FIND YOUR NEXT CHESS MOVE

Madeline Vossbrinck

AGENDA

- Background
- Tools
- Minimax Algorithm
- Neural Network Model
- Evaluation
- App Demo
- Conclusions

BACKGROUND

- Obtain improvement to my own chess game
- Chess.com has over 60 million members
  (https://www.chess.com/members)
- US Chess Federation paid membership peaked at over 96,000 before the COVID-19 pandemic
  (https://new.uschess.org/sites/default/files/media/documents/us-chess-2020-annual-report.pdf)

TOOLS

EDA, Algorithms, Functions

Python Packages:

- Pandas
- NumPy
- Chess
- Matplotlib
- Keras
- Tensorflow

Chess App

- Flask
- HTML
- Javascript
- CSS
- Heroku

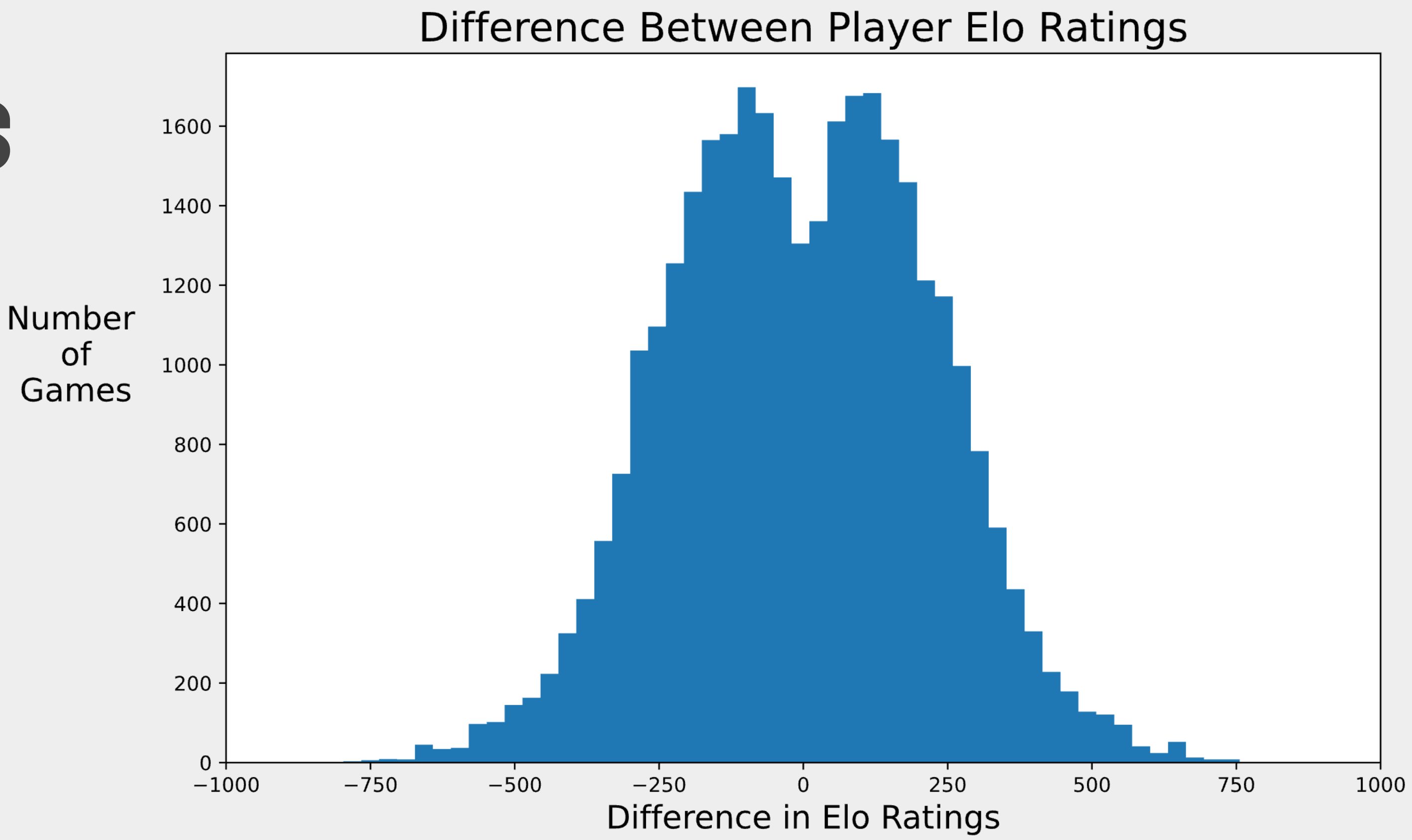
MINIMAX ALGORITHM

- Algorithm commonly used in game theory
  - Creates a tree of next possible moves
