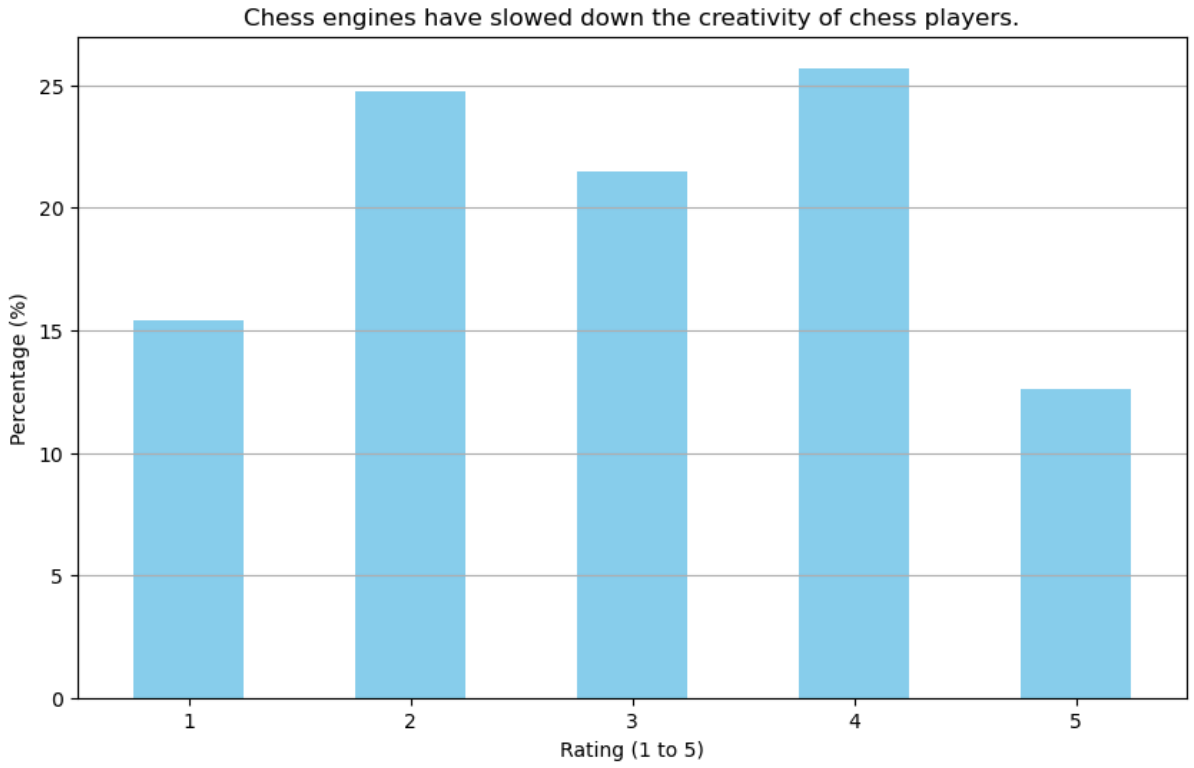


3) Chess engines have slowed down the creativity of chess players.

Overall Insight: The perception of chess engines' impact on player creativity paints a diverse picture. With a mean score close to 3 (2.95, to be precise), there seems to be a delicate balance between those who believe engines stifle creativity and those who believe otherwise.

Distribution: From the participants, 40.19% gave ratings of 1 or 2, indicating a sentiment that chess engines do not necessarily impede, and might even bolster, creativity.. Conversely, 38.32% of respondents chose either 4 or 5, suggesting that a nearly equivalent proportion believes engines may indeed hamper creativity. It's particularly noteworthy that the mode, or most frequent response, was 4, suggesting a lean towards engines being somewhat hurtful towards creativity.

Variability: The standard deviation of 1.28 signals a wider range of opinions compared to other questions. This degree of dispersion underscores the debate around this topic within the chess community.

