*FULL NAME: SOWRAB CHOWDHURY | CANDIDATE NUMBER:3417*

*AI-based chess training game.*

Brampton Manor Academy | 13202

Table of Contents

[Analysis 2](#_Toc18001559)

[Project background 2](#_Toc18001560)

[Current solution 2](#_Toc18001561)

[Relevant research 2](#_Toc18001562)

[End user and target audience 2](#_Toc18001563)

[Methods and sources 2](#_Toc18001564)

[Objectives 2](#_Toc18001565)

[Proposed solution 2](#_Toc18001566)

[Modelling 2](#_Toc18001567)

[Design 3](#_Toc18001568)

[High level overview 3](#_Toc18001569)

[Sketches 3](#_Toc18001570)

[Pseudocode 3](#_Toc18001571)

[Libraries 3](#_Toc18001572)

[Technical Solution 4](#_Toc18001573)

[System overview 4](#_Toc18001574)

[Key code 4](#_Toc18001575)

[Complex programming techniques 4](#_Toc18001576)

[Defensive programming 4](#_Toc18001577)

[Coding style 4](#_Toc18001578)

[Testing 5](#_Toc18001579)

[Evaluation 6](#_Toc18001580)

[Objectives 6](#_Toc18001581)

[End user and target audience 6](#_Toc18001582)

[Areas for improvement 6](#_Toc18001583)

[Conclusion 6](#_Toc18001584)

[Appendices 7](#_Toc18001585)

[Annotated code 7](#_Toc18001586)

# Analysis

## Project background

Chess is a world-renowned board game that still today appears from small friendly competitions, to world championships, such as the likes of Gary Kasparov. Although chess may be a game that is accessible for everyone, the game gives the challenge of: planning strategy, prediction and sacrifice to players, and mastering it would prove them as being skilled individuals. But what if we took out a human part of this problem? Could a machine, through mathematical computations, be a solution to the problem that 2 human players must be present for a physical game of chess to be playable?

Hence the idea of my project aims to solve this problem. Firstly, it aims to create, through machine learning, a computer-controlled chess player that can play against a human opponent. Secondly, my product will aim to challenge chess players physically, rather than through game apps. This will be achieved by implementing the AI onto a robotic arm, which will position its pieces after the AI has taken its own move. The idea of the robotic setup is to avoid players from being spoiled in the game. For example, in chess game apps, the program will often tell players where each of his pieces can move, and remove moves that would directly result into risking players losing due to checkmates. However, in a physical game of chess, these advantages are not given to players, hence the product is a more suitable way of practicing this game, as it aims to be similar to a physical game of chess.

to challenge beginning chess players, in terms of moves available, in a way that mimics a physical chess game. This can be achieved by initially restricting the user from being shown the moves available when a piece is clicked. Instead, can only see the moves as it is being shown in a normal board. The program will notify the user if an invalid move is made. Despite the restrictions, the user also has the choice to see which moves are available to make on the display after clicking a piece and after pressing a key. Similarly, this will bring me to my third aim, in which the player can decide to use the “Helper” function. The function will show the user the best move the chosen piece can take through calculations the are similar to how the AI chooses, but in a way that aids the user into making the move to increase its chance of winning.

Although an algorithm such as Minimax can potentially perform better than this AI, the problem comes in the fact that the game may become too difficult for beginners to play against. Also, the method of machine learning ensures that a difficulty, specified by the player can be selected unlike making an algorithm that is fixed to its own difficulty. This will both challenge experienced players and invite new ones to the game of chess.

## Current solution

When talking about the AI, as stated above, there are currently multiple apps where players can challenge themselves against AI through different difficulties. For example, one popular app for Android is called “Chess 2018”, where users can play against each other locally through the same phone, or play against a computer-controlled chess player which Is divided into 3 different difficulties. However, when talking about the physical component of my project, current solutions exist within final-year projects from university students. This would often be built with different approaches, such as through a simulation on the original computer.

In terms of apps to play chess games, current products take advantage of the fact that this game is not played physically. This in turn eases the player when facing the decision of moving chess pieces by showing moves available just after choosing a piece. Display-wise, I will be using Pygame to create a display for my program, along with its game-over screens with reasons and a menu where the user can set its initial parameters,

## Relevant research

Before I began my project, I wanted to know how to build my AI chess player. I started by looking at different machine learning algorithms, such as support vector machines or random forests. However, the most suitable to its purpose tends to be a neural network, such as Multi-layer perceptron (MLP). The purpose of this is not just to produce a probability for each setup, but also allow parameters to be modified, such as the number of layers, or the number of neurons for each layer. This will allow the user to change the AI’s difficulty within the game. To program the AI, we would also need to set out rules of chess to prevent it from making moves outside its list of available moves. Although it is necessary, I found a python library called “python.chess”, containing move generators, validators and its own chess board engine to play the game. Through this, I would be able to generate a list of available moves for each player, and therefore allowing the AI to choose the best move from the list given.

