test.py: Pytorch implementation of Lc0 engine, specifically T79 engine.

Further work: Verify operation by loading in weights from file. At the end of the file some code has been written to read the weights from the weights file but the format in which it is saved is not included in the documentation, so you have to read the code that produced it to figure it out.

Network produced based on: <https://lczero.org/dev/backend/nn/>

(Current, not classical)

Main.py: Run maia, Lc0 and stockfish using their exe files

Never used after creation

For tree generator: <https://github.com/jkormu/Leela-chess-Tree>

Ignore read me and just clone the repo. Use readme to understand how to use the app

Replace engine.py using my engine.py to save the graphs after each run

Further work: Make a save button instead of saving every run, include option for user to choose name to specify engine used.

Readtree.py: Load trees and compare them.

Functions:

most\_visited\_move: Exactly what is says on the tin, the move visited move visited in the graph.

suggested\_move: Best move according to engine as well as the node it is

continuation: best continuation (Hard coded to 3 moves in, please change possibly add as argument)

weak\_engines\_move\_for\_strong\_move: Function that takes in the move of the strong engine and figures out what the weaker engine thinks is the best continuation. Checks if they agree or disagree. If they disagree, shows the continuation they each suggest. Further work: include NLP to explain why it missed the move (Possibly blind to knight moves etc)

number\_of\_leaves: Breadth of trees

eval\_of\_first\_moves: Compares average Q value (position evaluation) in the first layer of tree. Shows how good the engines are at ignoring bad moves from the start. +1 is added to all evaluations to not reward large negative values. This needs refining as some positions are bad for player so need to reward finding best of a bunch of bad moves.

num\_of\_nodes\_from\_source: Number of nodes in first layer