  - Calculates maximum outcome using heuristic, assuming opposing player will try to minimize outcome of the current player
- Baseline recommender uses heuristic based on relative chess piece values

Neural Network Model

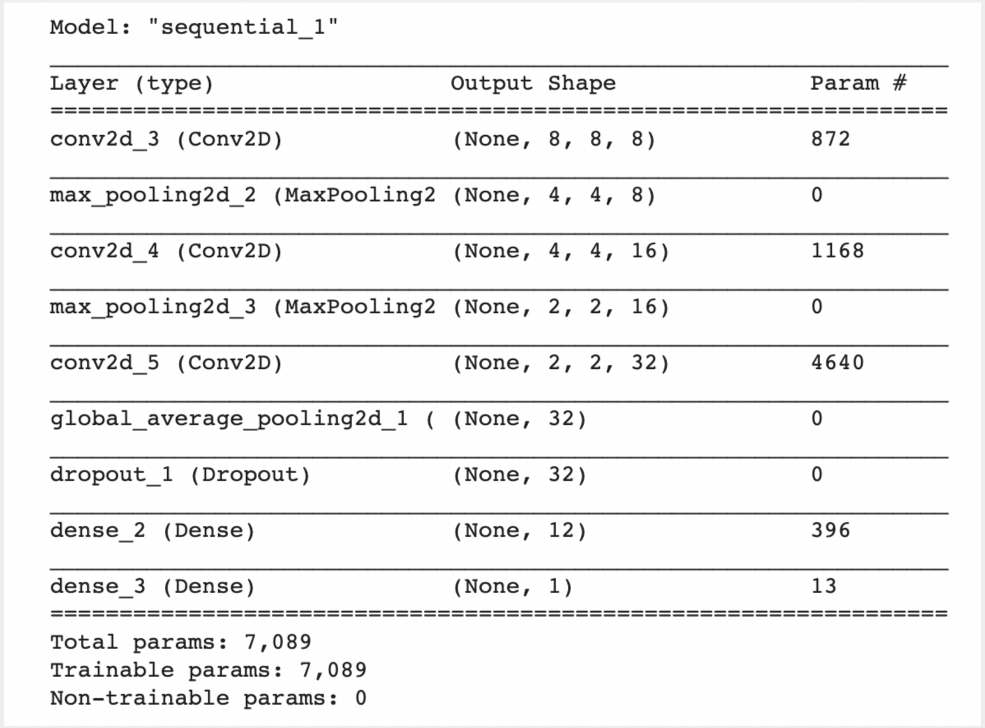
- Convolutional Neural Network
- Data Set:
    - Chess games from Lichess Open Database (https://database.lichess.org/)
    - September 2014
    - All players had Elo ratings of at least 1800 and at least one player had an Elo rating of 2100 or higher
    - 32,740 games with 2,397,813 total moves

ELO RATINGS


Difference Between Player Elo Ratings
Number of Games
0
200
400
600
800
1000
1200
1400
1600
−1000
−750
−500
−250
0
250
500
750
1000
Difference in Elo Ratings

FINAL CONVOLUTIONAL NEURAL NETWORK MODEL

- 3 Convolutional Layers
- 2 Pooling Layers
- 1 Global Pooling Layer
- 1 Dropout Layer
- 2 Dense Layers