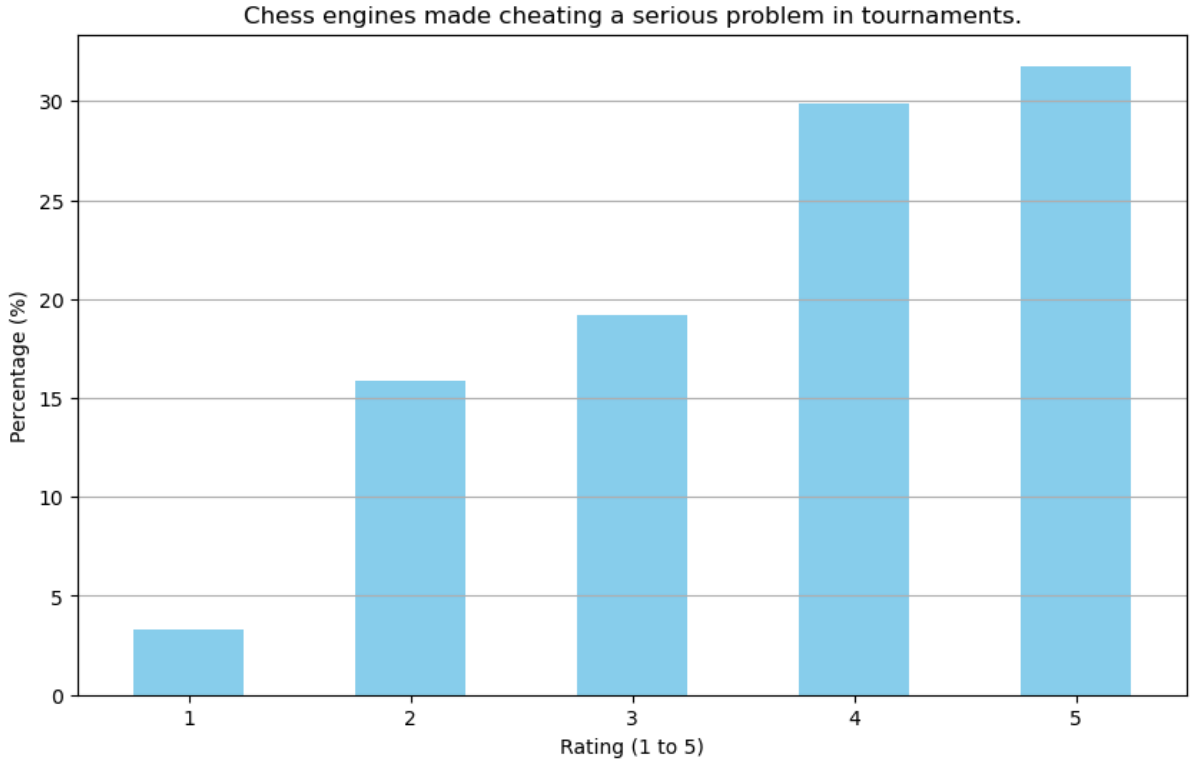


4) Chess engines made cheating a serious problem in tournaments.

Overall Insight: A notable proportion of respondents hold the belief that chess engines have escalated cheating concerns in tournaments. This is evident from a mean score of 3.71, which trends towards the agreement side of the scale.

Distribution: A significant 61.68% of participants marked either 4 or 5, signaling their affirmation that chess engines have indeed introduced or amplified cheating concerns in the competitive realm. The mode for this question stands out, with a score of 5, which indicates strong agreement with the statement, underscoring the weight of the issue in the minds of many respondents.

Variability: With a standard deviation of 1.17, there is a moderate spread in responses. While the dominant sentiment leans towards recognizing the cheating problem linked to engines, the variability suggests that there's still a diversity of opinion on this topic within the community.



Correlation between demographic and linear scale answers

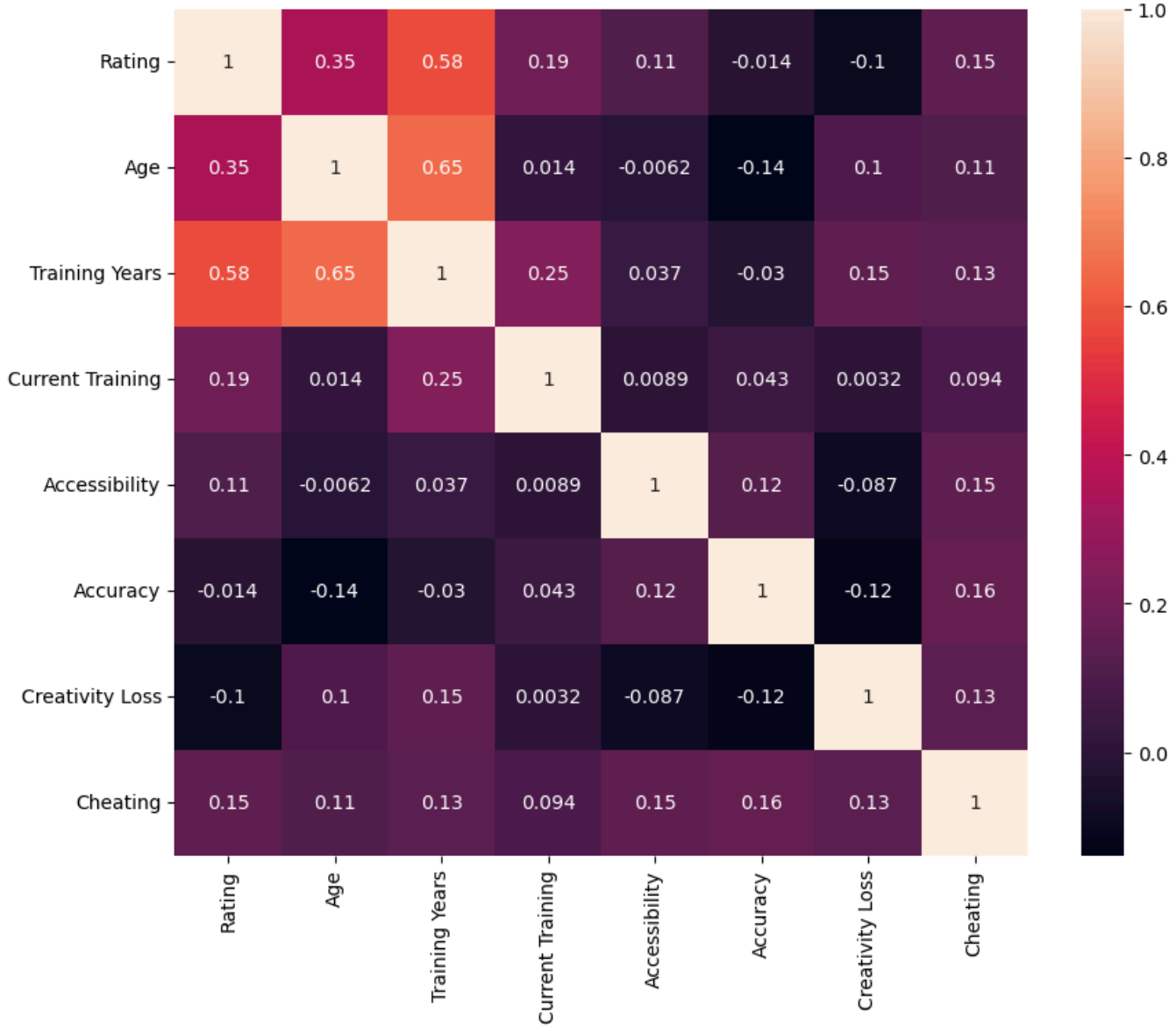
Spearman's rank-order correlation was used in order to inspect possible correlation between the answers in the four linear questions and the participant's demographic information (Rating, Age, Years of training and if they currently train).

