Chess Cheating Detection Using Deep Learning

Calvin Pozderac, Philip Barron, Nathaniel Tamminga

Motivation

- Magnus Carlsen vs. Hans Niemann (2022)
- Increase in online formats
- Former World champion Kramnik

chessbase.com

Crystal Fuller/Saint Louis Chess Club

Stockfish

Stockfish is a chess engine that looks at a given chess position, and at all legal moves up until a certain *depth*.

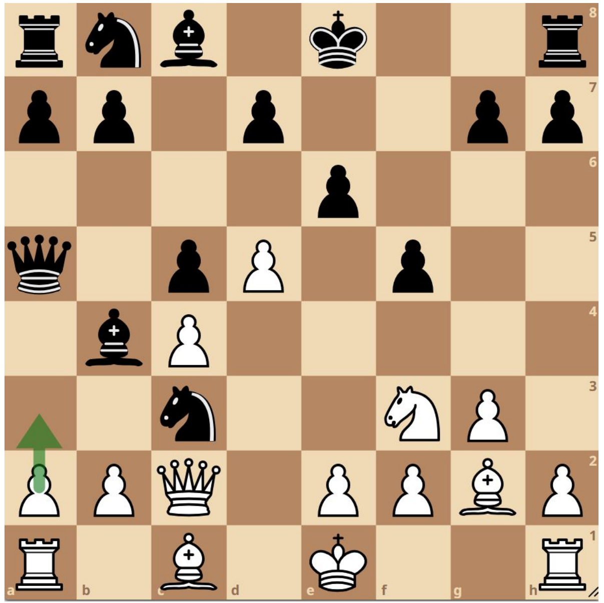
The longer Stockfish runs, it can look deeper into future positions to determine the best possible moves.

Calculating is a daunting task – for instance from the starting position in chess there are 69.3 trillion possible positions after 10 totals moves between White and Black.

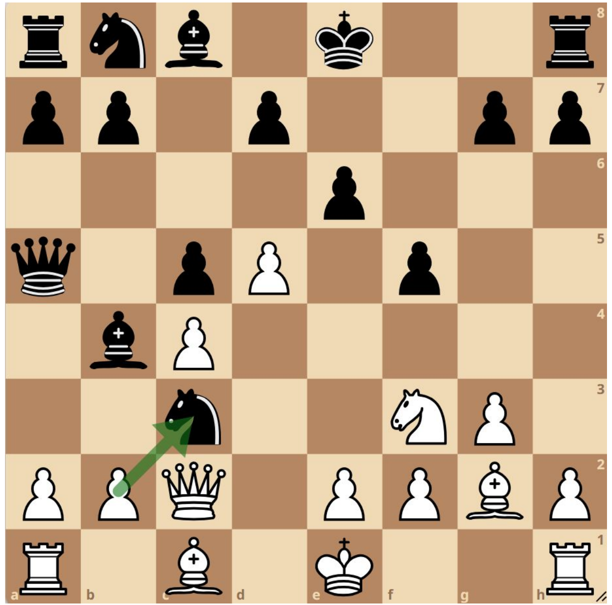
Data Sources

- 4 million classical GM Over The Board games
- 66 million engine evaluated positions

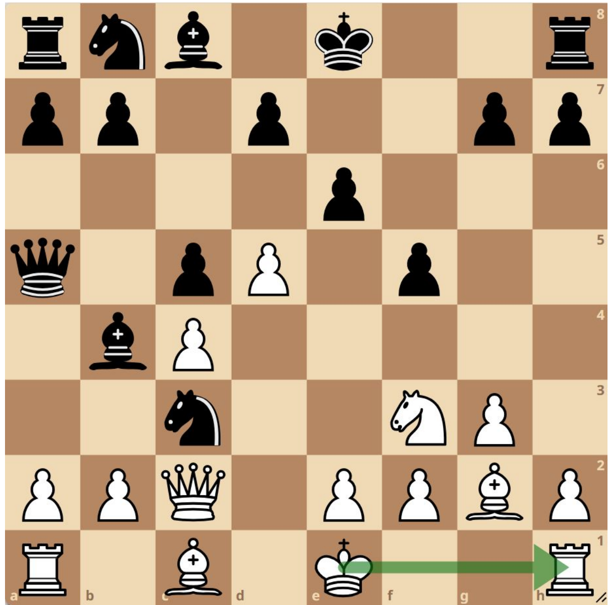
Human-only



Human or Engine

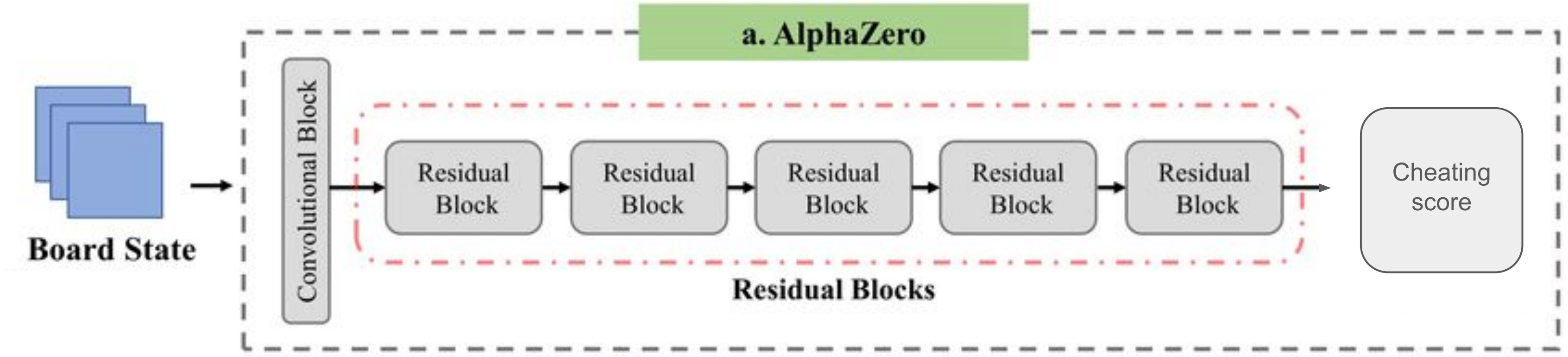


Engine-only



Convolutional Neural Network

- Inspired by AlphaZero (achieved superhuman chess play in 2017)
- Input tensor encodes the board and move into a (20,8,8) size tensor
- Binary cross entropy loss with SGD