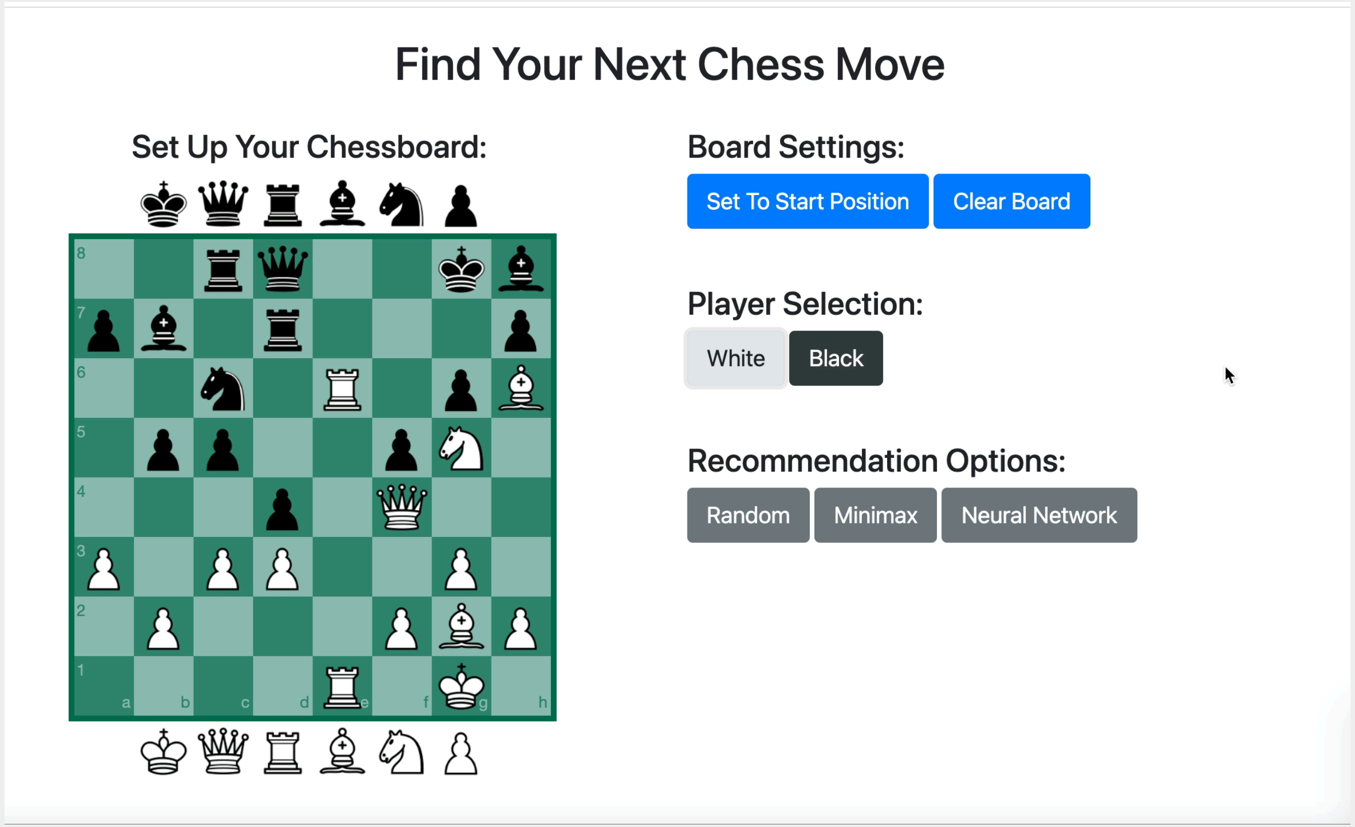
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 8, 8, 8)           872
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 8)           0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 16)          1168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 16)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 2, 32)          4640
_________________________________________________________________
global_average_pooling2d_1 ( (None, 32)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                396
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 13
=================================================================
Total params: 7,089
Trainable params: 7,089
Non-trainable params: 0
```

EVALUATION

- Utilized data set of August 2014 Lichess games where both players had Elo ratings of at least 2400
- 9,069 Total Moves Assessed

Algorithm	% of Moves Matched
Random Move	5.99%
Minimax Algorithm	6.04%
Neural Network Model	9.00%

APP DEMO



Find Your Next Chess Move

Set Up Your Chessboard:

Board Settings:

Set To Start Position | Clear Board

Player Selection:

White | Black

Recommendation Options:

Random | Minimax | Neural Network

CONCLUSIONS

- Creating an app changed the way I approached building out Python functions for the algorithms
- There is no perfect evaluation strategy for this particular problem
- There may not always be one “best” move in chess

THANK YOU!