Spearman's rank correlation is a non-parametric measure of the monotonic relationship between two datasets, which means it assesses how well an arbitrary monotonic function could describe the relationship between two variables, without making any other assumptions about the particular nature of the relationship between the variables.

The values in the following heatmap represent the correlation coefficients.

Correlation coefficients range from -1 to 1.

- 1 indicates a perfect positive correlation.
- -1 indicates a perfect negative correlation.
- 0 indicates no correlation.



The model indicates that the correlation between the answers in the four linear questions and the participant's demographic information is very weak, <0.2 in all cases, meaning that the answers seem to be unrelated to the age, rating or training habits of the participants. This result was also supported from further regression analysis.

Age Groups

To achieve deeper investigation, the participants were divided into groups according to their age. The chosen groups were <22, 23-40 and 40+ years.
When categorizing participants based on age, the chess landscape during their formative years of learning and playing was taken into account. As presented in the previous history chapter, chess engines have seen a dramatic evolution over the past few decades

Here's the rationale behind the age groupings:

- <22 years old: Players in this category started their chess journey in an era where chess engines were already dominant. These individuals likely used highly advanced chess software as integral tools in their learning process, shaping a unique approach to the game.

- 23-40 years old: This age bracket represents players who began learning chess during the transitional period of chess engines. While these engines were good, they had not reached their full potential. Thus, players from this era balanced traditional learning with the emerging benefits of technology.

- 40+ years old: The most veteran group, these players typically started chess when chess engines were still in their nascent stage. Their formative years were less influenced by technology, leaning more on traditional methods and human mentorship.

Mean scores of each question for each Age Group:

Accessibility <22 3.857143 22-40 3.878049 40+ 3.761905	Accuracy <22 4.385714 22-40 4.439024 40+ 4.119048
Creativity Loss <22 2.914286 22-40 2.768293 40+ 3.142857	Cheating <22 3.614286 22-40 3.719512 40+ 4.023810

In the ANOVA tests that were conducted, age groups acted as the independent variable, and Accessibility, Accuracy, Creativity Loss, and Cheating as dependent variables.
The F-statistic is a ratio of variances. It's computed by taking the ratio of the variance explained by the model (between-group variance) to the variance not explained by the model (within-group variance).

- An F-statistic is computed for each of the metrics.
- If all group means are equal, the F-statistic would be close to 1.
- A larger F-statistic indicates that the group means are more different from each other than we would expect by chance.

The P-value associated with the F-statistic tells us the probability of getting an F-statistic as extreme as, or more extreme than, the observed value, given the null

hypothesis is true. If this probability is very low (typically ≤ 0.05), we reject the null hypothesis (The null hypothesis for ANOVA is that there is no difference among group means.), suggesting at least two group means are significantly different from each other.

Accessibility F-statistic: 0.14717776594689286 P-value: 0.8632383205659313	Accuracy F-statistic: 1.3732998690046851 P-value: 0.2557590667844242
Creativity Loss F-statistic: 1.1854841781802177 P-value: 0.3078364830212332	Cheating F-statistic: 1.7095427399895042 P-value: 0.183705436993428

While there were observable differences in mean scores across age groups, the ANOVA results suggest these differences are not statistically significant (P-value > 0.05). This might indicate that age, categorized by exposure to chess engines, does not play a significant role in influencing the opinions of the participants on these metrics.

