Exploring AlphaZero in Chess

Sam Johnson

***Key Words*—Reinforcement learning, neural networks, chess, chess engines, Stockfish, AlphaZero, Monte Carlo Tree Search, evaluation**

I. Abstract

In the chess community there is a general desire for constantly improving chess engines. These engines help top chess players improve their skills and train for top tournaments. They are considered by many to be the ultimate chess teacher, as they have the ability to see things a human cannot. There have been many advancements over the years with the impressive culmination of DeepMind's AlphaZero. AlphaZero is a general framework that has learned to play chess as well as other games with an impressive level of performance. The DeepMind paper describes how AlphaZero was able to soundly beat Stockfish 8 [10].

This paper takes the open-source version of AlphaZero, Leela Zero, and pits it against the more modern Stockfish 15 and other top chess engines. Ultimately, it does better than average, prevailing against some engines, tying others, and losing to Stockfish 15. We also visualize chess engine evaluations over time and Leela Zero's behavior to get a better idea of how this reinforcement learning algorithm does so well.

II. Introduction to AlphaZero

A. Performance against Stockfish 8

AlphaZero was introduced by DeepMind in 2017 [10]. It shocked the chess world by defeating Stockfish 8, which was the greatest chess engine at the time. We display the results of AlphaZero's 100 games against Stockfish 8 in figure 1. We can see that out of 100 games played, AlphaZero won 28 games, drew 72 games, and lost 0 games. This is an impressive performance.

Game	White	Black	Win	Draw	Loss
Chess	*AlphaZero*	*Stockfish*	25	25	0
	Stockfish	*AlphaZero*	3	47	0
Shogi	*AlphaZero*	*Elmo*	43	2	5
	Elmo	*AlphaZero*	47	0	3
Go	*AlphaZero*	*AG0 3-day*	31	–	19
	AG0 3-day	*AlphaZero*	29	–	21

Fig. 1: AlphaZero vs Stockfish 8

B. How Traditional Chess Engines Work

Before AlphaZero came into the picture, chess engines had already made substantial progress over decades of work. They generally had handcrafted evaluation functions that they used to determine the value of a given position. These functions were created by grandmasters that intuitively knew which player was better off in a given position. Some heuristics that were commonly used included material advantage, having a bishop pair, pawn structure, king safety, outposts, trapped pieces, how active the pieces were, etc.

After that, a mini-max search was performed to look down a series of moves until a leaf node was reached. Alpha-beta pruning was used to reduce the number of variations that had to be looked at.

Additionally, a carefully tuned opening book was used to quickly decide which moves to make at the beginning of the game. No computation or mini-max searching was required here.

Similarly, an endgame tablebase, which contained stored endgame positions, instructed the engine on how to finish a game where fewer than six pieces remained.

Chess engines have changed somewhat with the introduction of AlphaZero but many chess engines still rely to some extent on handcrafted evaluation functions, opening books, and endgame tablebases.

978-1-4673-8988-4/17/$31.00 ©2017 IEEE

C. How AlphaZero Works

AlphaZero implements none of the ideas taken from traditional chess engines. There is no handcrafted evaluation function. Similarly, neither opening books nor endgame tablebases are used. Instead, AlphaZero uses a pure self-learning approach through reinforcement learning.

AlphaZero has a deep neural network with the board position as input and outputs a vector of move probabilities. It also outputs a scalar value representing the expected value coming from that position.

AlphaZero also uses a Monte Carlo Tree Search to self-play games. It chooses moves by selecting states with a low visit count, high move probability, and high value.

AlphaZero learns over time by adjusting its neural network parameters through gradient descent. Two different loss functions are used: mean-squared-error and cross entropy loss. They are given below.

$$l = (z - v)^2 - \pi^\top \log \mathbf{p} + c\|\theta\|^2$$

$$(\mathbf{p}, v) = f_\theta(s)$$

D. The Impact of AlphaZero

AlphaZero had a huge impact on the chess engine community. After people realized that a neural network - reinforcement learning based approach to chess engines was better, many existing engines modified their frameworks to incorporate these new ideas. Stockfish 15, for example, incorporates a neural network [3].

