A Survey on Chess Engine Using Deep Learning

Shreya B. Ahire[1], Santosh S. Kale[2], Sumit U. Mali[3]
Department of Computer Engineering, NBNSSOE,
Ambegaon, Pune, Maharashtra, India

Abstract

Chess games have been avantgarde in the history of artificial intelligence. Traditional chess engines such as Stock fish and Komodo have been developed using heuristics with thousands of rules handcrafted by human chess professionals. However, in recent developments, Alpha Zero adopts a completely different strategy and replaces these handmade rules with deep neural networks. In addition, the generic Algorithm used to develop Alpha Zero knows nothing beyond the basic rules of the game, but it achieved superhuman performance in chess, go and shogi games by learning from self-playing games. In addition, you can see how the chess engine evolved over time using different technology and algorithms. In addition, alpha zero is a method of generalizing the algorithm of chess, shogi (shogi), and Go. Starting with random play, I studied how not to be given domain knowledge other than game rules, and Alpha Zero became the best in chess games in a few hours of self-learning. Finally, by observing how the use of deep learning affects the performance of the chess engine and the movement of chess games played on these two engines, Alpha Zero is summarized as compared to other traditional chess engines.

***Keywords**: Chess Engine, Deep Learning, Artificial Intelligence, Alpha Zero.*

I INTRODUCTION

Chess is ranked among the most popular board games and everyone plays everywhere, including city parks, living rooms, schools and well-published formal competitions. This is a two-person strategy board game played by a chess board, a checkered game board with 64 squares placed on 88 grids. In a chess game, one player starts with 16 items: one king, one queen, two rooks, two knights, two bishops, and eight pawns. In computer chess, chess engine means computer program that analyzes the variant position stakes of chess and chess and generates movements and lists that are considered next the strongest millions of chess games. From the beginning of chess to the last tournament of the top chess players, millions of chess games are recorded. Chess engines, on the other hand, have been continuously improved until you can easily beat the world's chess champions. The power of the chess engine is believed to be readily available for a variety of chess games that bring out the attenuation of chess players, data scientists and re-searchers over the past 30 years. For example, professional players use the chess engine on a large base to look for powerful uniqueness. When chess players generally encounter movements played for the interpretation of the chess engine to analyze if they do not miss op opportunities or blunders at some point. The chess engine increases the courage to play every year. At least, increasing the possibility of processing that can be a better range than ever calculation in a certain time. In addition, programming methods have been improved to make the engine more selective and better positioned on the lines it analyzes. Chess engines often use the vast lying "book" previously calculated to fold its playing power for the first few movements. Some chess engines, along with previous calculations and the best movement, hold a record of the chess position in the data database, basically, a kind of chess position that reappears. Since these positions are pre-calculated, the engine simply serves one of the indicated movements in the collected data, saving calculation time and Champion Stock fish and IBM's groundbreaking Deep Blue, which seem to play traditional chess engines including computer chess in the world more powerful and faster It relies handmade thousands

Copyright ⓒ2020 SERSC

of rules and heuristics is active human players who will probably try to write descriptions of past events that will probably happen in the game. However, Alpha Zero takes a completely different approach and replaces this handmade guideline with deep neural networks and general-purpose algorithms that do not understand the game's information beyond the basic rules.

II MOTIVATION

The research on computer chess is as old as computer science itself. Charles Babbage, Alan Turing, Claude Shannon and John Von Neumann devised hardware, algorithms and theories for examining and playing chess games. Subsequent chess turns into a generation of artificial intelligence researchers, a major challenging task for a superhuman-level program of computer chess that plays the best role in high performance. However, while these systems are highly tuned to that area and cannot be generalized to other games without human effort, common gameplay system remains relatively weak. The long-standing ambition of artificial intelligence is to create a program that learns for itself from the first principle instead.