Rating Groups

Similarly, the participants were divided into groups according to their rating. The groups were:

- Novice: <1000
- Intermediate: 1001-1500
- Advanced: 1501-2000
- Expert: 2001+

Mean scores of each question for each Rating Group:

Accessibility <1000 3.631579 1000-1500 3.833333 1501-2000 3.830508 2001+ 4.125000	Accuracy <1000 3.947368 1000-1500 4.520833 1501-2000 4.457627 2001+ 4.125000
Creativity Loss <1000 3.263158 1000-1500 3.104167 1501-2000 2.677966 2001+ 2.958333	Cheating <1000 3.157895 1000-1500 3.895833 1501-2000 3.542373 2001+ 4.250000

ANOVA test results:

Accessibility F-statistic: 0.6983853715100412 P-value: 0.5544740023736704	Accuracy F-statistic: 1.9579178391519896 P-value: 0.12291021837063208
Creativity Loss F-statistic: 1.4620293906631503 P-value: 0.22737293088922209	Cheating F-statistic: 4.252351163993382 P-value: 0.006508188901478772

The only aspect where the results indicate a significant difference in perceptions across the rating groups is Cheating. With an F-statistic of 4.2524 and a P-value of 0.0065, the differences in perceptions of cheating across the rating groups are statistically significant at the 0.01 level. This implies that there's a less than 1% chance that the observed differences in perceptions occurred by random chance.

- Novices, with ratings below 1000, seem to have a relatively lower concern of engine-assisted cheating with a mean value of 3.158. This could perhaps be attributed to their limited exposure to competitive chess or lesser awareness of advanced cheating methods.

- Intermediate players, those with ratings ranging from 1000-1500, perceive chess-engine cheating a bit more with a mean value of 3.896. This group might have started encountering sophisticated cheating methods, or they may be more conscious of the competition and its associated malpractices.

- Advanced players (1501-2000) have a slightly lower perception compared to intermediate players, with a mean of 3.542. This decrease might be due to their proficiency in recognizing genuine skill versus engine-assisted play or perhaps being more focused on their gameplay than potential cheating.

- Expert players, those with ratings above 2001, have the highest numbers with a mean value of 4.250. This suggests a heightened awareness or concern about chess-engine cheating, perhaps due to their higher stakes in competitive play. At higher levels of competition, the potential rewards (financial, reputational) are significant, making cheating a more pronounced concern.

Qualitative Insights

Upon analyzing the answers of the open-ended questions there were noticed themes in the participants' opinions.

Thematic overview on the participants' perception on the positive impact of chess engines:

- **Evaluation and Accuracy:** The advent of chess engines has revolutionized the way players analyze and evaluate their games. These engines provide a reliable framework for understanding the implications of each move, ensuring that players can identify both subtle mistakes and masterful tactics. The precision of their evaluations has enabled players of all levels to refine their strategies, delve deeper into the complexities of the game, and play more accurately.
- **Self-learning, Improvement and Efficiency:** Chess engines serve as invaluable tools for players committed to honing their skills. They accelerate the learning process, offering insights into various phases of the game, from openings to

endgames. This access to instant feedback and high-quality compositions ensures that players can continuously improve, decreasing their reliance on coaches and fostering a more independent, self-taught approach to mastery. Competitive players, often pressed for time, find chess engines indispensable for efficient game preparation. The ability to receive instantaneous evaluations and conduct swift post-game reviews ensures that players can maximize their preparation, allowing them to focus on nuanced strategies and better anticipate opponents' moves.

- **Openings and Theoretical Advancements:** With the support of engines, there has been a profound deepening of understanding in opening theory. Players can explore and evaluate unconventional or revitalized opening lines, often leading to groundbreaking theoretical advancements. This objectivity in evaluating diverse opening strategies has enriched the game's dynamism, offering fresh perspectives and approaches to age-old dilemmas.
- **Accessibility and Democratization:** Chess engines have democratized access to elite-level insights, leveling the playing field for enthusiasts around the world. Now, irrespective of geographical location or financial constraints, aspiring players can tap into sophisticated analytical tools, reducing their reliance on human experts and bridging the gap between amateurs and professionals.
- **Innovation and Creativity:** The insights from engines often challenge traditional chess wisdom, prompting players to consider unconventional and dynamic moves. This wave of innovation has injected a fresh dose of creativity into the game, inspiring players to experiment with unique strategies and explore paths less traveled, thereby expanding the boundaries of chess artistry.
- **Growth and Popularity:** Chess engines have played a pivotal role in the game's recent resurgence in popularity. They've empowered content creators to produce nuanced analysis, making chess more engaging for broader audiences. This technological embrace has not only elevated the quality of chess content but also drawn new enthusiasts to the game.
- **Objectivity and Truth:** By offering an objective standard, chess engines have settled many debates surrounding the value of specific positions or moves. Their evaluations present a truth, a definitive insight into the game's nature, furthering our understanding of whether chess, with its current rules, is solvable and the nature of that solution.

Thematic overview on the participants' perception on the negative impact of chess engines:

- **Cheating**: The advent of chess engines has given rise to a significant problem in both over the board and online chess: cheating. This issue has reached such an extent that many players now view online chess platforms with mistrust. Even top-level players are not immune, with some facing accusations of unfair play. Special devices dedicated to cheating have also emerged, further exacerbating the issue.
- **Over-Reliance on Engines:** Many players, particularly those new to the game, have come to depend heavily on chess engines for guidance. This dependency often manifests as players memorizing moves suggested by engines rather than understanding the logic behind them. The result is a decline in human evaluation and

analysis, with players finding it challenging to assess positions without an engine's assistance.

