**Advanced Narrow AI in Chess**

**Overview, Performance and Shortcomings**

In this essay, I would like to illustrate the difference in evaluations of tree-search engines (such as Stockfish) and neural networks (such as AlphaZero) and how the latter tend to outperform the former.

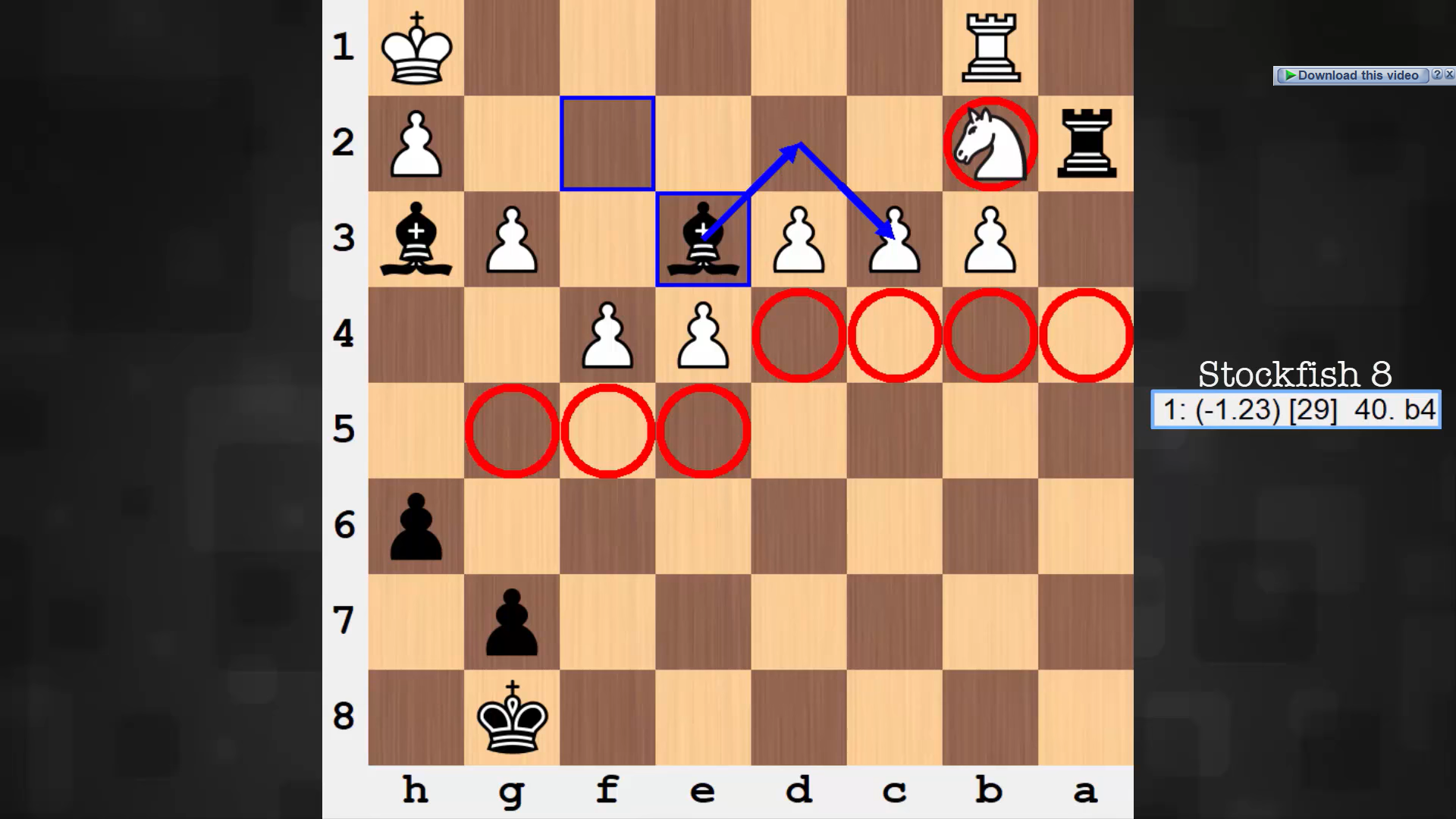
First, how humans play chess should be discussed first. These are the five steps of learning chess by humans:  
1) Learning the rudimentary knowledge  
2) Learning the openings  
3) Learning the tactics: In the middle game, there are no longer any fixed routes or methods. The positions totally depend on the knowledge of tactics and mastery by players.  
4) Learning long-term positional advantage: this is extremely crucial because if two sides manage to reach the end game, those who have better positional advantage will almost surely win the game.  
5) Repeating this again and again until players will have a sense of “chess” and will improve their decisions on making a move by their large inventories of experience in their head. This one is applied for neural network chess.

