Midway Report

Elliot Harris

December 18, 2023

1 Definitions

Before commencing into this paper, below are some definitions to help structure the language of this paper, and provide consistency where natural language often fails to.

1. Chess Engine: References any program, software, computer, or other non-human entity that is designed specifically to play the game of chess. Used interchangeably with chess computer, computer (if the context is chess), or chess artificial intelligence. For the most part, every chess engine has two components, an evaluation function, which is used to attach a score to a static chess position, and a search function, which is used to generate and eliminate as many positions as possible to find an ideal candidate move.

2. Elo rating: A mathematical rating invented by Professor Arpad Elo which is now used in more areas than just chess, but commonplace in determining an objective level that an engine or human can play chess. [3]

3. Strength: A less exact word for the elo of a chess engine. In other

words, how good a player is at chess.

4. Artificial Intelligence: An umbrella term for any instance in which a computer replicates intelligence in a narrow or broad field. Will usually be abbreviated to AI.

5. Neural Network: A specific type of artificial intelligence, also under the umbrella of machine learning, in which nodes or 'neurons' are trained in various fashions to output probabilities in order to predict, generate, or otherwise formulate an output based on input.

2 Background

Artificial Intelligence has emerged as one of the most impactful technologies humanity has ever wielded. Healthcare, business, policy, analytics, education, and even industries such as gaming or entertainment have drastically changed due to the powerful predictive and generative capabilities of AI [7]. Despite all the new potential uses of AI and deep neural networks with billions of parameters, one of the first tests of a new computing system or intelligence paradigm links back to the two millennium old game of chess [2]. Whether humanity's obsession with chess is linked to its perceived connection to intelligence, or because it so aptly allows for a proxy war between kings, countries or important figures, the fact remains that chess has held international attention for centuries. In fact, chess popularity burgeoned during the pandemic, and continues its popular trend today, with sites such as Chess.com boasting more than 150 million members [6].

Chess was almost synonymous with early AI. Alan Turing, sometimes deemed as the father of modern computing, sought to develop a chess program, Turochamp, as one of his first attempts at artificial intelligence [8]. Since then, some of the most successful examples of chess AI breakthroughs have involved specialized chips designed for the game of chess, such as Carnegie Mellon's ChipTest in 1987 [5] or IBM's Deep Blue, the computer that famously defeated Russian grandmaster Gary Kasparov in 1997. That match is also widely accepted as the first moment AI surpassed human ability in the game [2]. Probably the next most important milestone in AI's role in chess was Google's Alpha Zero in 2017, which learned to play chess at an unprecedented strength playing games only against itself in an unsupervised deep learning model. [13]. However, not only did these breakthroughs advance our understanding of how computers might master the game of chess, they also provided an undeniable piece of evidence for the efficacy and power of artificial intelligence. Chess, a game that involves both sides having access to all available data, a nearly infinitely deep search tree of possibilities, and a standardized Elo rating system to objectively measure strength, provides the perfect test for a new AI approach. Before unleashing a potentially life-altering system to diagnose a patient, or make a large financial investment, it makes sense that AI first prove itself on the battleground of 64 squares.

As chess engines have become more sophisticated and powerful, so has the broader world of AI. For instance, Large Language Models (or LLMs for short) have become increasingly rampant in both research and real-world use. These models can consist of over ten billion parameters [16], and can at least convince the untrained eye of their understanding in not only language, but also in their ability to code, do math and logic, and much more. Even in other applications of neural networks, there has been a wide and concerning difficulty in visualizing, or even explaining some of the outputs to these models [15]. Misinformation, racist, sexist or otherwise biased decisions, and sometimes just completely incorrect outputs are not rare [7]. And while incorrect predictions are nothing new to the world, the inability to detect where and why they are happening is problematic [7]. Neural networks have also become commonplace in the top chess engines, with engines like Stockfish switching over to a neural network evaluation function and seeing a huge increase in strength [14]. The issue is that it was once possible to ascertain and learn from AI and apply it to human action, but now it is nearly impossible to glean how it achieves its output. In the context of chess, a strong chess engine is not quite as useful, if a human player cannot analyze its decisions and apply similar logic to their own game. It is this problem of transparency that begs the central research question: Is it possible to create an AI that is both transparent and powerful?