III LITERATURE SURVEY

In computer chess, the chess engine is a computer program that analyzes the position of chess and chess distinct form and creates movements and lists that are considered to be the strongest. The chess engine is usually a backend with a command-line interface without graphics or windows. The engine is typically used in the face of a graphical user with windows such as chess base or Win board, allowing the user to interact via a keyboard, mouse or touch screen. This allows users to play against multiple engines without having to learn each and other new user interfaces.

Nearly 20 years have passed since IBM's Deep Blue Supercomputer first defeated world chess champion Gary Kasparov in standard tournament rules. Since then, the computer that does chess has become much stronger, and the best people leave little chance against the modern chess engine running on the smartphone. However, the computer is faster, but the way the chess engine works has not changed. The process of finding all possible future movements to find the next, depending on the brute force their power tat which is best. Of course, no one can adapt it or you can come anywhere near. Deep Blue was searching for about 200 million positions per second, but Kasparov was probably searching for less than five positions per second. But he played at essentially the same level. Obviously, humans have tricks on their sleeves that the computer has not yet mastered. The trick is to evaluate the chess position and narrow down the most profitable path of search. This dramatically simplifies the calculation task because it prunes all possible moving trees into just a few branches. Matthew Lai created an artificial intelligence machine called Giraffe that evaluated human-like positions and taught them to play chess in a completely different way from traditional chess engines.

Out of the box, the new machine is played at the same level as the best conventional chess engine, many of which have been fine-tuned over the years. On the human level, it is the top 2.2% of tournament chess players, equivalent to FIDE International Master Status. The technology behind Lai's new Machine is neural network. This is a way to process information inspired by the human brain. It consists of multiple layers of nodes connected in a way that changes when the system is trained. This training process uses many examples to recognize the presence of faces in an image to fine-tune the connection so that the network uses specific inputs to produce specific output. In the last few years, neural networks have become very powerful thanks to two advances. The first is to better understand how to tweak these networks, thanks to a much faster computer. The second way is to train your network with a dataset that consists of many names. This allowed computer scientists to train a much larger network organized into many layers. These so-called deep neural networks have become very powerful, and now they routinely outperform humans with pattern recognition tasks such as face recognition and handwriting recognition. So it's not surprising that deep neural networks find chess patterns, and that's exactly the approach Lai took. His network consists of four layers that examine each position on the board together in three different ways. The first step is to examine the global state of the game, such as the number and type of pieces on each side, to see which pieces work or move casting rights. The second is to look at the piece-centric features such as the position of each piece, and the final aspect is to map the squares where each piece attacks and trains his network with carefully generated datasets taken from the actual chess game. This data set is essential for the correct distribution of positions."For example, these positions are virtually non-come out in a real game, so it makes no sense to train the system in a position with three queens per side," he says. You also need to have a lot of unequal positioning diversity beyond what usually happens in top-level chess games. This is because in real chess games, unequal positions rarely occur, but they are always built in searches that the computer performs internally. And this dataset must be huge. A large number of connections in a neural network must be fine-tuned during training, which can only be done with a large number of datasets. Sometimes the dataset is too small to recognize the different patterns that occur in the real world.

Lai randomly selected 5 million positions from the database of computer chess games to generate a dataset. Then he created more diversity by adding random legal moves to each position before using it for training. In total he created 175 million positions in this way. The usual way to train these machines is to manually evaluate all positions and use this information to teach them to recognize the strong and weak ones on the machine. But this is a big job for a position of 175 million people. It can take place by other chess engine, but Lai’s aim was more aspiring. He wanted the machine to learn himself. Instead, he used bootstrap technology to play against himself with the goal of improving the prediction of its own assessment of future positions. This is because there is a fixed reference point (later the game wins, loses, or draws) that determines the value of the position. In this way, the computer learns which positions are strong and which positions are weak.