With older engines making changes and catching up to AlphaZero (Leela Zero), it's unclear if Leela Zero can hold onto the title of *Best Engine*.

E. Leela Zero

The network weights obtained by the trained version of AlphaZero weren't publicly made available, so an open source project known as Leela Zero was created to exactly implement the AlphaZero project. Leela Zero relies on volunteers to donate CPU time in order to train its network. The network has now played approximately 980 million games of chess and has reached a very high level of performance [8]. Leela Zero now has a web page with useful information and an easily downloadable program. Leela Zero, which is a clone of AlphaZero, is the actual chess engine we will be pitting against rival chess engines.

III. Competitor Chess Engines

I looked at a couple of web pages [1] [6] to find other top chess engines. Some of the engines on the list weren't free to download and I ignored them. I ended up with the following engines, each of which were freely available to download:

- Komodo
- Shredder
- Stockfish
- Rybka
- Xiphos

Here is a brief description of each engine:

Komodo is a chess engine that has done well, even winning the World Computer Chess Championship in 2019. The latest version of Komodo is Dragon 3. For this study, I used Komodo 13, which is an older version. [2] As a result, this study may be slightly unfair to Komodo, as an older version of its chess engine is being used. In any case, it is a very powerful engine.

Shredder was developed by Stefan Meyer-Kahlen. It is one of the most successful chess programs in history, having won 19 titles as World Computer Chess Champion since 1996. [9]

Stockfish is an engine that has been around for a while but continues to improve with each new version. Originally, it had a handcrafted evaluation function. The latest version (Stockfish 15) uses a neural network to evaluate different positions. [3] As a result, it is much stronger than Stockfish 8 from several years ago.

Rybka was developed by IM Vasik Rajlich in the early 2000s. It won the World Computer Chess Championship every year from 2007-2010. It was

also involved in a controversy involving plagiarized code. [4]

Xiphos was created by Milos Tatarevic in 2021 using a handcrafted evaluation function. It was inspired by Gary Kasparov's book "Deep Thinking".

Each of these engines will have to face off against Leela Zero to see whether reinforcement learning brings any advantages to chess performance.

IV. Initial Game Setup

After running a few quick experimental chess games between Leela Zero and Stockfish, I quickly found that the games were more or less deterministic, with the same moves being played every game. This occurs because the network is fixed and the engines always output the same evaluations for the same position.

This wasn't very exciting, so I decided to start the games several moves into the position using a PolyGlot opening book. A PolyGlot opening book is created by analyzing thousands of games and then recording the frequency of each move after each opening position. Then, a move can be randomly selected based on the probability distribution of moves after that position. Figure 2 shows an example of the PolyGlot from the initial position in the game.

Move	Weight
1. e4	40%
1. d4	30%
1. c4	10%
1. Nf3	10%
1. b4	5%
1. g4	5%

Fig. 2: PolyGlot Example

Using the method described above, we make five weighted random moves for each player to arrive at a non-deterministic position to start the game. Under this method, it is highly unlikely to generate the same position twice.

It is possible that one player could have a slight advantage resulting from the randomly determined starting position. To ensure fairness, both engines get to play once as white and once as black after the selection of a given position.

In figures 3 and 4 we show Xiphos vs Leela Zero and Leela Zero vs Stockfish respectively. As you can see, both starting positions are very different and will result in very different games.

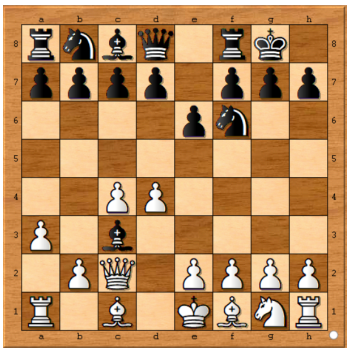

Fig. 3: Xiphos - Leela Zero Start Position

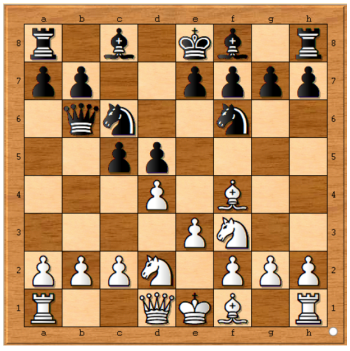

Fig. 4: Leela Zero - Stockfish Start Position

V. Rules

A. Time Constraints

Due to limited computational resources, we opt for rapid chess games. An engine can use up to 10 seconds for each move. Time cannot be carried over between moves.