- **Loss of Diversity and Creativity:** Chess engines have inadvertently steered players towards certain strategies and away from others. Some openings, once popular and revered, are now avoided because they're perceived as inferior by engines. This reduction in strategic diversity is accompanied by a decline in creative play. Players often focus on tiny advantages suggested by engines, resulting in an increased number of drawn games. The game's romantic and fantastical elements, which once inspired awe, have also faded.
- **Impact on Learning and Improvement:** Chess engines have reshaped the learning curve for many players. Beginners, in particular, may be misled by engine recommendations. The easy availability of solutions and evaluations has also diminished the emphasis on post-game analysis, a critical learning tool. Consequently, players' analytical skills have suffered, with many leaning towards laziness and reluctance to delve deep into the game's principles. The emphasis has shifted towards memorizing openings, leading to a predictable and less spontaneous form of play.
- **Engine Influences Spectators' Expectations:** With the widespread use of engines, spectators' perceptions and expectations of players have shifted. There's a prevalent belief that players should always opt for the engine's top move, leading to unrealistic expectations. This trend has also led to undue criticism, with even grandmasters facing flak based on engine evaluations.
- **Philosophical and Aesthetic Concerns:** Chess, once viewed as an art form, is now caught in debates revolving around the accuracy and beauty of moves. The game's mystical and philosophical aspects, which once intrigued many, are overshadowed by cold, precise engine evaluations.
- **Technical and Logistical Concerns:** The presence of engines has necessitated increased security measures to prevent cheating, especially in competitive settings. This has been particularly challenging for smaller tournaments, where the ubiquity of technology, such as smartphones, poses a constant threat.

The provided responses reflect mixed feelings towards chess engines, with an overall leaning toward the positive side. A majority of respondents see engines as a beneficial tool, recognizing their utility in improving one's game, analyzing positions, and preparing for tournaments. However, alongside the acknowledgment of their benefits, there are concerns, the most frequently mentioned being cheating, over reliance and loss of creativity. However, the general consensus from the responses indicates that, while there are reservations and concerns, engines are largely appreciated and seen as a positive influence on the game when used appropriately.

6) Conclusions

Over the course of this research, the transformative influence of technology on the venerable game of chess emerged unmistakably. The birth and evolution of chess engines, pivotal in molding the contemporary landscape of chess, have unveiled both remarkable opportunities and pressing concerns.

The intricate technical underpinnings of these engines, as explored in this thesis, showcased the harmonization of chess strategy with computational prowess. Their progressive trajectory, running parallel to broader technological milestones, underscores their growing refinement and impact on chess.

Players, spanning the spectrum from novices to seasoned maestros, have tapped into the capabilities of these engines to hone strategies, dissect games, and bolster gameplay. This was corroborated by survey results and expert dialogues, which underscored the profound imprint of engines on today's chess education and competition.

But beyond these academic and professional spheres, chess engines have democratized the game. They've played an instrumental role in making chess more accessible to the masses, breaking down barriers of geography, age, and expertise. The digital realm, teeming with online platforms and apps powered by these engines, has witnessed a surge in engagement, bringing chess to fingertips worldwide. This heightened accessibility has not only nurtured budding enthusiasts but has also amplified the global popularity of the game.

However, intertwined with these positives are shadows of over-reliance and potential malpractices. Concerns are burgeoning in the chess fraternity about illicit engine use during live matches, jeopardizing the ethos of fair competition. Moreover, the hazard of players becoming overly dependent on these digital aids, possibly at the expense of their intuitive grasp of the game, cannot be overlooked.

In essence, while chess engines are undeniably revolutionary, they echo the age-old adage that with great power comes great responsibility. Like any potent technology, their true value lies in how they're used. They should serve as tools to enhance understanding and skill, not as crutches that diminish the human element of chess. As we navigate the future of chess in this digital age, the onus is on players, educators, and the broader community to harness the power of chess engines responsibly, ensuring the game's integrity and spirit remain intact.

7) Suggestions for Further Research

1) **Influence of chess on technological advancement**

Throughout the research, several instances emerged where the ambition to enhance chess engines catalyzed algorithmic innovations. Notable examples include the development of alpha-beta pruning and Bellman's Principle of Optimality.

A paper titled "Chess as the Drosophila of AI" [McCarthy, 1990] posited a critique, questioning if chess genuinely played a pivotal role in AI's progression. The contention was that earlier chess engines primarily harnessed brute computational power, focusing on calculating numerous possible moves rather than integrating genuine AI techniques. However, this perspective underwent a transformation with the advent and integration of neural networks in chess engines. Given this evolution, it warrants a deeper examination of chess's role in shaping the trajectory of technology and AI.

2) **Ethical Dimensions of Chess Engines**

Delve deeper into the ethical implications of using engines, especially in competitive settings. Investigate the ways tournaments can ensure fairness and what kind of regulations or technologies could help.