After training the giraffe, the last step is to test it, and here the results make for interesting reading. Lai tested his machine in a standard database called a Strategic Test Suite consisting of 1,500 positions selected to test the engine's ability to recognize different strategic ideas. "For example, one theme tests the position of open file control, and another is to understand how the values of bishops and knights change relative to each other in different situations. Yet another theme is to test the understanding of the center control," he says. The results of this test are graded out of 15,000. Lai uses this to test the machine at various stages during training. When the bootstrap procedure starts, the giraffe rapidly reaches a score of 6,000 and finally peaks at 9,700 after just 72 hours. Lai says it matches the best chess engine in the world. " All of its evaluation features are carefully hand-designed behemoths with hundreds of parameters, which have been manually and automatically adjusted for several years, many of which are tackled by human’s grandmaster, so it's amazing," he said. He adds. Lai uses the same kind of machine learning approach to determine the probability that a particular move is likely to be worth pursuing. It is important because it prevents unnecessary searches down unprofitable branches of the tree and dramatically improves computational efficiency. Lai says this probabilistic approach predicts the best movement in 46% of the time and puts the best movement in the top three rankings (70%). So, the computer doesn't have to worry about other movements. This is an interesting task that represents a major change in the way the chess engine works. Of course, it's not perfect. One of the disadvantages of giraffes is that neural networks are much slower than other types of data processing.

Lai says giraffes take about 10 times longer than traditional chess engines to find the same number of positions. But even with this drawback, it is competitive. "Giraffe can play at the level of FIDE International Master on modern mainstream PCs," says Lai. In contrast, the top engine is played at the Super Grand Master level. It is still impressive. "Unlike most chess engines that exist today, giraffes derive their playing power from an intuitive and complex positioning concept because they can accurately evaluate tricky positions rather than being able to see very far. He's a human being, but for some long-time chess engines are elusive," Lai said. "This is especially important in the opening and end game phases to play very well. And this is just the beginning. Lai says it should be easy to apply the same approach to other games. Prominent is the traditional Chinese game of Go, where humans still have an impressive advantage over silicon competitors. Perhaps Lai can then have that crack.

A. *Fundamentals*

i. *Reinforcement Learning*

Reinforcement learning is a goal-oriented algorithm that learns how to achieve complex goals (goals) or maximize them in a specific dimension. For example, maximize the points you earn in your game with a lot of movement. They start with a blank slate and can achieve superhuman performance under the right conditions. Like some children's incentives by spanking and candy, these algorithms are punished when making the wrong decisions and rewarded when making the right one.

ii. *Neural Networks*

Neural networks are a series of algorithms modeled on the human brain and are designed to recognize patterns. They interpret sensory data through a kind of mechanical perception, labeling, or clustered raw input. The pattern they recognize is a number contained in a vector and must translate all real-world data, including images, sounds, text, and time series.

Neural networks help us cluster and classify. You can think of these layers as clustering and classification layers on top of the data you store and manage. Group unlabeled data ac-cording into similarities in the example inputs and classify the data if you have a labeled dataset that you want to train. (Since neural networks can also extract functions provided to other algorithms for clustering and classification, deep neural networks It can be thought of as a larger machine learning application that includes algorithms for reinforcement learning. Indicates classification and regression.

B. *Related Works*

The techniques used in traditional chess engines are as follows

i. *MIN MAX*

Minmax is a decision rule used in decision and game theory to minimize losses that can occur in worst-case scenarios.

Example 1: In the middle of a game, there are two possible movements. Let's also assume that there is some way to assess how favorable the game's condition is for you at any given time. Such algorithms take the state of the game and output numbers from 1 to 10 (10 means you win, 1 means your opponent victory).

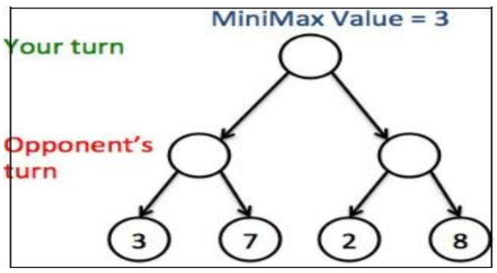

Fig..1 *The game tree for Min Max*

In such a tree, minmax has a value of 3. Given that your opponent plays perfectly, it's the best way you can. You'd probably want to reach the node at 8, but if you move to the next subtree, your opponent chooses to go left (because he wants to keep that number as low as possible), and you end up with 2.

ii. ALPHA-BETAPRUNING

Alpha-beta is an algorithm that attempts to reduce the number of nodes evaluated when the Minmax algorithm is run in the search tree. For example, in the previous example, there were four leaf nodes that were evaluated. However, you do not actually need to evaluate the leaf node to the right. As soon as the value of the third node is observed to be 2, it is better to move to the left, so the last node can be ignored.

