**Project Report**

**Chess AI Using Neural Networks and Reinforcement Learning**

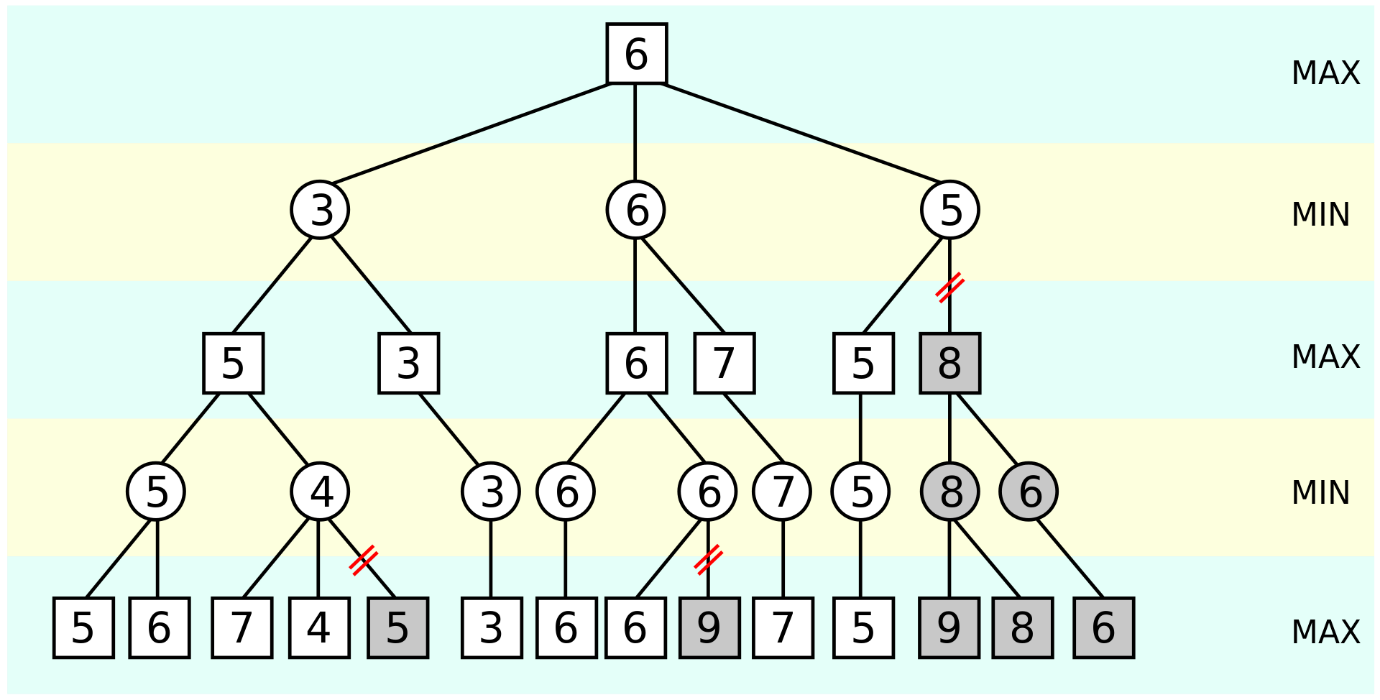
**Introduction:**

Chess has long been a challenging game for artificial intelligence (AI) due to its complex branching factor and strategic depth. Using this AI we will explore how we created our own neural network in order to develop and enhance a competitive chess AI which can understand and make intrusive decisions with more dept and accuracy.

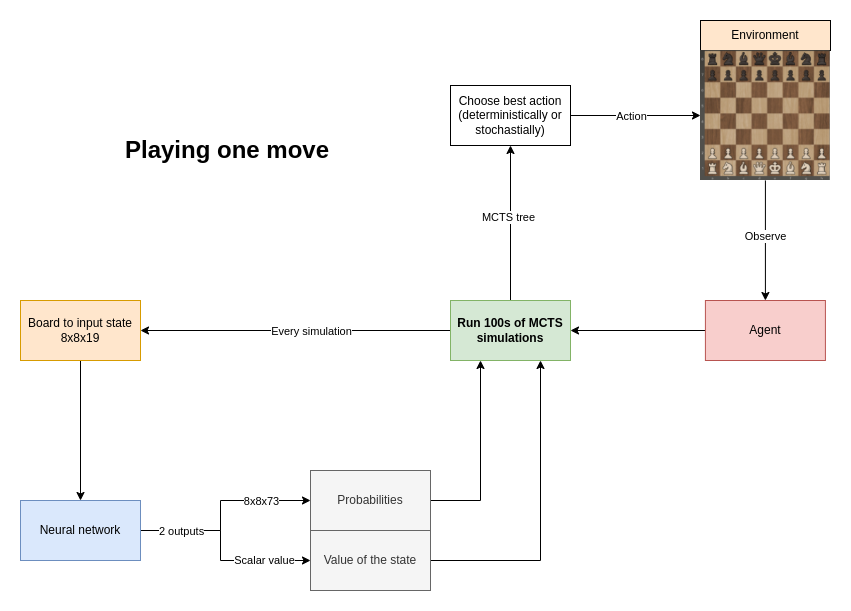
**Algorithm & Methodology**:

**HOW DO NORMAL CHESS ENGINES WORK?**

Normal chess engines work with the minimax algorithm: the engine tries to find the best move by creating a tree of all possible moves to a certain depth, and cutting down paths that lead to bad positions (alpha-beta pruning). It evaluates a position based on which pieces are on the board.



**HOW DOES MY CHESS ENGINE WORK?**This chess engine is based on AlphaZero by Deepmind. It uses a neural network to predict the next best move. The neural network learns by playing against itself for a high amount of games, and using their results to train the network. The newly trained neural network is evaluated against the old network by playing many games against each other, and the best network is kept. This process is repeated for a long time.



**The neural network**

Input layer: 19 8x8 boards of booleans

[Input example](https://github.com/SoloSid69/chess-deep-rl/blob/main/code/tests/input_planes/full.png)

20 hidden layers:

1. Convolutional hidden layer
2. 19 residual blocks with skip-connections

2 outputs:

1. The win probabilities of each move (73 boards of 8x8 floats)
2. The value of the given board (scalar)

[Output example](https://github.com/SoloSid69/chess-deep-rl/blob/main/code/tests/output_planes/unfiltered.png)

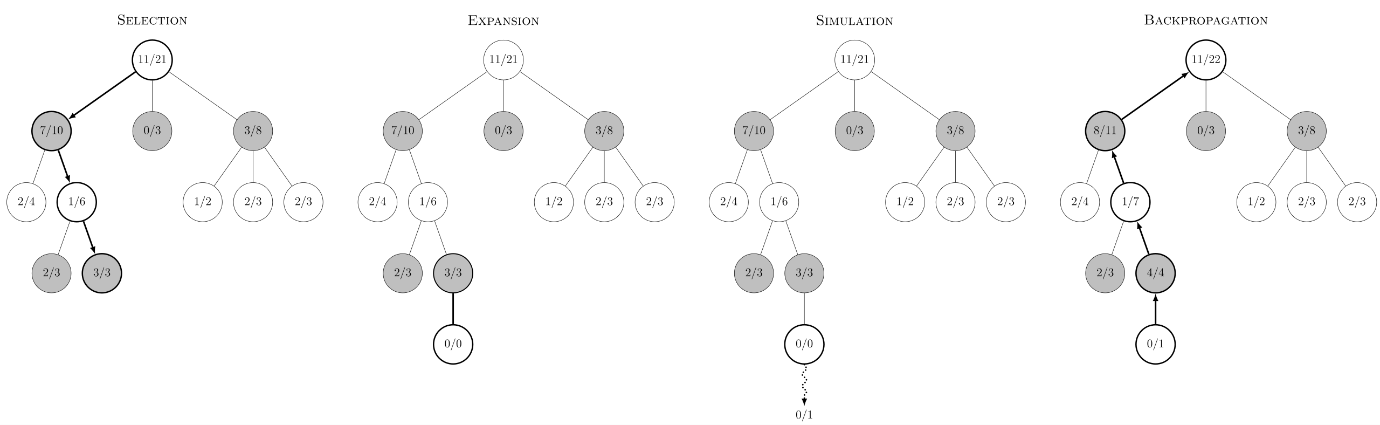
=> 30+ million parameters

Every move, run a high number amount of MCTS simulations. AlphaZero uses an custom version of MCTS.

**Normal Monte Carlo Tree Search:** (<https://en.wikipedia.org/wiki/Monte_Carlo_tree_search>)

1. Selection: Traverse the tree randomly until a leaf node is reached.
2. Expansion: expand the leaf node by creating a child for every possible action
3. Simulation: 'rollout' the game by randomly choosing moves until the end of the game.
4. Backpropagation: backpropagate the result of the rollout to the root node.