3 Summary

This senior project will create a series of chess engines, differing in three key approaches in the implementation of the evaluation function. However, as a basis for this endeavor, there first needs to be a skeleton chess engine that can understand the rules of chess, as well as the ability to optimally search through the many plies of a chess game's search space. This task, which will ultimately result in the completion of a search function, will be completed in the C++ programming language. There is a large foundation of iterative research on how one can either create such a program, or borrow such a shell from a successful chess engine such as Stockfish [14] or Komodo [10]. While interesting and highly relevant to the results of my research, this search-space problem is actually better left to algorithmic solutions, such as the min-max algorithm with a multitude of optimizations (alpha-beta pruning, quiescence search, etc.) [4]. Further, although the result of this program will naturally be unique, it is not the main emphasis of this senior research project, and this first undertaking is more for a personal understanding of chess engine architecture. None of the search-optimizations will be original, and even the 'under-the-hood' representation of the game of chess in the engines will be delegated to Github User Disservin's chess library [1].

The Artificial Intelligence aspect that is of interest to this project is in the evaluation function. For this function there will be three implementation approaches: First, and most central to the research question, is a genetic algorithm based evaluation. Paramount to any genetic algorithm is the fitness function. While it is undoubtedly true that the ideal fitness function for a straightforwardly competitive game such as chess is the game result of the various chromosomes playing each other, such an approach is too computationally expensive. As a result, the project's approach will be based on the research from David Tabibi et al. paper on simulating evaluation functions [9], where the fitness function is based on how accurately the evaluation function of the chess engine matches that of an 'expert' system. The 'expert' will be tested in two ways, first it will take the shape of another strong computer, and the second, the expert will be the decisions of chess Grandmasters. While an expert computer will be much higher in Elo than any expert human, it is possible that a computer based more on human play will better answer the question of transparency, whilst still being able to play at a higher playing strength.

The main reason that a genetic algorithm is the approach of choice is in it's interpretability. The analyzer has access to all features and their subsequent values. At any point in the training of the model, one can stop and look at each subsequent model's strengths and weaknesses. And further, there is one driving function that the model is trying to optimize, and thus the results can easily be traced to the logic of a singular fitness function. Lastly, because genetic algorithms aim to replicate the process of natural selection and the passing down of traits as found in nature and biology, the overall intuitiveness of the model is clear from even a non-technical perspective.

Compared to the neural network, which aims to simulate the largely unknown human processes in the brain, genetic algorithms hold an enormous advantage in their transparency. It finally must be noted that a list of finite and predetermined features allow for much more ease in understanding what a model takes into account en route to its final output.

The second piece of Artificial Intelligence is "hard-coded" intelligence. As shown in Figure 1, I have decided on a list of features heavily influenced by the features of [9]. I will extract each feature from a given chess position, and each engine will be tasked with weighting the features. For the hard-coded features, I will simply utilize intuitive guesses, and draw upon my own chess knowledge and agreed upon standards [8] to provide a baseline strength. My hope is that a more formal and thorough approach, such as a genetic algorithm or a traditional machine learning approach will yield better results. While the original source included 43 features, I have simplified or modified this list to the 28 features that proved to be the most important.

Parameter	**Description**
Pawn Value	The value of one pawn
Knight Value	The value of one knight
Bishop Value	The value of one bishop
Rook Value	The value of one Rook
Queen Value	The value of one Queen
Passed Pawn Mult	Pawn with no other pawns in its way
Doubled Pawn Penalty	Pawns stacked on the same file
Isolated Pawn Penalty	Pawn with no adjacent pawns
Backward Pawn Penalty	Pawn with all adjacent pawns advanced
Weak Square Penalty	Square on your side where no pawns can attack
Passed Pawn Enemy King Square	Rule of the square
Knight Sq Mult	Optimal Knight Position
Knight Outpost Mult	Knight is on a weak square
Bishop Mobility	How many squares a bishop can move to
Bishop Pair	Possessing both colored bishops
Rook Attack King File	Rook is on same file as enemy king
Rook Attack King Adj File	rook is on adjacent file to enemy king
Rook 7th Rank	Rook is on second to last rank
Rook Connected	Rooks can guard each other
Rook Mobility	How many squares a rook has
Rook Behind Passed Pawn	Rook is behind a passed pawn
Rook Open File	Rook has no pawns in its file
Rook Semi Open File	Rook has no friendly pawns in its file
Rook Atck Weak Pawn Open Column	Rook can directly attack a weak enemy pawn
Queen Sq Mult	Optimal Queen Position
King Friendly Pawn	How many and how close are friendly pawns
King No Enemy Pawn Near	King is far away from enemy pawns
King Pressure Mult	How many enemies can get to the king in one move

Figure 1: Table of Position Features with Descriptions

The last artificial intelligence approach for the problem of creating strong or novel chess engines will be a neural network approach. As mentioned before, this is the current standard for many strong engines. Despite decades of testing and handcrafting evaluation functions, the neural network has now gotten powerful and optimized enough to offer a significant increase in strength for modern chess engines [4]. While I may not be able to completely replicate the strength of these vast networks that are providing major improvements to our understanding of chess, I may be able to implement them as a way to create a more 'human' style of play, such as found in Microsoft's Maia Chess [11].