In general, it only takes a couple of seconds for an engine to find a reasonably good move. Each additional second of computational time is worth less than the previous in terms of improved evaluation.

As you would expect, however, more time leads to better moves.

B. Other Rules

The standard rules of chess apply. A win is worth one point, a draw is worth half a point, and a loss is worth zero points.

The standard rules of chess apply. Human players traditionally resign if they find themselves in what they deem to be an unwinnable position. Similarly, if they find themselves in a position where improvement isn't possible, they will likely agree to a draw.

The difference in these games is that the engines don't communicate with each other. They continue playing until the game has terminated with a win, a loss, or a draw. Thus, engine games tend to be much longer than human games, sometimes reaching 100 moves or more.

VI. Implementation

Each of the described engines was located and downloaded from its respective websites. The engines were then loaded into Python using the python-chess library. [5] The python-chess library was then used to create the chess board, evaluate each position for 10 seconds, and update the board. The matplotlib plotting library was used to create many of the visualizations in this paper. [7]

VII. The Results

We have Leela Zero play against other engines several times using a Dell Inspiron 5584 laptop running on an Intel i5 8th generation processor with no GPU use.

In figure 5 we display a stacked bar chart showing the points scored by Leela Zero, playing as white against each of its opponents.

We can see that Leela Zero tied with Komodo and Xiphos. Leela Zero completely dominated both Shredder and Rybka, with them unable to score even a draw. Stockfish, however, crushed Leela Zero, with Leela scoring just 0.5 points.

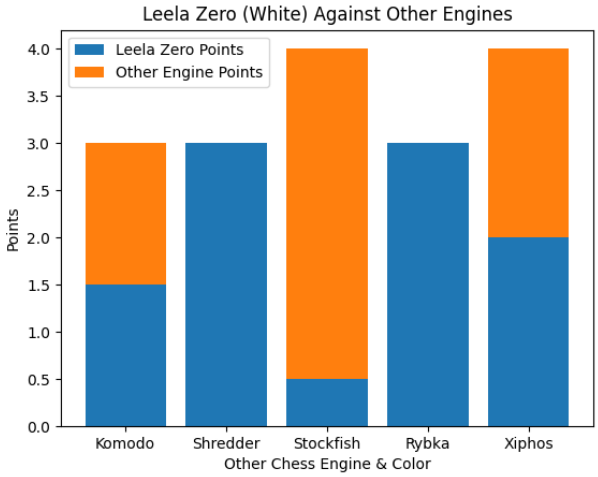

Fig. 5: Leela Zero as White

Figure 6 displays Leela Zero playing with the black pieces. We can see that Leela once again tied with both Komodo and Xiphos. Leela dominated Shredder and Rybka, giving up only half a point. Stockfish once again crushed Leela.

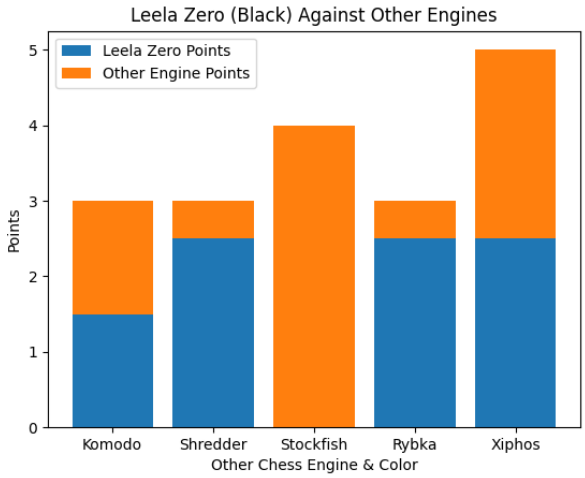

Fig. 6: Leela Zero as black

Finally, I was curious as to whether Leela Zero would exhibit more dominance in lengthier games. I pitted Leela Zero against Komodo in two 60 second per move games initialized from random positions as previously described. Leela lost one game and one game ended in a draw, which is more or less consistent with earlier results. I didn't

run any more lengthier games due to the fact that each game took about two hours to run.