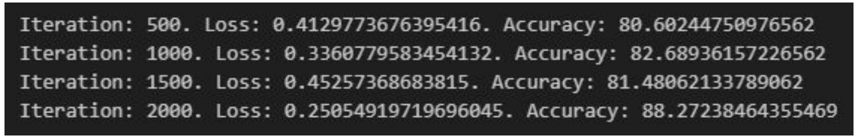


Feed-Forward Neural Network

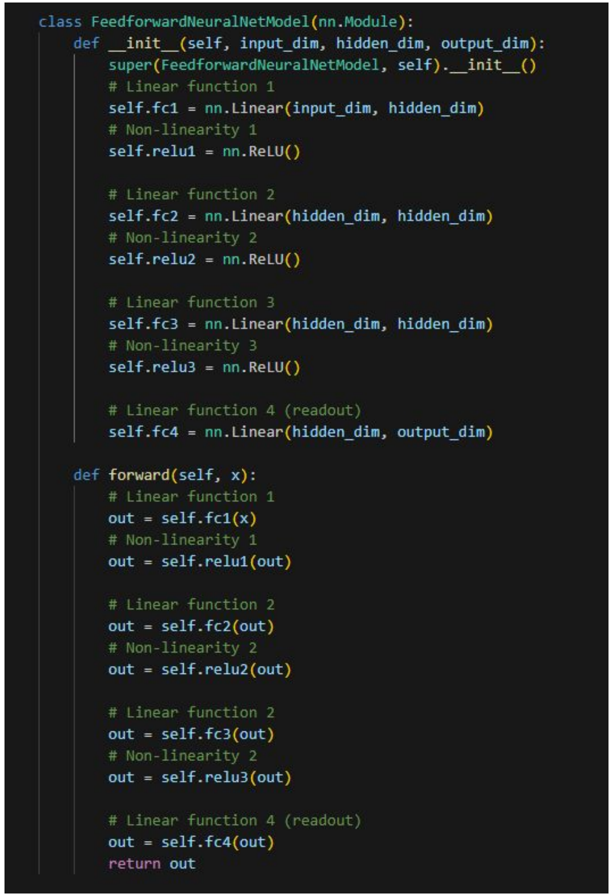
The FNN trained faster than the CNN, but did not have good accuracy.

FNN architecture is seen to the left. Training loss and accuracy is below.

Best Training Accuracy: 88%



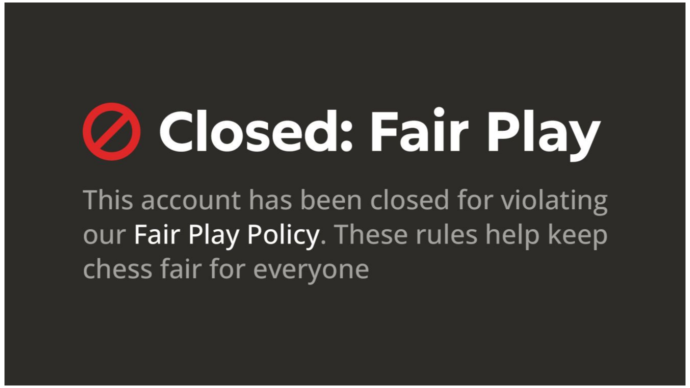
```
Iteration: 500. Loss: 0.4129773676395416. Accuracy: 80.60244750976562
Iteration: 1000. Loss: 0.3360779583454132. Accuracy: 82.68936157226562
Iteration: 1500. Loss: 0.45257368683815. Accuracy: 81.48062133789062
Iteration: 2000. Loss: 0.25054919719696045. Accuracy: 88.27238464355469
```



```
class FeedforwardNeuralNetModel(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(FeedforwardNeuralNetModel, self).__init__()
        # Linear function 1
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        # Non-linearity 1
        self.relu1 = nn.ReLU()

        # Linear function 2
        self.fc2 = nn.Linear(hidden_dim, hidden_dim)
        # Non-linearity 2
        self.relu2 = nn.ReLU()

        # Linear function 3
        self.fc3 = nn.Linear(hidden_dim, hidden_dim)
        # Non-linearity 3
        self.relu3 = nn.ReLU()

        # Linear function 4 (readout)
        self.fc4 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        # Linear function 1
        out = self.fc1(x)
        # Non-linearity 1
        out = self.relu1(out)

        # Linear function 2
        out = self.fc2(out)
        # Non-linearity 2
        out = self.relu2(out)

        # Linear function 2
        out = self.fc3(out)
        # Non-linearity 2
        out = self.relu3(out)

        # Linear function 4 (readout)
        out = self.fc4(out)
        return out
```

Future Directions

- Train data for longer with more compute
- Combine the move identifier into a game detection protocol
- Apply chess detection algorithm to Lichess data
- Test Niemann’s suspicious moves



Closed: Fair Play

This account has been closed for violating our Fair Play Policy. These rules help keep chess fair for everyone

chess.com

lichess.org