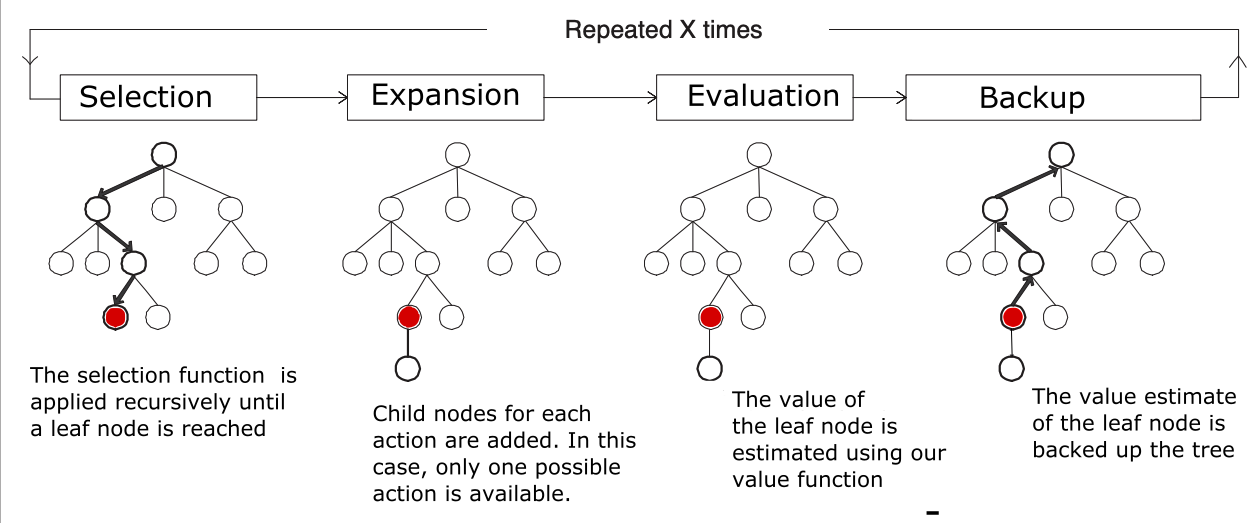
In chess, normal MCTS would be incredibly inefficient, because the amount of actions every position can have is too high (step 1), and the length of the game can be very long when choosing random moves (step 3).

[](https://github.com/SoloSid69/chess-deep-rl/blob/main/code/img/MCTS-wikipedia.png)

**AlphaZero's MCTS**

AlphaZero uses a different kind of MCTS:

* step 1 (Selection) is not random, but based on neural network predictions and upper confidence bound
* step 3 (Simulation) is replaced by the value prediction received by the neural network (Evaluation)

[](https://github.com/SoloSid69/chess-deep-rl/blob/main/code/img/MCTS-alphazero.png)

**To run one MCTS simulation:**

To traverse the tree, keep selecting the edges with maximum Q+U value

* Q = mean value of the state over all simulations
* U = upper confidence bound
* Do this until a leaf node is reached (= a node which has not been visited/expanded yet)

Expand the leaf node by adding a new edge for every possible action in the state

* Input the leaf node into the neural network
* The output:   
   a. The probabilities

b. The value of the state

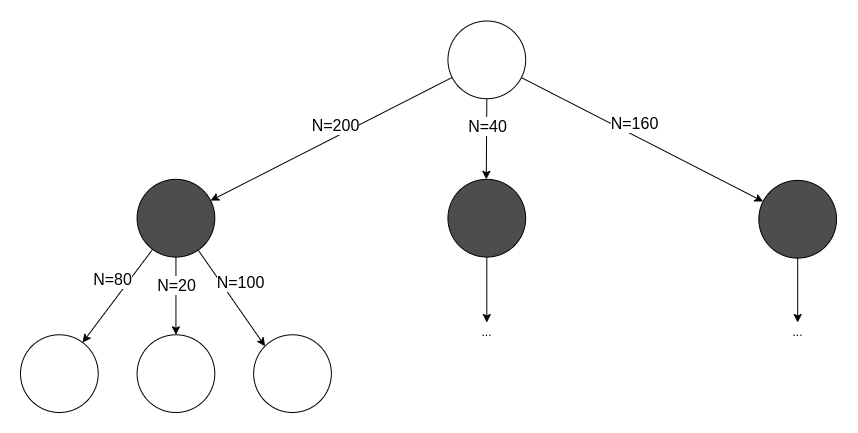
* Initialize the new edge's variables with these values:
  + N = 0
  + W = 0
  + Q = 0
  + P = p\_a (prior probability for that action)
* Add nodes (new states) for each action to the tree

Backpropagation

* From the leaf node, backpropagate to the root node
* For every edge in the path, update the edge's variables
  + N = N + 1
  + W = W + v, v is the value of the leaf node predicted by the NN in step 2.
  + Q = W / N