Summary of a survey

After going through many papers and articles on chess engines, you can conclude that most of the existing chess engines work with a naive approach. Most of them use brute force instead of intelligent predictions like Alpha Zero.

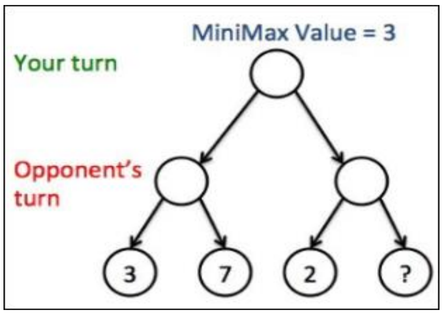

Fig. 2 The game tree for Alpha beta pruning

C. Methodology

i. Alpha Zero

In about 1047 different chess positions, all available movements, and the computational cost is too high to thoroughly search for any movement that may follow the game. Therefore, most chess engines, including Alpha Zero combine, search algorithm with an evaluation function that provides an estimate of the good way of any position points in the game. The traditional chess engine uses a variant of what is called Alpha Beta Tree Search enhanced by dozens of game-specific search heuristics and combines this with an evaluation feature designed by an expert chess player. In contrast, alpha zero uses a powerful alternative to alpha beta tree search, which instead develops its own evaluation function and has the added advantage that it can be previously taken into account using Monte Carlo Tree Search (MCTS), You will learn completely alone and knowledge of which movements are promising and which movements are not promising. This allows the search to focus primarily on promising and related variations. Furthermore, MCTS is robust against inaccuracies of the evaluation function, which is averaged over many different positions. So where does the prior knowledge come from? The neural network takes the position of the current game as input, returns the probability of movement for each possible movement, the strongest movement (this is the time called the policy network), and an estimate of the value of the current position (at some time is a value network). This output leads the Monte Carlo tree toward stand-in the most promising segment of the game tree. By reducing the number of moves considered at each position, the probability of movement reduces the width of the search. Thus, it is possible to estimate the value of the non-terminal position, it is possible to determine the value of the result of a predetermined variation even before the end of the game, to reduce the depth of the required search in the tree. Importantly, the same algorithm can reach superhuman abilities in multiple games without adapting the architecture for each particular game. In other words, the system displays the degree of generality: the same process of Monte Carlo Tree search led by neural networks is trained in self-renewal reinforcement learning and proves to be effective across several domains without the need.

ii. Monte Carlo Tree Search

a. What is Monte Carlo tree search?

MCTS is an algorithm that updates nodes in the tree to find the final solution and selects an Expand simulation to understand the best move from a series of moves. This method is repeated until you reach the solution and learn the resulting game policy.

b. How does the Monte Carlo Tree search work?

Select: This process is used to select the nodes on the tree that have the greater chance of winning. For example, given the chance of winning after the first move 4/6, node 2/3 has the highest chance of winning, given the 2/3,0/1, 1/2 movement. The selected node is searched from the current state of the tree, and the selected node is at the end of the branch. The selected node is most likely to get its path, so it is more likely to reach the solution faster than another path in the tree.

Expand: After selecting the right node. Expansion is used to increase. Expanding the selected node and creating a large number of child nodes increases the options in the game. In this case, there is only one child node. These child nodes are future movements that can be played in the game. For the time being, nodes that have not been expanded further are leaves.