3) **Future of Chess Engines**

Speculate on the future advancements in chess engines as technology continues to evolve. The landscape of chess engines is poised for transformative advancements with one of the most promising technological frontiers being quantum computing. The integration of quantum computing promises not only to enhance the capabilities of chess engines but also to deepen our understanding of the game's near infinite complexities.

8) References

1. Akl, S. G., & Newborn, M. M. (1977). The principal continuation and the Killer Heuristic. *Proceedings of the 1977 Annual Conference on - ACM '77*. https://doi.org/10.1145/800179.810240

2. Boulé, M. (2002). *An FPGA move generator for the game of chess* (thesis). McGill University, Montreal, Canada.

3. Ellis Jones, P. (2017, February 9). *Generating legal chess moves efficientlyPeter Ellis Jones*. Peter Ellis Jones. https://peterellisjones.com/posts/generating-legal-chess-moves-efficiently/#:~:text=Pseudo%2Dlegal%20move%20generation,enemy%20pieces%20on%20(captures)

4. Frayn, C. (2005, August 1). *Computer Chess Programming Theory*. Computer Chess Programming theory. https://www.frayn.net/beowulf/theory.html

5. GeeksforGeeks. (2022, June 13). *Minimax algorithm in Game theory: Set 1 (introduction)*. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

6. Hyatt, R. M. (2004). Chess program board representations. https://web.archive.org/web/20130212063528/http://www.cis.uab.edu/hyatt/boardrep.htm. Originally published at http://www.cis.uab.edu/hyatt/boardrep.html. Archived from the original on 12 February 2013

7. Pettersson, J. (2006, December 18). *[guide] move generation*. Mediocre Chess. https://mediocrechess.blogspot.com/2006/12/guide-move-generation.html

8. Schaeffer, J. (1989). The history heuristic and alpha-beta search enhancements in practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*(11). https://doi.org/10.1109/34.42858

9. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and Tree Search. *Nature*, *529*(7587). https://doi.org/10.1038/nature16961

10. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2017, December 5). *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*. arXiv.org. https://arxiv.org/abs/1712.01815

11. Wang, B. (2021, January 11). *Monte Carlo Tree Search: An introduction*. Medium. https://towardsdatascience.com/monte-carlo-tree-search-an-introduction-503d8c04e168

12. Walter, J. (2019, December 24). The mechanical turk: How a chess-playing hoax inspired Real Computers. Discover Magazine. https://www.discovermagazine.com/technology/the-mechanical-turk-how-a-chess-playing-hoax-inspired-real-computers

13. Schaffer, S. (1999). Monoskop. https://monoskop.org/images/2/2d/Schaffer_Simon_1999_Enlightened_Automata.pdf

14. Montfort, Nick (2003). Twisty Little Passages: An Approach to Interactive Fiction. MIT Press. p. 76. ISBN 0-262-63318-3

15. Automates la nature torres - cyberneticzoo.com. (n.d.). https://cyberneticzoo.com/wp-content/uploads/2011/01/Automates-La-Nature-Torres-1914.pd

16. Velasco, JJ (2011-07-22). "Historia de la tecnología: El ajedrecista, el abuelo de Deep Blue". Hipertextual (in European Spanish). Retrieved 2017-08-14.

17. Tula Giannini and Jonathan P. Bowen. Life in Code and Digits: When Shannon met Turing. 2017. DOI: 10.14236/ewic/EVA2017.9

18. Kasparov, G., & Friedel, F. (2017). Reconstructing turing's "Paper machine." *EasyChair Preprints*. https://doi.org/10.29007/g4bq

19. Copeland, B. J., Bowen, J. P., Sprevak, M., Wilson, R. J., et al. (2017) The Turing Guide. Oxford University Press.

20. Turing, A. M., & Copeland, B. J. (2004). The essential turing. Oxford University Press.

21. Alan Turing vs Alick Glennie (1952) Turing Test. (n.d.). https://www.chessgames.com/perl/chessgame?gid=1356927

22. Friedel, F. (2017, September 23). Reconstructing turing's "Paper machine." Chess News. https://en.chessbase.com/post/reconstructing-turing-s-paper-machine

23. CS221. (n.d.). https://stanford.edu/~cpiech/cs221/apps/deepBlue.html

24. Shannon, C. E. (1950a). Programming a computer for playing chess. Taylor & Francis.

25. Claude Shannon. Claude Shannon - Chessprogramming wiki. (n.d.). https://www.chessprogramming.org/Claude_Shannon#cite_note-3

26. Soni, J. (2017, August 16). *The man who built the chess machine*. Chess.com. https://www.chess.com/article/view/the-man-who-built-the-chess-machine

27. Prinz, D. (1988). Robot chess. Computer Chess Compendium, 213–219. https://doi.org/10.1007/978-1-4757-1968-0_21

28. Anderson, H. (1986). Metropolis, Monte Carlo, and The MANIAC. https://sgp.fas.org/othergov/doe/lanl/pubs/00326886.pdf

29. Isenberg. (2020). Maniac I. MANIAC I - Chessprogramming wiki. https://www.chessprogramming.org/index.php?title=MANIAC_I

30. [Lewis, N. (2022). 70 years of electronic computing. LANL Discover RSS. https://discover.lanl.gov/news/0412-maniac/

31. Lasar , M. (2011, August 5). Brute Force or intelligence? the slow rise of Computer Chess. Ars Technica. https://arstechnica.com/gaming/2011/08/force-versus-heuristics-the-contentious-rise-of-computer-chess

32. Bernstein, A., & de V. Roberts, M. (1988). Computer v chess-player. Computer Chess Compendium, https://d1yx3ys82bpsa0.cloudfront.net/chess/computer-v-chessplayer.bernstein-roberts.scientific-american.june-1958.062303059.pdf.