**After these simulations, the move can be chosen:**

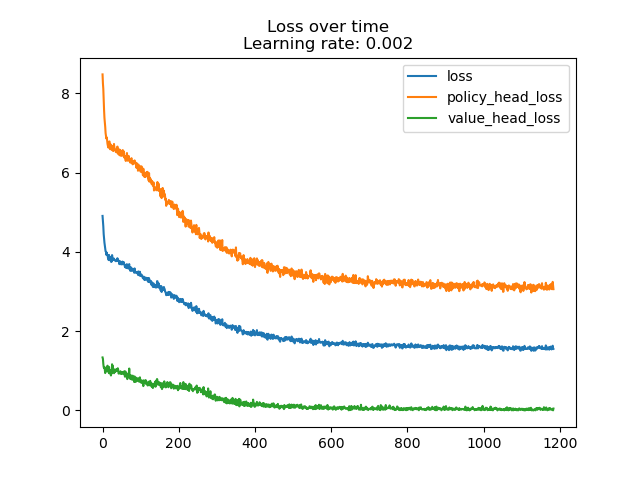
* The move with greatest ***N*** (deterministically)
* According to a distribution (stochastically): π∼ ***N***

[](https://github.com/SoloSid69/chess-deep-rl/blob/main/code/img/MCTS-choose-move.png)

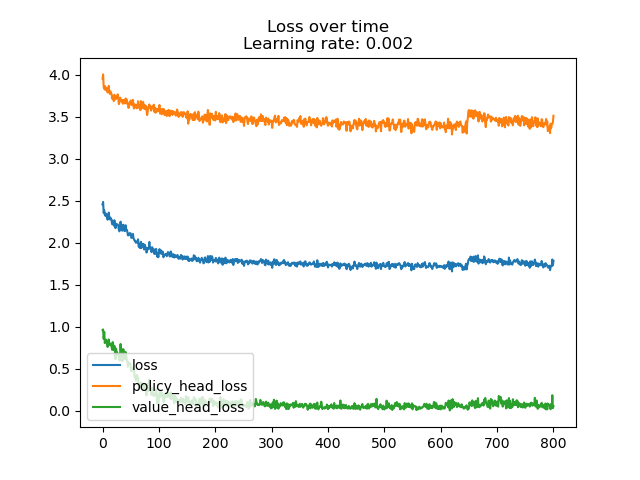
**Result Analysis:**

**(Graphs aaenge):**

**1st Training:**



**2nd Training(1st iteration with learning rate = 0.002):**



**2nd Training(2nd iteration with learning rate = 0.02):**

A graph of loss and loss

Description automatically generated

**2nd Training(3rd iteration with learning rate = 0.2):**

A graph of loss and loss

Description automatically generated

**Conclusion & Future work:**

Future efforts will focus on improving model accuracy, optimizing computational performance, and exploring reinforcement learning for advanced strategies.

**Bibolography:**

**Wikipedia articles & Library documentation**

[1]"Deep reinforcement learning," Wikipedia. Jan. 29, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Deep_reinforcement_learning&oldid=1068657803>

[2]“Reinforcement learning,” Wikipedia. Jan. 15, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=1065862559>

[3]“AlphaZero,” Wikipedia. Jan. 15, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=AlphaZero&oldid=1065791194>

[4]“AlphaGo,” Wikipedia. Jan. 25, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=AlphaGo&oldid=1067772956>

[5]“AlphaGo Zero,” Wikipedia. Oct. 14, 2021. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=AlphaGo_Zero&oldid=1049954309>

[6]“Monte Carlo tree search,” Wikipedia. Jan. 23, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Monte_Carlo_tree_search&oldid=1067396622>

[7]“Minimax,” Wikipedia. Jan. 18, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Minimax&oldid=1066446492>

[8]“Alpha–beta pruning,” Wikipedia. Jan. 30, 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Alpha%E2%80%93beta_pruning&oldid=1068746141>

[9]“python-chess: a chess library for Python — python-chess 1.8.0 documentation.” <https://python-chess.readthedocs.io/en/latest/> (accessed Feb. 01, 2022).

[10]“Technical Explanation of Leela Chess Zero · LeelaChessZero/lc0 Wiki,” GitHub. <https://github.com/LeelaChessZero/lc0> (accessed Feb. 01, 2022).

**AlphaZero & AlphaGo Zero specific articles & papers**

[11]D. Silver et al., “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,” arXiv:1712.01815 [cs], Dec. 2017, Accessed: Feb. 01, 2022. [Online]. Available: <http://arxiv.org/abs/1712.01815>

[12]“A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” <https://www.science.org/doi/10.1126/science.aar6404> (accessed Feb. 01, 2022).