Overall, Leela Zero performed decent with both the white and black pieces. It crushed Shredder and Rybka, tied Komodo and Xiphos, and was defeated by Stockfish. Leela Zero, however, can no longer be considered the best engine. That honor now goes to Stockfish 15.

It is worth noting that Leela Zero, with its complex neural network, was likely created to run on a GPU. In our experiments, however, it only ran on a CPU. This could be an area of future exploration.

VIII. How Leela Zero Thinks

Recall that for each move chess engines evaluate the strength of their position in pawns. Thus, 0 would mean an equal position, 0.5 would mean a slight advantage, and 3+ would mean a large advantage.

In this section we show the evaluation plots from both engine's perspectives for the first 10 games that were run. There are 2 games against each of Leela Zero's competitors, one with Leela as white and one with Leela as black.

In figure 7 we display the evaluation plot of the game between Leela Zero as white and Komodo as black. The white line is white's evaluation from its point of view while the black line is black's evaluation from its point of view. Thus, a positive white value would mean that white *thinks* it's winning while a positive black value would mean that black *thinks* it's winning. Conversely, a negative white value would mean white *thinks* it's losing and a negative black value would mean black *thinks* it's losing.

Leela Zero recognizes that it has taken a massive five-point lead by move 25. Komodo, however, thinks it's only one point behind. As more moves are played, Leela Zero's lead only grows, with Komodo eventually realizing that the game is lost. The game ends with Leela Zero checkmating Komodo.

Note that we limit the peak evaluation advantage to 10 for plotting purposes. The real evaluations outputted by the chess engine can reach 100, 200, or even infinity when one engine sees a path to checkmate. In practice, once one engine has reached an evaluation of at least 10, it isn't possible for the other engine to recover.

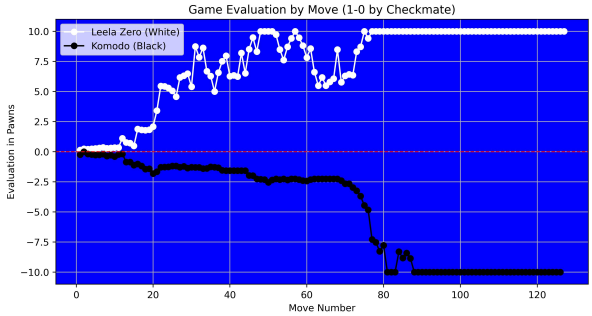

Fig. 7: Leela Zero (White) vs Komodo (Black)

In figure 8 we can see that Komodo (white) starts with a small advantage, as evaluated by both players. Leela Zero (black) works hard to mitigate this advantage. By move 35 or so, the position has equalized. The game later ends in a draw from five-fold repetition. This wasn't a bad result from Leela Zero, given that black has a slight disadvantage from moving second.

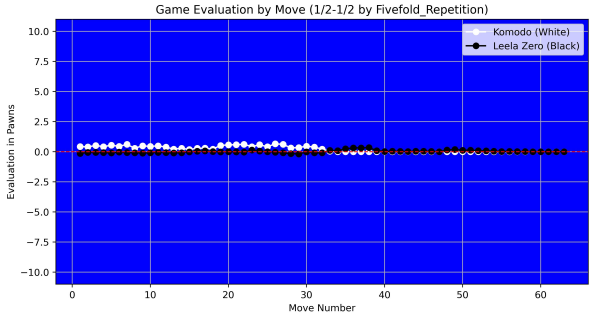

Fig. 8: Komodo (White) vs Leela Zero (Black)

In figure 9 Leela Zero (white) gains a quick advantage against Shredder (black) and proceeds to win via checkmate.