To summarize, I will create three distinct chess engines in the C++ programming language, with all iterations implementing the same core features, search, and chess representations and abstractions. The engines will vary in their approach to weighting the features, through genetic algorithms, human intuition, or a neural network. They will also vary in their objective, whether it be to replicate strong computer chess, grand-master level play, or provide a more accessible output for human education.

4 What I have done so Far

Thus far, I have conducted and written research on many topics that are in my senior project's area of interest. These sources make up an annotated bibliography of 17 sources, ranging from topics such as genetic algorithms and their applications to chess, to open source chess software and the architecture of chess engines. I have also accomplished some solid foundational steps towards beginning to understand the technical underpinnings of this endeavor. I initially, in the interest of speed and comfortability, planned to write the vast majority of code in Python. In this vein, I built a starter project consisting of a pythonic object oriented genetic algorithm solution to the travelling sales problem. Using a real-world data-set of significant size (170 cities), I was able to create a program that could not only estimate a solution within minutes, but also beat the estimates of various CHAT-GPT Turbo suggestions.

I also, and perhaps a little more relevantly, began work on the bare-bones Python chess engine that I would iteratively improve. I successfully implemented a min-max algorithm with alpha-beta pruning, move-ordering, a principal variation heuristic within an iterative deepening approach, and a transposition table. However, the disadvantages of using Python for a computationally heavy project became transparent upon the implementation of that least feature, the transposition table. Although every top chess engine uses their own version of a transposition table, usually with Zobrist hashing [8], the saved search space was no match for the extra computation that was needed for hashing and re-hashing hundreds of thousands of chess positions. In an interpreted language such as Python, this was far too slow.

I also spent a week or so looking into Python speedups, delving into libraries such as PyPy [12], Cython, or even delegating key parts of the code to C. However, despite some initial success, the overall concern remained true, that the speed in which Python was able to run even basic commands was an insurmountable bottleneck when dealing with more than a million search nodes per search.

Given these discoveries, I opted to build the entire code base in C++. C++ has a couple distinct advantages to C (and the obvious advantage to Python in speed and it being a compiled language). C++ allows for an object oriented approach, which given the reusability standards and overall structure of a multi-layered project, I believe this to be a benefit. It also is the language of choice for Stockfish, one of the world's best chess engine. The only downside for restarting the base chess engine code in C++ is the learning curve and the nature of C++ containing a lot more boilerplate and complexity than Python. Still, after building a basketball simulation with C++ and using shared pointers, class enums, and a whole host of C++ features, I am confident I will be able to code the final product.

I have also decided on a chess library programmed in C++, from the Github user Disservin [1]. In fact, this library was used to build a 3300 elo [8] chess engine, and Perft tests (Performance Test Move Path Enumeration) on the readme.md document for the repository show the high speed that it can generate positions (220 million nodes per second) [1].

References

[1] Disservin's Chess Library. C++ library for chess-related operations, 2023.

[2] Akdemir, A. Tuning of chess evaluation function by using genetic algorithms.

[3] Author. Prof. arpad e. elo is dead at 89; inventor of chess ratings system. *The New York Times* (November 14 1992).

[4] Bijl, P., and Tiet, A. Exploring modern chess engine architectures. *Victoria University, Melbourne* (2021).

[5] Carnegie Mellon University. The iconclast.

[6] Chess.com. Chess.com members, December 2023.

[7] Christian, B. *The Alignment Problem: Machine Learning and Human Values*. W. W. Norton & Company, October 2020.

[8] contributors, C. P. W. Chess programming wiki, 2019. Accessed: October 1, 2023.

[9] David-Tabibi, O., Koppel, M., and Netanyahu, N. S. Expert-driven genetic algorithms for simulating evaluation functions. *Genetic Programming and Evolvable Machines 12*, 1 (feb 2010), 5–22.

[10] Komodo Chess. Komodo Chess Website. https://komodochess.com/, Accessed in 2023.

[11] Microsoft. The human side of ai for chess, 2020.

[12] PyPy Development Team. PyPy: A fast, compliant alternative implementation of the python language. https://www.pypy.org/, 2023.

[13] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

[14] Stockfish. Stockfish Chess Engine, 2023. GitHub repository.

[15] Suh, A., Appleby, G., Anderson, E. W., Finelli, L., Chang, R., and Cashman, D. Are metrics enough? guidelines for communicating and visualizing predictive models to subject matter experts. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–16.

[16] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. A survey of large language models, 2023.