For most modern engines, such as Stockfish, Komodo, Fritz and Houdini, step 1 and step 2 have been well encoded into their database. That is why sometimes in TCEC (Top Chess Engine Championship), engines are not allowed to use opening tables to determine which one’s algorithm is genuinely more robust in a match.

The main difference between tree-search engines and neural networks is the step 5. No matter how much Stockfish plays, it will never become smarter. However, for Leela and AlphaZero, each match will bolster their knowledge of chess. In 100 matches settings, without any use of opening data and libraries, reinforcement learning is much more likely to win. To demonstrate this, this is a remarkable example:

AlphaZero (Black) vs Stockfish (White)

In this game, Stockfish decides to offer a sacrifice that aims to weaken the queen’s side structure of Alpha Zero on move 13th. Alpha Zero accepts the sacrifice, making Stockfish exchange 2 pawns for a minor piece. In the progress, Stockfish has gained many pawns and has earned more than 2 points in materials.



However, AlphaZero has carefully crafted a strategy of attacking in the king’s sides, steering the game towards the position above. Although Stockfish has so many pawns, they do not contribute anything to the current battle in the White’s King side. At the end of the game, White (Stockfish) has to sacrifice a minor piece and lose in lesser materials, despite having more materials in this position.

It turns out that the sacrifice Stockfish made earlier was inaccurate in the long run. AlphaZero’s strategies are subtle, yet highly effective:

+ Consolidating the position of its own, making the king as safe as possible  
+ Limiting the range of the opponent’s pieces. In particular, AlphaZero loves to hinder the movement of the light square bishop of the opponent.

+ Made a long-term sacrifice halfway through the game, if possible, in exchange for better position.

With these steps, AlphaZero has far excelled at step 4 above than any traditional engines.

There are many factors behind the excellent performance of neural networks. In traditional engines, they utilize two functions: one to evaluate and one based on the evaluation to search for optimal moves. However, the approach above is limited by computational capacity and by the accuracy of the evaluation function, given the fact that chess is still a finite game, despite its immense number of possible games (Claude Shannon estimate there are 10^120 possible chess games within 40 moves). [1]

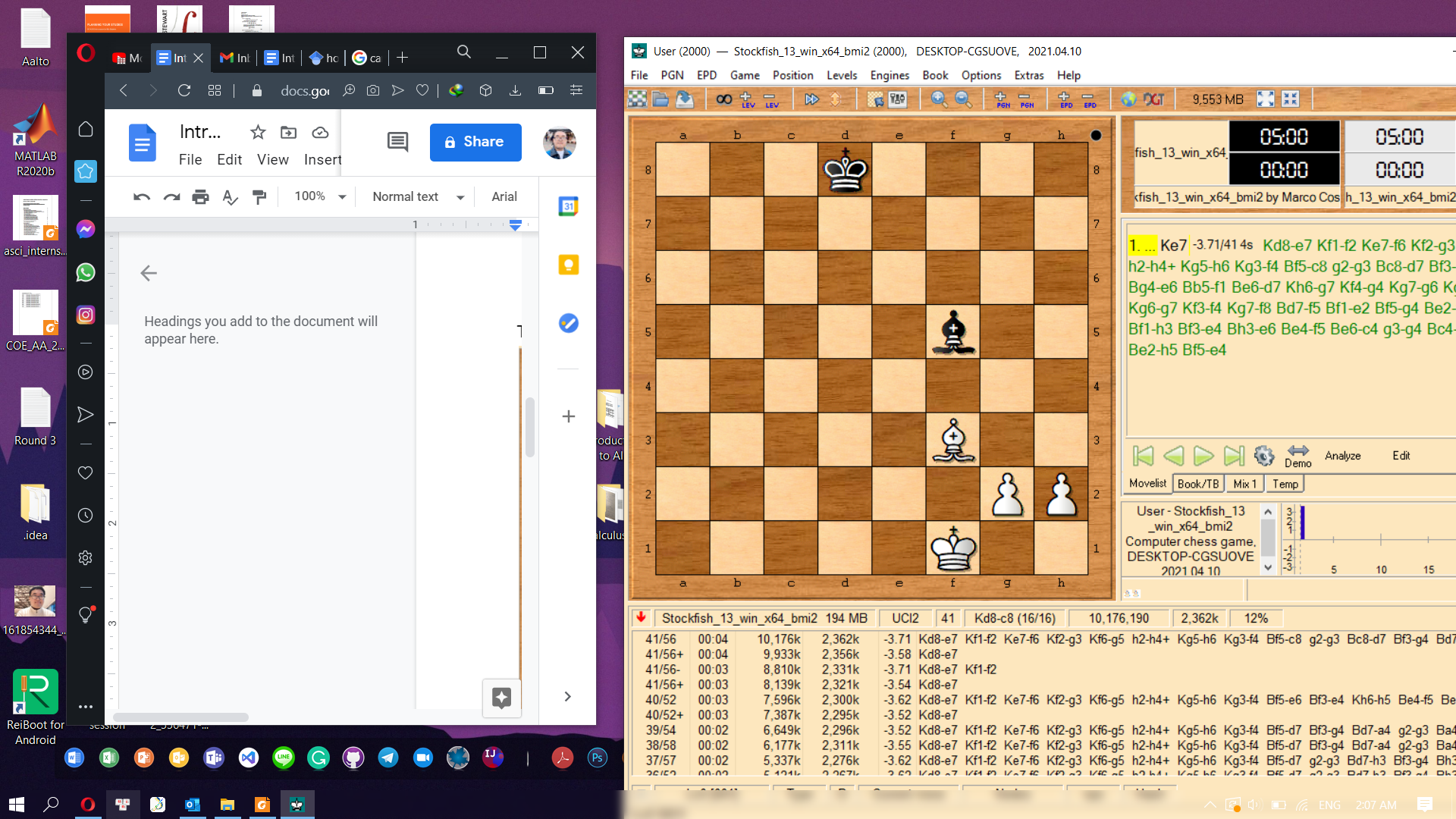
Convolutional neural network has chosen another more effective path: it searches through the sample space with much fewer nodes and applies chess principles (an example will be illustrated below) into its evaluation.