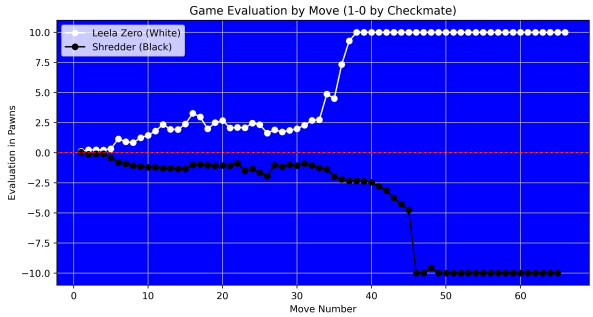

Fig. 9: Leela Zero (White) vs Shredder (Black)

In figure 10, Leela Zero demolishes Shredder again, this time with the black pieces!

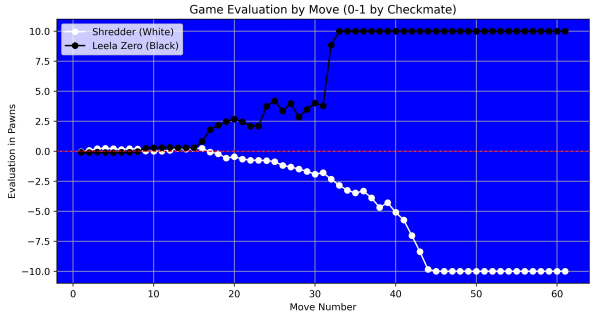

Fig. 10: Shredder (White) vs Leela Zero (Black)

In figure 11 Leela Zero (white) holds on for dear life until move 45, when Stockfish (black) begins winning, eventually crushing Leela Zero.

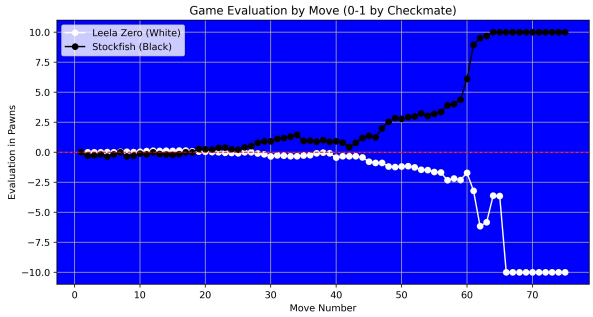

Fig. 11: Leela Zero (White) vs Stockfish (Black)

In figure 12 Stockfish (white) quickly checkmates Leela Zero (black).

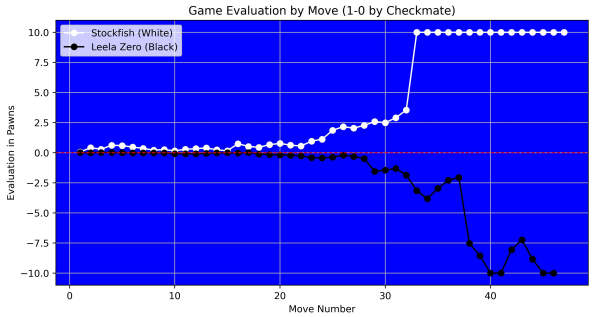

Fig. 12: Stockfish (White) vs Leela Zero (Black)

Figure 13 against Rybka is rather erratic with a lot of oscillation in the evaluations by both players. Leela Zero eventually takes the win.

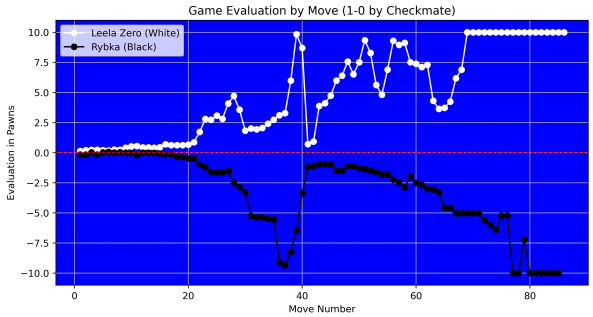

Fig. 13: Leela Zero (White) vs Rybka (Black)

Leela Zero (black) once again faces off against Rybka (white) and holds a draw in figure 14.