[13]“engines - Understanding AlphaZero,” Chess Stack Exchange. <https://chess.stackexchange.com/questions/19353/understanding-alphazero> (accessed Feb. 01, 2022).

[14]“How does AlphaZero learn to evaluate a position it has never seen?,” Chess Stack Exchange. <https://chess.stackexchange.com/questions/19401/how-does-alphazero-learn-to-evaluate-a-position-it-has-never-seen> (accessed Feb. 01, 2022).

[15]“Figure 2: MCTS in AlphaGo Zero. | Nature”, Accessed: Feb. 01, 2022. [Online]. Available: <https://www.nature.com/articles/nature24270/figures/2>

[16]J. Varty, “Alpha Zero And Monte Carlo Tree Search.” <https://joshvarty.github.io/AlphaZero/> (accessed Feb. 01, 2022).

[17]J. Varty, AlphaZeroSimple. 2022. Accessed: Feb. 01, 2022. [Online]. Available: <https://github.com/JoshVarty/AlphaZeroSimple>

[18]“Was AlphaZero taught castling?,” Chess Stack Exchange. <https://chess.stackexchange.com/questions/37468/was-alphazero-taught-castling> (accessed Feb. 01, 2022).

[19]T. M. Blog, “A Single-Player Alpha Zero Implementation in 250 Lines of Python.” <https://tmoer.github.io/AlphaZero/> (accessed Feb. 01, 2022).

[20]“AlphaZero |.” <https://sebastianbodenstein.net/post/alphazero/> (accessed Feb. 01, 2022).

**Diagrams**

[21]“AlphaGo Zero Explained In One Diagram | by David Foster | Applied Data Science | Medium.” <https://medium.com/applied-data-science/alphago-zero-explained-in-one-diagram-365f5abf67e0> (accessed Feb. 01, 2022).

**Tutorials**

[22]“AlphaZero, a novel Reinforcement Learning Algorithm, in JavaScript | by Carlos Aguayo | Towards Data Science.” <https://towardsdatascience.com/alphazero-a-novel-reinforcement-learning-algorithm-deployed-in-javascript-56018503ad18> (accessed Feb. 01, 2022).

[23]D. Foster, “How to build your own AlphaZero AI using Python and Keras,” Applied Data Science, Dec. 02, 2019. <https://medium.com/applied-data-science/how-to-build-your-own-alphazero-ai-using-python-and-keras-7f664945c188> (accessed Feb. 01, 2022).

[24]D. Foster, “How To Build Your Own MuZero AI Using Python (Part 1/3),” Applied Data Science, Feb. 23, 2021. <https://medium.com/applied-data-science/how-to-build-your-own-muzero-in-python-f77d5718061a> (accessed Feb. 01, 2022).

[25]“Simple Alpha Zero.” <https://web.stanford.edu/~surag/posts/alphazero.html> (accessed Feb. 01, 2022).

[26]D. Straus, “AlphaZero implementation and tutorial,” Medium, Jan. 27, 2020. <https://towardsdatascience.com/alphazero-implementation-and-tutorial-f4324d65fdfc> (accessed Feb. 01, 2022).

Updated article: [27]“How I trained a self-supervised neural network to beat GnuGo on small (7x7) boards | by Darin Straus | Analytics Vidhya | Medium.” <https://medium.com/analytics-vidhya/how-i-trained-a-self-supervised-neural-network-to-beat-gnugo-on-small-7x7-boards-6b5b418895b7> (accessed Feb. 01, 2022).

[28]cody2007, alpha\_go\_zero\_implementation. 2021. Accessed: Feb. 01, 2022. [Online]. Available: <https://github.com/cody2007/alpha_go_zero_implementation>

**Interesting videos**

[29]Lex Fridman, David Silver: AlphaGo, AlphaZero, and Deep Reinforcement Learning | Lex Fridman Podcast #86, (Apr. 03, 2020). Accessed: Feb. 01, 2022. [Online]. Available: <https://www.youtube.com/watch?v=uPUEq8d73JI>

[30]DeepMind, RL Course by David Silver - Lecture 1: Introduction to Reinforcement Learning, (May 13, 2015). Accessed: Feb. 01, 2022. [Online]. Available: <https://www.youtube.com/watch?v=2pWv7GOvuf0>

[31]Aske Plaat, Keynote David Silver NIPS 2017 Deep Reinforcement Learning Symposium AlphaZero, (Dec. 10, 2017). Accessed: Feb. 01, 2022. [Online]. Available: <https://www.youtube.com/watch?v=A3ekFcZ3KNw>