33. IBM and Chess " chessmaniac. (2014.). https://www.chessmaniac.com/ibm-and-chess/

34. Proydakov , E. (1997). M-20 Computer. russian computer museum. https://www.computer-museum.ru/english/m20.htm

35. Georgy Adelson-Velsky, Evgenii Landis (1962). An algorithm for the organization of information. Proceedings of the USSR Academy of Sciences, (Russian) English translation by Myron J. Ricci in Soviet Mathematics Doklady, No. 3

36. Isenberg et al., G. (2018). Revision history of “Georgy Adelson-Velsky.” Revision history of “Georgy Adelson-Velsky” - Chessprogramming wiki. https://www.chessprogramming.org/index.php?title=Georgy_Adelson-Velsky&action=history

37. Bellman, R. (1965). On the Application of Dynamic Programming to the Determination of Optimal Play in Chess and Checkers. Proceedings of the National Academy of Sciences of the United States of America, 53(2), 244–247. http://www.jstor.org/stable/72572

38. Bellman, Richard Ernest (1961). Adaptive control processes: a guided tour. Princeton University Press.

39. Bellman, R. (1954). The theory of dynamic programming - RAND corporation. https://www.rand.org/content/dam/rand/pubs/papers/2008/P550.pdf

40. Isenberg, G. (2020a). WCCC 1974. WCCC 1974 - Chessprogramming wiki. https://www.chessprogramming.org/WCCC_1974

41. Anonymous. (2014). Kaissa Chess Program " Chessmaniac. https://www.chessmaniac.com/kaissa-chess-program/

42. ICGA. 1st World Computer Chess Championship - Stockholm 1974 (ICGA Tournaments). (n.d.). https://www.game-ai-forum.org/icga-tournaments/tournament.php?id=7

43. chessgames.com. (n.d.). The Chess Games of Kaissa (computer). https://www.chessgames.com/perl/chessplayer?pid=48721

44. Weik, Martin H. (March 1961). "Programmed Data Processor". Ed Thelen's Nike Missile Web Site. A Third Survey of Domestic Electronic Digital Computing Systems. Archived from the original on March 21, 2022. Retrieved July 6, 2018

45. Jennings, Peter (January 1978). "The Second World Computer Chess Championships". BYTE. pp. 108-18.

46. Silver, A. (2022, October 28). The pioneers of cheating in Chess. Chess News. https://en.chessbase.com/post/the-pioneers-of-cheating-in-chess

47. Hyatt, R. (1981). Cray Channels, 3(2) http://archive.computerhistory.org/projects/chess/related_materials/text/3-1%20and%203-2.Cray_Channels_Vol-3_No-2.Checkmate.Cray_Blitz.Hyatt.1981/Cray_Channels_Vol-3_No-2.Checkmate.Cray_Blitz.Hyatt.1981.062303019.sm.pdf

48. Friedel, F. (2015, June 6). Kasparov and Thirty Years of Computer Chess. Chess News. https://en.chessbase.com/post/kasparov-and-thirty-years-of-computer-chess

49. Gary Kasparov, "The Chess Master and the Computer," The New York Review of Books 57 February 11, 2010

50. Hans Berliner (1988). Pennsylvania State Chess Championship - HiTech Becomes First Computer Senior Master. AI Magazine Volume 9 Number 3 (© AAAI), pdf]

51. Bent Larsen vs deep thought (computer) (1988). (n.d.). https://www.chessgames.com/perl/chessgame?gid=1472093

52. Hans Berliner HITECH CHESS REPORT - Hitech Defeats Denker in AGS Challenge Match AI Magazine Volume 10 Number 1 (1989) (© AAAI) https://link.springer.com/chapter/10.1007/978-1-4613-9080-0_5]

53. Friedel, F. (2021, November 9). Thirty Years! happy birthday Fritz (2). Chess News. https://en.chessbase.com/post/thirty-years-happy-birthday-fritz-2

54. Garry Kasparov (2017). Don't Fear Intelligent Machines: Work with Them. ICGA Journal, Vol. 39, No. 2
https://www.youtube.com/watch?v=NP8xt8o4_5Q&ab_channel=TED

55. Loviglio, J. (1997). The New York Times.
https://archive.nytimes.com/www.nytimes.com/library/cyber/week/073097chess.html

56. Computer History Museum. (2005).Deep Blue Cartoon | Computer History Museum. https://www.computerhistory.org/chess/stl-431e1a079ea63/

57. Kasparov, G. K., & Greengard, M. (2017). Deep thinking: Where machine intelligence ends and human creativity begins. Public Affairs.