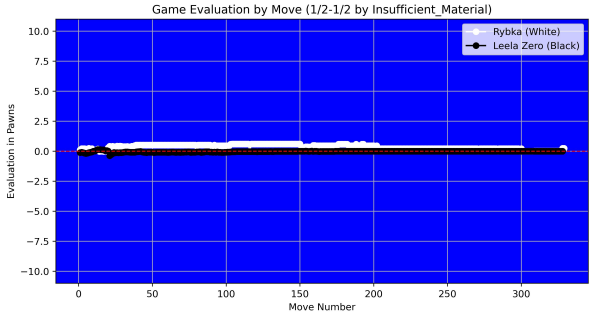

Fig. 14: Rybka (White) vs Leela Zero (Black)

Leela Zero (white) and Xiphos (black) remain locked in combat until they draw by stalemate in

figure 15.

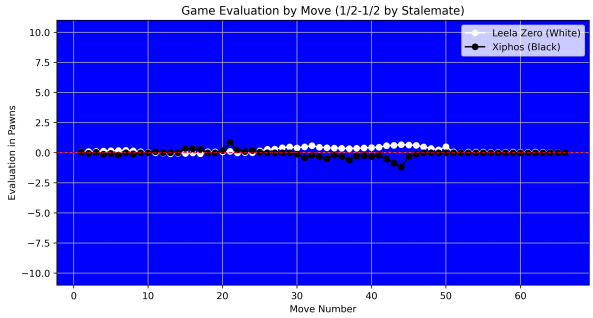

Fig. 15: Leela Zero (White) vs Xiphos (Black)

Xiphos (white) and Leela Zero (black) have a rematch in figure 16. Leela Zero has a nice lead by move 35 and finishes things off from there.

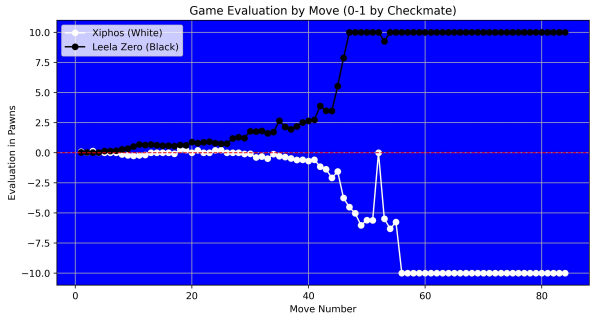

Fig. 16: Xiphos (White) vs Leela Zero (Black)

IX. Conclusion

In this project we began by discussing how AlphaZero works and how it revolutionized modern chess engines that traditionally relied on handcrafted evaluation functions.

Then we pitted Leela Zero, the open source version of Alpha Zero, against other strong chess engines of today. We also analyzed its evaluations over time. In the end, it performed decently, thoroughly beating some engines, tying others, and being defeated by Stockfish 15. We then analyzed the game evaluations to see how Leela can sometimes see things that other engines can't.

As previously noted, Leela, with its convolutional layers, was likely created to run on a GPU. Future work could include running Leela on such a platform and/or giving each engine more time for analysis. Perhaps Leela could be more dominant in such conditions.

References

[1] Chess engine: Top 10 engines in the world.

[2] Dragon by komodo chess - world champion chess engine.

[3] Official-stockfish/stockfish: Uci chess engine.

[4] Rybka - chess engines.

[5] Niklas Fiekas. Chess: A chess library for python.

[6] Andrew Hercules. Top 10 strongest chess engines in 2022, Aug 2022.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[8] Gary Linscott. Progress, 2022.

[9] Stefan Meyer-Kahlen. About shredder chess, 1997.

[10] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.