**AlphaZero Paper:**

“Recently, the AlphaGo Zero algorithm achieved superhuman performance in the game of Go, by representing Go knowledge using deep convolutional neural networks (22, 28), trained solely by reinforcement learning from games of self-play (29). In this paper, we apply a similar but fully generic algorithm, which we call AlphaZero, to the games of chess and shogi as well as Go, without any additional domain knowledge except the rules of the game, demonstrating that a general-purpose reinforcement learning algorithm can achieve, tabula rasa, superhuman performance across many challenging domains.”

(DeepMind, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, 2017)

DeepMind first created the algorithm of AlphaGo Zero on the game of Go before starting their research on Chess. They have said “superhuman performence” was achieved using “deep convolutional neural networks”. In the game of Chess, they have used a “similar but fully genetic algorithm”.

“A landmark for artificial intelligence was achieved in 1997 when Deep Blue defeated the human world champion” (DeepMind, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, 2017)

This shows that the algorithm AlphaZero used was sufficient enough to beat the world chess champion. They used “high-performance alpha-beta search”

“Instead of an alpha-beta search with domain-specific enhancements, AlphaZero uses a general purpose Monte-Carlo tree search (MCTS) algorithm.” (DeepMind, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, 2017)

This means that alpha beta was slow and they needed something better to use. MCTS was a better option in terms of finding better moves and time efficiency.

“At the end of the game, the terminal position sT is scored according to the rules of the game to compute the game outcome z: -1 for a loss, 0 for a draw, and +1 for a win.”

(DeepMind, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, 2017)

This is the score for the outcome of AlphaZero.

“In AlphaGo Zero, self-play games were generated by the best player from all previous iterations. After each iteration of training, the performance of the new player was measured against the best player; if it won by a margin of 55% then it replaced the best player and self-play games were subsequently generated by this new player. In contrast, AlphaZero simply maintains a single neural network that is updated continually, rather than waiting for an iteration to complete.”

(DeepMind, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, 2017)

This paragraph says that the best players from the previous generations are bred to make a better AI. If a winner has won by 55%, it replaces the best player. AlphaZero is using a single neural network that updates after every game rather than each generation which helps it speed up the process of learning significantly.

**Artificial Neural Networks:**

“Each connection, like the [synapses](https://en.wikipedia.org/wiki/Synapse) in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a [real number](https://en.wikipedia.org/wiki/Real_number), and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a [weight](https://en.wikipedia.org/wiki/Weight_(mathematics)) that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.”

(Wikipedia, Artificial Neural Network, 2020)

This page describes what an artificial neural network is, how it learns and how it is very similar to the human brain.

**Evaluation:**

Chess has different piece values for each piece and sometimes it could change depending on the position. For example, the most common values for the pieces are; Pawn = 1, Knight = 3, Bishop = 3, Rook = 5, Queen = 9.

However there are some extra factors;

* Bishop is always worth more than 3 Pawns
* Knight is always worth more than 3 Pawns
* Rook is always worth more than 5 Pawns
* Sometimes Bishop is worth more than a Knight
* Bishop + Knight > Rook + Pawn
* Rook + 2 Pawn > Bishop + Knight > Rook + Pawn
* Bishop + Knight = Rook + 1.5 Pawn
* Queen + Pawn = 2 Rook

This means that;

* Bishop > Knight > 3 Pawns
* Bishop + Knight = Rook + 1.5 Pawn
* Queen + Pawn = 2 Rook
* Bishop + 2 Pawn > Knight + 2 Pawn > Rook

(ChessProgrammingWiki, Simplified Evaluation Function, 2018)

Although the 1-3-3-5-9 system of point totals is the most commonly given, many other systems of valuing pieces have been proposed. Several systems give the bishop slightly more value than the knight. A bishop is usually slightly more powerful than a knight, but not always; it depends on the position.

(Wikipedia, Chess Piece Relative Value, 2020)

The pieces not always have the same value. In the Alternative Valuations different systems have a different value for each piece. As can be seen, AlphaZero assigned the values to be; Pawn = 1, Knight = 3.05, Bishop = 3.33, Rook = 5.63, Queen = 9.5. This shows that by using AI and Machine Learning, a better approximation of the value system in chess can be found.

“For example, pawn structure can have terms for isolated, doubled, backward, advanced, passed, protected passed, connected passed, holes, semi-open and open files, pawn majorities, phalanxes, and many other formations. Other special factors that are often considered are: development of the minor pieces, rooks on open files or the seventh rank, doubled rooks, outpost knights (knights in central locations protected by a pawn and not subject to attack by an opposing pawn), possession of the bishop pair, bishops on the long diagonals, pieces occupying or bearing on spaces around the opposing king, and mobility of the kings (kings shouldn't be 'cramped', hence subject to mate-on-the-move). Some terms, such as king safety in an endgame with few pieces, can and should be ignored depending on context.”