Principles in neural networks indicate that a disadvantage will be amplified and leading to a loss and vice versa, an advantage will be amplified cultivating in a victory. Neural networks thus assign 1 for a winning position and 0 for a losing position [1]. When an opponent does not play the very best move each time, the neural networks will capitalize on this small inaccuracy and lead the game towards a disadvantageous position for its opponent.

Particularly, neural networks utilized an alpha–beta search (adversarial) of the current game tree for each position that seeks for a selected number of moves [2]. Apart from the opening and the end game, the minimax move is determined so that it forces the opponents to deal the least damage. The search’s depth consists of a few plys (two ply corresponds to two moves, one made by white, one made by black) to shorten the time of evolutionary computation [2]

For a long time, chess engines have been dominating the game and humans are no longer their match, since computers can calculate up to one hundred million to one billion variations each move. However, traditional tree-search chess engines such as Stockfish are not perfect, since they do not understand the intuitions of chess like humans: they are like the Chinese room. They respond well to each variation made, but they do not intrinsically understand chess.

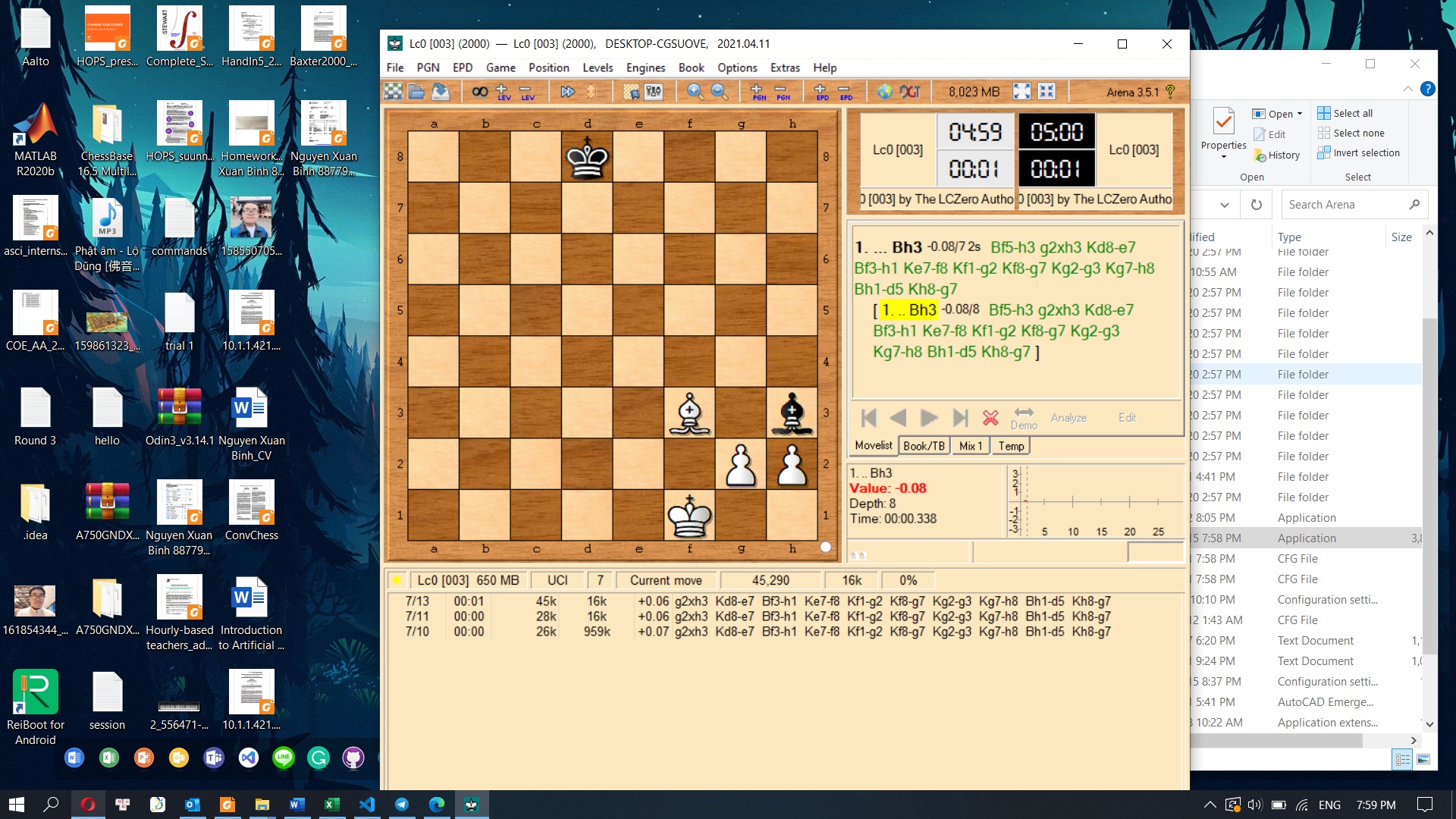
This is a position that tells apart the “intelligence” between tree-search and neural networks: the famous drawn position by Black.



At first it seems that Black is losing since White has two pass pawns. Indeed, that is what the chess engine (Stockfish) currently thinking. But it turns out that Stockfish doesn't visualize the rule that if the color of the bishop doesn’t match the color of the promotion square in the corner, it will lead to a draw (tie).

The solution is Black Bishop to square h3 (Stockfish thinks moving the King to square e7; it does not find out Bishop to h3 as the crucial next move). Black sacrifices the bishop. If white does not take it, black bishop will take the pawn on g2 the next move and the game drawn due to the rule above (noted that the g7 pawn cannot avoid being captured since white king is pinned by black bishop). If White takes the black bishop with the pawn, the game is drawn as well due to the same rule.

