

PYTHON PROJECT – V2 Report

AI Chess BOT

GROUP 13

Members –

Ayush Tankha

Jatin Singh

Peter Barnabas Keszthelyi

TIME TO PLAY!

In the final stage of our project, we write a snippet of code that allows the player to interact with the chess engine we created.

To initialize the game, one must choose some starting parameters like color of the pieces he/she wishes to play with and the depth of the engine to play against.

After choosing the initial parameters our chess game has the following UI interface –

```
Play as (type "b" or "w"): w
Choose depth: 3
r n b q k b n r
p p p p p p p p
. . . . .
. . . . .
. . . . .
. . . . .
P P P P P P P P
R N B Q K B N R
<LegalMoveGenerator at 0x7f9b89189e50 (Nh3, Nf3, Nc3, Na3, h3, g3, f3, e3, d3, c3, b3, a3, h4, g4, f4, e4, d4, c4, b
4, a4)>
To undo your last move, type "undo".
Your move: e4
r n b q k b n r
p p p p p p p p
. . . . .
. . . . .
. . . . P . . .
. . . . .
P P P P . P P P
R N B Q K B N R
The engine is thinking...
r n b q k b n r
p p p p . p p p
. . . . p . . .
. . . . .
. . . . P . . .
. . . . .
P P P P . P P P
R N B Q K B N R
```

Note – Our interface shows all possible chess moves in a certain position for the human so in case a person is not familiar with the proper square notation, he/she can refer to the values encoded in the “(...)” above.

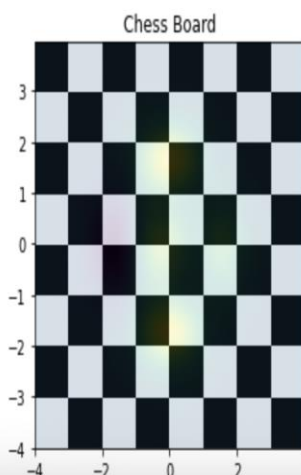
Additions that could not be made due to time constraints –

We wanted to create a real chess board using tkinter and importing the chess piece images while using pygame to run the application like a proper game. However, we faced many difficulties to create functions for each added piece on our own and fit the pieces appropriately in the chess squares by adjusting their height and width. This was not only time consuming but there was already a Chess library in the Python database which was much easier to use and we didn't have to code everything from scratch. Finally we decided to drop this idea and focus on our AI algorithm for the chess engine.

Below is a snippet of our futile attempt to create the chess game from scratch –

```
j: 1 import matplotlib.pyplot as plt
    2 import numpy as np
    3 from matplotlib.colors import LogNorm

j: 1 dp, dy = 0.015, 0.05
    2 p = np.arange(-4.0,4.0,dp)
    3 y = np.arange(-4.0,4.0,dy)
    4 P,Y = np.meshgrid(p,y)
    5 extent = np.min(p), np.max(p), np.min(y), np.max(y)
    6 z11 = np.add.outer(range(8),range(8))%2
    7 plt.imshow(z11,cmap = "binary_r", interpolation = "nearest", extent = extent, alpha = 1)
    8
    9 def chess(p,y):
   10     return (1 - p/2+p**5+y**6) * np.exp(-(p**2+y**2))
   11 z22 = chess(P,Y)
   12 plt.imshow(z22, alpha = 0.2, interpolation = "bilinear", extent = extent)
   13 plt.title("Chess Board")
   14 plt.show()
```



Limitations –

Our algorithm for the speed_chess bot can take a lot of time to evaluate positions after depth = 5. It is because of the exponential increase in decision trees and different cases for different chess position to choose from. The alpha-beta pruning helps to cut the time in the min-max algorithm but still this situation occurs simply due to the large number of possible positions in the game of chess.

Future scope –

There is always a better chess engine. Stockfish 8 and Leela Chess are some of the most advance chess engines out there and the research scientists are still improving on the algorithms of chess engines. So there is definitely scope for us to improve our algorithm so that we can have more depth in our chess engine moves. As of now we cannot really put an evaluation on the elo rating of our chess engine as of now but we hope someday we can do so and make it win against a real lower level chess engine.

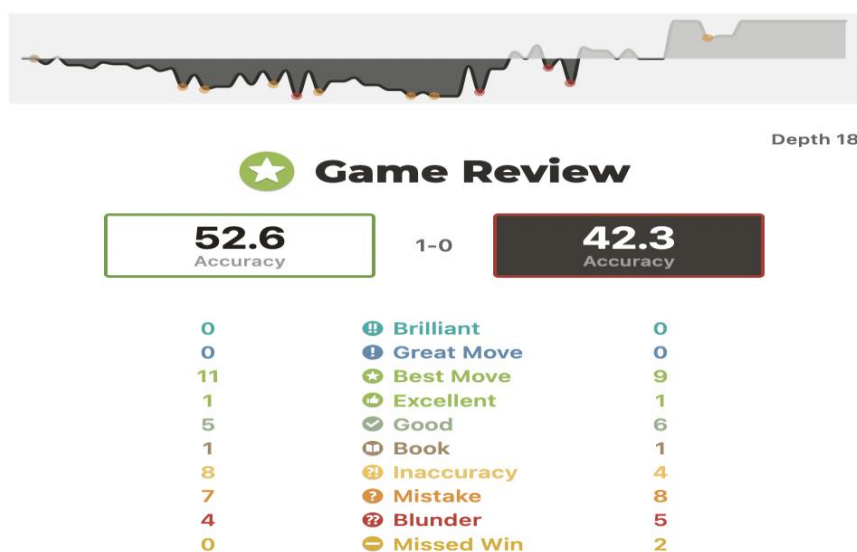
Testing our Speed_Chess Bot

So we manually played the Speed_Chess Bot (Depth = 4) moves against Jimmy the Bot on Chess.com. Jimmy the Bot has an elo rating of 600 and our Bot was able to defeat him in 37 moves with white pieces.

Key Observation –







Our bot made inaccuracies in certain moves but it never gave up material on immediate next move which means that the decision trees were correctly able to evaluate chess positions so as to not result in loss of a piece in the immediate next move. We can safely assume that at depth = 4 , our bot has at least an elo rating of 600.

Below is the summary of the game (you can follow the whole game in the Chess Bot Testing Folder) –



Here is some additional analysis from Chess.com
Speed_Chess (White) V/S Jimmy_bot (Black)

PERFORMANCE		WHITE	BLACK
Prediction		100%	0%
Accuracy		52.60	42.30
Best Move %		32.4%	27.8%
Avg Diff		1.30	2.10
Total Moves		37	36

PERFORMANCE BY PIECE		WHITE	BLACK
	Accuracy	86.80	40.80
	Moves	2 (5%)	5 (14%)
	Accuracy	-	22.10
	Moves	1 (3%)	1 (3%)
	Accuracy	31.10	92.30
	Moves	8 (22%)	2 (6%)
	Accuracy	52.20	70.90
	Moves	11 (30%)	9 (25%)
	Accuracy	71.00	38.10
	Moves	11 (30%)	14 (39%)
	Accuracy	47.90	-
	Moves	4 (11%)	5 (14%)