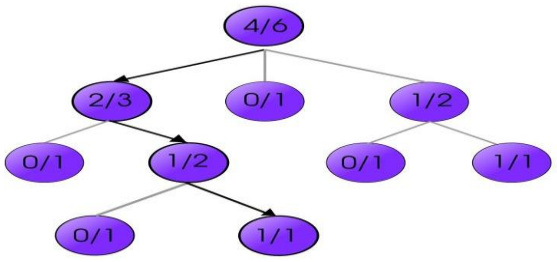

Fig. 3 Selection of node

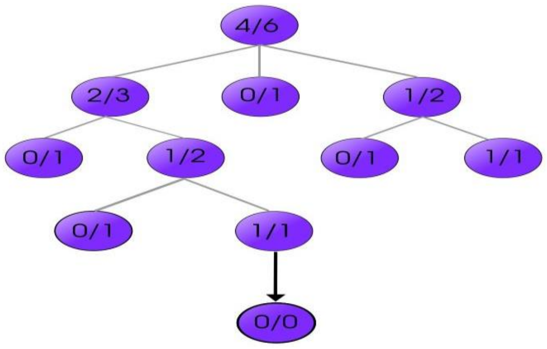

Fig 4 Expansion of tree

Simulation: Explore because no one knows/the best child of any node They leave a group Move to the best form for each form and lead to the correct answer under the tree.

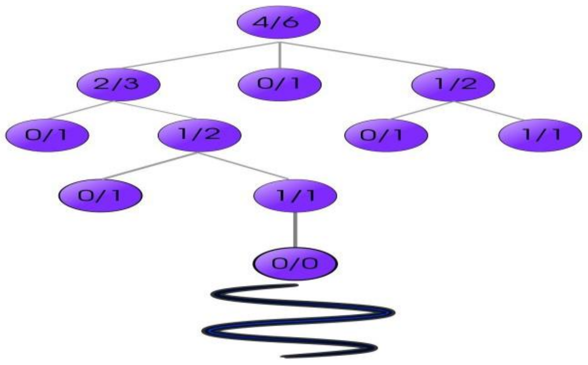

Fig.5 Simulation of tree

c. How to find the best children who lead us to the right solution?

Reinforcement learning is used to make even more random decisions in the game. Down from all child nodes. The reward is then given to all child nodes by calculating how close the random decision output from the final output required to win the game.

iii. How Alpha Zero Uses Deep Learning

i. Domain Knowledge

Alpha Zero provided the following domain knowledge about each game:

1. The location that describes the move and the input features that describe the output features are configured as a set of planes. In other words, the neural network architecture matches the grid structure of the board.
2. Alpha Zero is provided with the perfect knowledge of game rules. They are used during MCTS to simulate the results of a series of moves, determine the end of the game, and score simulations that have reached the end state.
3. Knowledge of the rules is also used to encode the input surface (e.g., cast ring, repetition, no progress) and output surface (piece movement, promotion, shogi fragment).
4. The scale of the search noise uses a large number of common legal movements.
5. The chess and shogi game are over and the result of the draw has been assigned. More than 722 steps of Go games ended and scored under The Rules of Tromp Taylor, as in previous works. Alpha Zero did not use opening books, end-game table-based, or domain-specific heuristics.

ii. Neural Network Training

To learn each game, untrained neural networks play millions of games against themselves through a trial-and-error process called reinforcement learning. At first it plays completely randomly, but over time, the system is more likely to win, learn from loss, pull out to adjust neural network parameters and choose a favorable move in the future. The amount of training required by the network depends on the style and complexity of the game, with 9 hours of chess, 12 hours of shogi, 13 days of Go. Some computers play games that simulate results while playing (for example, jellyfish playing Backgammon). Alpha Zero simulates millions of games during training but does not use technology to simulate until the end of the game (random rollout) during play. Once Alpha Zero training is complete, the latest neural nets in the policy (how to select movement) and evaluation (how to evaluate positions) are used in match play. Previous versions of Alpha-Go of Deep Mind were used to make random rollouts while playing. Alpha zero does not need this because its value network is already advanced and no additional rollouts add values during play. However, as a result, there is no randomness built into alpha zero. The trained network guides the search algorithm known as Monte Carlo Tree Search (MCTS) and is used to select the most promising movements in the game. For each movement, alpha zero is searched for only a small part of the position taken into account by the traditional chess engine. For example, chess only searches for 60,000 positions. The position of chess every second for about 60 million positions in stock fish.

a. How to program a chess neural network?