Now let us try the same position with Leela Zero, an open-source neural network chess. Surprisingly, Leela Zero found out the correct move within a second, which is Bishop to h3 according to the analysis above. By incorporating chess principles into the reinforcement learning, neural networks can find out the correct move that traditional engines cannot.



Besides the difference above, tree-search engines are noted to struggle with closed positions, and it is relatively weak in understanding fortress tactics. The reason behind this is that a fortress requires an accumulation of moves to build a fortress impenetrable by the other side, which leads to a draw by the team who is building the fortress. Engines do not grasp the concept of fortress well, so in many situations where it could have been drawn, the engine determines that a side is lost.

Many modern engines are also incapable of exceptionally long sequences of moves that lead to a win by one side using zugzwang tactic (a tactic that one side making any move will bring a disadvantage to themselves).

These are some shortcomings of evaluation - search chess engines at the time being, but these will be unraveled soon by neural networks soon.

Currently, neural networks are still limited to the public and many have expressed that it is unfair for Stockfish to play against AlphaZero. Nakamura, a chess grandmaster, comments on this unfairness:

"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 running on what would be my laptop. If you want to have a match that is comparable, you have to have Stockfish running on a supercomputer as well."

Despite criticism of not allowing tree search engines to play on the same hardware, neural networks AI has indeed demonstrated a real milestone in chess analysis.

References  
[1] Oshri, B., & Khandwala, N. (2016). Predicting moves in chess using convolutional neural networks.  
[2] Baxter, J., Tridgell, A. and Weaver, L., 2000. Learning to play chess using temporal differences. Machine Learning, 40(3), pp.243-263.