During my research, I also came across search algorithms that can be implemented onto my code to produce the best decision in its turn. These often include “MiniMax” or Alpha-Beta pruning (which is a method that reduces the number of nodes produced by the MiniMax algorithm). However, as stated above, I wanted my project to be built with a changing difficulty selection. Which will be limited through this method, since I would be forced to evaluate each move though a score rather than letting the program do it by itself.

The inspiration to create this project came through Deepmind’s Alphazero project, where the method of creating is similar to it. For example, the AI was trained through 30 million moves from games recorded on the internet, and by playing 500 matches against other existing AI chess players. This then lead to beating a respected chess champion Lee Sadol. Therefore, my method would intend to create an AI that Is similar to Deepmind’s chess player, but training the AI with a reduced training set(to reduce memory and time needed to train the AI).

## 

## End user and target audience

My target audience will be individuals who are currently beginners towards the game, especially players who do not understand the rules of chess or the role of each piece. As stated from the project background, my product does not aim to tell the user which moves are available, but rather validates if the move is correct. My product is also aimed for those who want to challenge themselves in a physical game of chess, but unable to do so because other people may be unavailable.

## Methods and sources

To create my AI, I would need to train it through a collection of chess games, and label them each as either win, draw or losses.

To do this, I found collections of chess game datasets in “lichess.org game database”, which can be found here <https://database.lichess.org/>. I would then turn each game (stored in PGN format), and convert it into a series of FEN lines, each labelled to its corresponding white-win, black-win, or draw. After this, each FEN line will be converted into a 64 numbers long array, which represents the 8x8 setup of the board and its pieces. This would be used to train my AI-chess player into predicting the probability for each player to win or to draw.

I will also make use of the python library “python-chess”, which can be downloaded form here(<https://pypi.org/project/python-chess/>). This library will contain its own board, move validator and move generator, allowing me to gain a list of possible moves for the AI’s turn, and also create a copy of the board in the original computer, where the AI can look at the board setup.

## Objectives

My main objectives are:

-To Train my AI with a series of labelled FEN lines from the “Lichess” datasets.

-To create an AI that decides which move to take from a given list of possible moves depending on the setups they produce.

-To give the user a selection of difficulties before starting the game, therefore choosing from saved models of the same AI, but with different set of parameters.

-To give the robotic arm a way of recognising each position of the chess board from the camera.

-To give the robotic arm the ability to move each piece to its designated location.

-To validate each move the user makes, and notifying if any move is incorrect.

-To create a display of the chess board using Pygame.

-Give the user the ability to move pieces by clicking pieces from one position to the other.

-Creating new screen displays when the game finishes, followed with a text on the screen with a reason (such as checkmate by user).

-To create a “Helper” button, which will show the user the possible squares that the chess pieces can move to on the game display

-To create a function that uses the AI to determine which move is best to use by the user’s chosen piece.

-To give the user multiple tries to choose the right move instead of closing the program.

-Creating a program that will allow me to save AI models into a text file using a library called pickle.

-To create 2 versions of the AI of the same difficulty, one which plays as a black player and the other as a white player.

-To create a menu where the user is given the choice of difficulty and whether to play as a black player or a white player, and also notifying how the program is used.

-To allow the user to restart the game within the match and after the game has ended.

## Proposed solution

My Robotic chess player will be controlled by the in-built AI, which will make decision about which moves to take when it is their turn. For example, when the AI’s turn comes, the AI will create a copy of the board after each move possible move is selected, and calculate the probability based on this setup. Once the move with the best probability has been found, the AI will describe its coordinates and move each piece to its location.

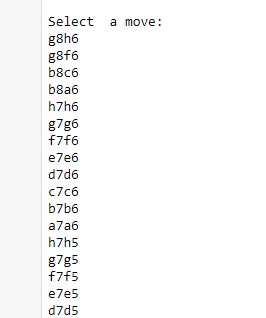
To check if the human player is done with its turn, a camera from above will take a screenshot of the current board setup and compare it to its previously saved setup(which is taken when the robotic arm finished its turn). If the new board is valid, then the robotic arm will process its decision, then move the chosen piece to its designated coordinates. However, if the move done by the human player is incorrect, an LED will flash notifying the player of the mistake.

## Modelling

The following slides would demonstrate how this will work:



By taking an example that the white pieces are controlled by the AI and the black pieces are the user’s, the AI player will first decide which move to take based on this setup. Once it is chosen, the move is executed and the setup will change.





In this case, the computer-controlled chess player has chosen to move its piece from e2 to e4 as shown in the figure above. This would therefore be represented in the format “e2e4” from the program when selecting the best move to take.

# Design

## High level overview

## Sketches

## Pseudocode

## Libraries

# Technical Solution

## System overview

## Key code

## Complex programming techniques

## Defensive programming

## Coding style

# Testing

## Test plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test #** | **Purpose** | **Test data** | **Expected outcome** | **Actual outcome** |
|  |  |  |  |  |
|  |  |  |  |  |

## Testing evidence

# Evaluation

## Objectives

## End user and target audience

## Areas for improvement

## Conclusion

# Appendices

## Annotated code