If you are configured to use NN, it is relatively easy to express. A typical NN is just a graph, and each node is a neuron. Each neuron has a current activation value, the transition expression for calculating the next activation value, the input value, i.e., calculates the activation value of all nodes with a link to it. More classical NN, the input layer, the output layer, the same neurons of each layer, and does not have a time-dependent, therefore, the array of input nodes, the sequence of the output nodes, and can be represented by the link graph of the nodes connecting them. Each node has the current activation value and a list of nodes to which the node is transferred. The power to calculate the output value simply sets the activation of the input neuron to the input values, iterates through each subsequent layer in turn, and uses a transition expression to calculate the activation value from the previous layer. When you reach the last (output) layer, you get results.

V CONCLUSION

The development of alpha zero is sure to cause a lot of reverberation.

AI community. no one expected it to beat very the best chess and Go engine without any domain knowledge using such an elegant, simple general-purpose algorithm. And no one thought it would have happened soon. Although it is still in its infancy, AlphaGo Zero is an important step toward stake in this goal. If similar technology can be applied to other structured problems, such as protein folding, reducing energy consumption, and exploring innovative new materials, the breakthrough can have a positive impact on society. In the world, first of all, the machine does chess by evaluating the board apart from using brute force and work out on any possible movement.

VI FUTURE SCOPE

As computer chess programs routinely defeat the Grand Masters, we entered a new era in decades, such as chess playing computers, and concluded the era. From now on, there will be many Developments including using the computer systems to enhance human play rather than throwing humans at machines. Other developments online show this same trend computer chess program makes it available for free download on handhelds and laptops, amateur and professional chess players improve chess games, learn, it is used on a daily basis to redisplay. Computer chess seems to have taken on the role of an expert helper to all chess players now after demonstrating its superiority to human players.

Alpha Zero is more than chess, shogi, or go. To build an intelligent system that can solve a wide range of real-world problems, you need the flexibility to adapt to new situations. There have been some advances toward this goal, but a system that can master certain skills at a very high-level replicates key challenges in AI research, but often fails when presenting slightly changed tasks. Alpha Zero's ability to master three different complex games and potentially any perfect information game is an important step to overcome this problem. This shows that a single algorithm learns how to discover new knowledge in different settings.

REFERENCES

[1] Demis Hassabis, David Silver, "AlphaZero: Shedding new light on the grand games of chess, Shogi and Go", Deep Mind, 2019

[2] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis,"A general reinforcement learning algorithm that masters chess, shogi and Go through self-play".,2018

[3] hu-Hsuan Hsueh and I-Chen Wu, Jr-Chang Chen, Tsan-sheng Hsu "AlphaZero for a Non-Determinis

[4] .JI. S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[5] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems* 2017, 4-9 *December* 2017, *Long Beach*, CA, USA, pages 5366–5376, 2017.

[6] Tord Romstad, Marco Costalba, Joona Kiiski, et al. Stockfish: A strong open source chess engine. *https://stockfishchess.org/.* Retrieved November 29th, 2017.

[7] Computer Shogi Association. Results of the 27th world computer shogi championship. *http://www2.computer-shogi.org/wcsc27/index_e.html.* Retrieved November 29th, 2017.

[8] Matthew Lai. Giraffe: Using deep reinforcement learning to play chess. Master’s thesis, Imperial College London, 2015.

[9] Stock_shchess.org. Home - stock -open source chess engine, 2015. URL http://stockfishchess.org.

ISSN: 2233-7857 IJFGCN
Copyright ©2020 SERSC