58. IBM. (n.d.). Deep Blue. IBM100 - Deep Blue.
https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

59. ChessBase. (2003B, January 30). Deep Junior Strikes Back. Chess News.
https://en.chessbase.com/post/deep-junior-strikes-back

60. [ChessBase. (2003A, January 29). Kasparov's "Fingerfehler" lets the computer off the hook. Chess News.
https://en.chessbase.com/post/kasparov-s-fingerfehler-lets-the-computer-off-the-hook

61. ChessBase. (2006, December 5). Kramnik vs Deep Fritz: Computer wins match by 4:2. Chess News.
https://en.chessbase.com/post/kramnik-vs-deep-fritz-computer-wins-match-by-4-2

62. Hallsworth, E. (2003). Selective Search 103. The Computer Chess Magazine.

63. Stockfish - chess engines. Chess.com. (n.d.).
https://www.chess.com/terms/stockfish-chess-engine#accomplishments

64. Smatovic et al., S. (2023). Stockfish. Stockfish - Chessprogramming wiki.
https://www.chessprogramming.org/Stockfish

65. Doggers, P. (2018, October 5). AlphaZero chess: Reactions from top GMS, stockfish author. Chess.com.

https://www.chess.com/news/view/alphazero-reactions-from-top-gms-stockfish-author

66. McGrath et al. (2022). Acquisition of chess knowledge in alphazero | PNAS. https://www.pnas.org/doi/10.1073/pnas.2206625119

67. Kasparov, G. (2018). Chess, a Drosophila of reasoning. Retrieved from https://www.science.org/doi/pdf/10.1126/science.aaw2221

68. Somers, J. (2018, December 28). How the Artificial Intelligence Program alphazero mastered its games. The New Yorker. https://www.newyorker.com/science/elements/how-the-artificial-intelligence-program-alphazero-mastered-its-games

69. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T.P., Simonyan, K., & Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. ArXiv, abs/1712.01815.

70. Sadler, M., & Regan, N. (2019). Game Changer: Alphazero's groundbreaking chess strategies and the promise of ai. New in Chess.

71. Stockfish. (2020). Stockfish 12. https://stockfishchess.org/blog/2020/stockfish-12/

72. Leela chess zero. Alexander Thamm GmbH. (2023, January 24). https://www.alexanderthamm.com/en/data-science-glossary/leela-chess-zero/

73. PTI. (2020, May 23). "introduction of computers has changed the approach to chess": V anand. The Indian Express. https://indianexpress.com/article/sports/chess/computers-chess-approach-change-viswanathan-anand-6424099/

74. Wilkenfeld, Y. (2019). Can Chess Survive Artificial Intelligence? The New Atlantis, 58, 37–45. https://www.jstor.org/stable/26609113

75. Radley College. (2020, October 6). Do computers have a positive impact on chess? Inha Choi, remove. issuu. https://issuu.com/radleycollege/docs/junior_project_prize_2020__1_/s/11106301

76. Olson, Dr. R. S. (2014). A data-driven exploration of the evolution of chess: Game lengths and outcomes. Dr. Randal S. Olson.

https://randalolson.com/2014/05/24/a-data-driven-exploration-of-the-evolution-of-chess-match-lengths-and-outcomes/

77. The Guardian. (2020). YouTube. Retrieved July 31, 2023, from https://www.youtube.com/watch?v=2bSyq7Z9ErA&ab_channel=TheGuardian.

78. Cox, D. (2019, May 17). Anish Giri interview: “chess is extremely psychological.” Chess.com. https://www.chess.com/article/view/anish-giri-interview-chess-is-extremely-psychological

79. Johnson, B. (2022, August 30). EP 294- GM Fabiano Caruana on engines, the evolution of chess, the Candidates Tournament, and the World Championship cycle. The Perpetual Chess Podcast. https://www.perpetualchesspod.com/new-blog/2022/8/30/ep-294-ep-294-gm-fabiano-caruana-on-engines-the-evolution-of-chess-the-candidates-tournament-and-the-world-championship-cycle

80. Robinson, J. (2021, December 10). Computers revolutionized chess. Magnus Carlsen wins by being human. The Wall Street Journal. https://www.wsj.com/articles/magnus-carlsen-ian-nepomniachtchi-world-chess-championship-computer-analysis-11639003641

81. Rivlin, M. (n.d.). The Cheater Hunter. https://www.englishchess.org.uk/wp-content/uploads/2019/12/Andy-Howie.pdf . other.

82. Morse, B. (2022a, October 5). How do you even cheat in chess? Artificial Intelligence and Morse Code. CNN. https://edition.cnn.com/2022/09/28/sport/chess-how-to-cheating-explainer-spt-intl/index.html

83. Sergio. (2008, January 5). Ugly story at Vandoeuvre Open. Chessdom. http://www.chessdom.com/news/anna-rudolf

84. C-Squared. (2022). YouTube. Retrieved August 26, 2023, from https://www.youtube.com/watch?v=Uvqr7WmOn6I&t=3017s&ab_channel=C-Squared.