(Wikipedia, Evaluation Function, 2020)

(ChessProgrammingWiki, Evaluation, 2020)

This paragraph says that different pawn structures lead to a different game and position of the minor pieces are important too. For example, mobility of the king is important as if it cannot move, it is more vulnurable to being checkmated.

**Policy Function:**

“The policy function is a measure of weight assigned to a tree of possible moves to examine.”

(Mclaughlin, 2019)

“Recall that the value function describes the best possible value of the objective, as a function of the state x. By calculating the value function, we will also find the function a(x) that describes the optimal action as a function of the state; this is called the policy function.”

(Wikipedia, Bellman Equation, 2020)

**Monte Carlo Tree Search:**

* “Selection: Start from root R and select successive child nodes until a leaf node L is reached. The root is the current game state and a leaf is any node that has a potential child from which no simulation (playout) has yet been initiated. The section below says more about a way of biasing choice of child nodes that lets the game tree expand towards the most promising moves, which is the essence of Monte Carlo tree search.”
* “Expansion: Unless L ends the game decisively (e.g. win/loss/draw) for either player, create one (or more) child nodes and choose node C from one of them. Child nodes are any valid moves from the game position defined by L.”
* “Simulation: Complete one random playout from node C. This step is sometimes also called playout or rollout. A playout may be as simple as choosing uniform random moves until the game is decided (for example in chess, the game is won, lost, or drawn).”
* “Backpropagation: Use the result of the playout to update information in the nodes on the path from C to R.”

(Wikipedia, Monte Carlo Tree Search, 2020)

In simpler words:

* Selection – A random move (node) is chosen.
* Expansion – Game will expand from that chosen move (node).
* Simulation – Different simulations will be played.
* Backpropogation – The chosen node is updated by going up the nodes.

**Reinforcement Learning:**

“Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).”

(Wikipedia, Reinforcement Learning, 2020)

In reinforcement learning, the system does not need labelled data in order to learn. It will explore (unexplored nodes) and exploit (explored nodes) the nodes to improve itself for the current problem.

Exploration – The program will choose a unexplored nodes move that couldve missed an important move.

Exploitation – The program will choose a random move that could be a mistake to find a better move in the end.

(ChessProgrammingWiki, UCT, 2020)

**Minimax:**

“Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc.). It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.”

(Wikipedia, Alpha Beta Pruning, 2020)

(Wikipedia, Minimax, 2020)

**Current Project:**

In my current game, I have developed a chess website that the user can play against the AI. The user is only allowed to play their pieces and make legal moves. There is also a reset game button to reset the board. The AI uses minimax with alpha-beta pruning.

I have used <https://chessboardjs.com/> to display the board on a website using JavaScript. Moving pieces with animations.

I have used <https://github.com/niklasf/python-chess> for the chess library. To see the board state, if the moves are legal, king is in check, king is pinned, en passant, castling, etc.

I have used <https://flask.palletsprojects.com/en/1.1.x/> for REST and connecting between JS and Python. Sending requests from the move made (chessboard.js) and getting a response from the move made (python-chess).

# References

ChessProgrammingWiki. (2018, May 16). *Simplified Evaluation Function*. Retrieved from Chess Programming Wiki: https://www.chessprogramming.org/Simplified\_Evaluation\_Function

ChessProgrammingWiki. (2020, August 5). *Evaluation*. Retrieved from Chess Programming Wiki: https://www.chessprogramming.org/Evaluation

ChessProgrammingWiki. (2020, July 16). *UCT*. Retrieved from Chess Programming Wiki: https://www.chessprogramming.org/UCT

DeepMind. (2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.* London: DeepMind.

DeepMind. (2018, Dec 07). *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.* London: DeepMind. Retrieved from https://science.sciencemag.org/content/362/6419/1140

DeepMind. (2020). *Assessing Game Balance with AlphaZero: Exploring Alternative Rule Sets in Chess.* London: DeepMind.

Mclaughlin, A. (2019, November 11). Retrieved from Quora: https://www.quora.com/How-does-Leela-Chess-work-What-are-its-neural-network-inputs-and-output

Wikipedia. (2020, September 1). *Alpha Beta Pruning*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Alpha-beta\_pruning

Wikipedia. (2020, October 12). *Artificial Neural Network*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Artificial\_neural\_network

Wikipedia. (2020, October 14). *Bellman Equation*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Bellman\_equation

Wikipedia. (2020, September 17). *Chess Piece Relative Value*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Chess\_piece\_relative\_value

Wikipedia. (2020, September 10). *Evaluation Function*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Evaluation\_function

Wikipedia. (2020, October 13). *Minimax*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Minimax

Wikipedia. (2020, September 26). *Monte Carlo Tree Search*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Monte\_Carlo\_tree\_search

Wikipedia. (2020, September 26). *Reinforcement Learning*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Reinforcement\_learning