***The Final Report for Bing Visual Search Skill: The EndGame***

**What are we doing?**

We are developing a visual skill that will aid the user to find the next approximately 2-3 best moves to play based on the current chess board situation. The user needs to simply take a photo of the current board state and identify if they are playing as black or white, and the program will output on a simulated chessboard what the next three best moves are.

**Why are we doing it?**

We are making this visual search skill as we are both enthusiasts of Chess. Henry and I have been playing the game casually for years and have just started a Chess Club at our own school. Chess is one of the world’s most popular and renowned board games, and thus will have a large audience for our visual search skill. There’s a large skill gap present in Chess, and we hope that this skill will allow those beginners to begin seeing the bigger picture and become better at chess.

**Competitors & Differentiation**

One of the main competitors to our skill would be a similar function present on the website Chess.com. However, the main distinction between our skill and Chess.com’s is ease of use. In Chess.com, the user would need to put in all the previous moves that both they and their opponent has played for the program to determine the next best move. However, in our program one needs to simply take a picture of the current state, allowing for a much easier use and quicker desired result.

**Goals**

1. Be able to identify the edges of the chess board
2. Correctly recognize each piece from a top down view
3. Create the current board state in a simulated environment (using words for each piece on the grid)
4. Calculate the 2-3 best possibilities for the next move with correct priority given to Checkmate
5. Correctly show the possibilities to the user and their lines.
6. Correctly reset back to home screen.

**Non-Goals**

1. Don’t need to recognize if they’re playing as black or white (we will just ask)

2. Don’t need to show the board itself, or the pieces the way they are because we will just show the simulated board.

3. Will Ask for Confirmation about the current board state and can be changed to meet the user’s actual board state if some pieces are misidentified (I.e. don’t need to technically have perfect identification).

4. Not needing to make the skill useable for every chess piece, we will train on the pieces that are used for official chess tournaments, which is the most standard set. Cause even if we mess up, look at #3.

5. Don’t identify the orientation of the board – assume it’s always the correct orientation (from white’s perspective).

**Key Metrics of Success**

* Accuracy in detecting which piece is which on each square
* Returning the moves predicted by Stockfish

**How will it work? (High Level)**

1. Detect the edges of the board
2. Cut out the square board from the picture
3. Break the picture into 64 pictures by dividing into 8x8 squares, each with a piece on them, break along uniform lines.
4. Load each square into customvision model to detect what piece it is.
5. Transpose that piece on to a virtual board (8x8 grid).
6. Plug the virtual board into Stockfish.
7. Get whether the user is black or white.
8. Return “score advantage” to show whether white or black has an advantage at current game state.
9. Return the best 3 possibilities for the next move by Stockfish and display their lines as well.

**Timeline**

* 1/18 – collect training data for each piece
* End of January – correctly identify each piece on the board, and display board virtually
* / - Figure out how to get user to replace each piece, ask for whether black or white
* Return stockfish next moves and display their lines

**User Experience in detail**

1. User would open the skill
2. On the home screen, there would be an appealing logo, along with 2 main buttons: 1. Upload the photo, to begin using the skill and 2. A tutorial button that will explain every step either in text or video form.
3. When uploading the photo, there will be some sort of confirmation to make sure that the board has all its edges viewable along with no visual clearly blocking the pieces on the board (top down view).
4. After having uploaded the photo, the user will be prompted to fill in some parameters. The first will be what color they are playing as, which is essential to determine the next best moves and evaluate the “score advantage”, and the second will be to confirm the board state.
5. The board state will be depicted in a simulated chess board matching that of the current state with words representing the pieces instead of the piece itself. The coordinates will also be shown on each axis. For example, a row:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Rook | Knight | Bishop | King | Queen | Bishop | Knight | Rook |

1. After confirming the board state, the score difference will be shown, and the next three moves will be depicted on the simulated board after having gotten the next three best moves from Stockfish. The coordinates of the chess pieces that are moving will also be shown (I.e. A3 to B5).
2. End Screen will likely prompt the user back to the home screen so they may take another photo, maybe display some interesting statistics taken from Stockfish about the current situation here as well (likelihood to win??).

**Technical Design**

* Architecture
* How you will build a model
* Training data collection
* Model Metrics

**Dependencies**

* External Data sources (from pictures)
* Bing Visual Search
* Stockfish (Chess Engine)

**Risks (are there any?)**

1. Doesn’t translate the current board state correctly to the simulated one, which leads to the user not using the skill as they must input every time and correct the board that the program automatically generated.

**Open issues**

None