

---

# DEVELOPMENT OF A CHESS AI PLAYER

---

By Kuzey Cimen



APRIL 30, 2021

UNIVERSITY OF ESSEX

Supervisors: Prof. John Gan, Dr. David Richerby

## Abstract

This project aims to use a Neural Network to teach an AI player (Artificial Intelligence), from a database of super grandmaster (>2700 rating) games.

### *Add AI Techniques and project goals*

The main goal is to train an AI player until it was strong enough to beat an average player in a chess game.

Evaluating the chess games that will be used to teach the AI using a different version of Minimax, Negamax. Minimax is an algorithm that checks every possible move and decides which one is the best using an evaluation function. Negamax is a more efficient approach to minimax. Evaluation is done by examining the move made from the current state to the next state.

An online platform is available to play against the Negamax AI player or the Trained AI player which uses a Neural Network. There is also an option to make the two AI players play against each other. In the future, the next step would be to implement user accounts, a leaderboard and a Player vs Player mode.

This report will cover the research that has been done, the background of the project, the design, the planning and implementation choices that were made and future work.

## Acknowledgement

I would like to say a special thank you to my supervisor, Professor John Gan. His support, guidance and overall insights in this field have made this an inspiring experience for me.

I would also like to thank my second supervisor Doctor David Richerby and my friends for helping me discover ideas for the features that I implemented.

Finally, I would like to thank my family for supporting me during the compilation of this dissertation.

## Table of Contents

Abstract .....	1
Acknowledgement.....	1
1 Introduction .....	5
2 Literature Review and Background Research .....	6
2.1 Mechanical Turk Chess AI .....	6
2.2 1950s Chess AI .....	6
2.3 1970s Chess AI .....	7
2.4 1980s Chess AI .....	8
2.5 Deep Blue vs Garry Kasparov - 1990s Chess AI .....	8
2.6 Modern Day Chess AI .....	8
3 Design.....	9
3.1 Chess .....	9
3.1.1 What is Chess.....	9
3.1.2 Rules of Chess.....	9
3.1.3 Chessboard .....	10
3.3 Programming Language.....	10
3.4 Chessboard Representation.....	10
3.5 The User Interface.....	11
3.6 Data Flow.....	12
3.7 Algorithms.....	13
3.8 Methods of Increasing Playing Strength .....	13
4 Implementation and Experiments .....	13
4.1 Minimax .....	13
4.2 Alpha-Beta Pruning .....	13
4.3 Iterative Deepening .....	14
4.4 Negamax .....	14
4.5 Transposition Table.....	14
4.6 Square Values.....	15
4.7 Neural Network .....	18
4.8 Platform .....	18
4.8.1 Python .....	18
4.8.2 JavaScript.....	18
4.8.3 Python-chess.....	19
4.8.4 Chessboard.js.....	19
4.8.2 JQuery .....	19

4.8.5 Flask .....	19
4.9 Online Implementation .....	19
4.9.1 Heroku .....	19
4.9.3 CSS, Layout and Colour Choices.....	19
4.9.4 User Experience.....	20
4.9.5 User Interface .....	20
4.9.8 Human-Computer Interaction .....	20
4.10 Technical Documentation .....	20
4.10.1 File Structure.....	21
4.10.2 Files Folder.....	22
4.10.3 Games Folder .....	23
4.10.4 Static Folder .....	23
4.10.5 Templates Folder.....	24
4.10.6 .gitignore .....	24
4.10.7 Procfile .....	24
4.10.8 README.md .....	24
4.10.9 app.py .....	25
4.10.10 board.py .....	28
4.10.11 config.py .....	28
4.10.12 entry.py .....	29
4.10.13 evaluate.py .....	29
4.10.14 learn.py .....	30
4.10.15 Requirements.txt.....	31
4.10.16 Runtime.txt .....	31
4.10.17 script.bat .....	31
4.10.18 script.js.....	31
4.10.19 script.sh .....	31
4.10.20 transposition_table.py.....	32
4.10.21 weights.py.....	33
4.10.22 weightHandler.py .....	33
4.10.23 style.css.....	34
5 Testing and Evaluation.....	36
5.1 Human Player vs Default AI Player Tournaments.....	36
5.2 Human Player vs Neural AI Player Tournaments.....	36
5.3 Neural AI Player vs Default AI Player Tournaments.....	36
5.4 Test Results.....	36

6 Findings and Technical Achievements.....	37
6.1 Quantity of Work Done.....	37
6.2 Quality of Work Done.....	37
7 Project Planning and Management.....	38
7.1 Agile Development.....	38
7.2 Gitlab .....	38
7.3 Momentum .....	39
7.4 Adapting to Change and Progress .....	39
7.5 Cumulative Flow Diagram .....	39
7.6 Bugs & Risks.....	39
8 Conclusion .....	40
8.1 Future Work .....	40
8.1.1 Accounts.....	40
8.1.2 Leaderboard.....	40
8.1.3 Multiple Bots .....	40
8.1.4 Ranking System.....	40
References .....	41
Appendices .....	43
Weekly Work Done.....	43

## 1 Introduction

The main goal and scope of this project is to train an AI player until it was strong enough to beat an average player in a chess game. The plan for the project was first using minimax to start and then developing and training a neural network for the end goal of the project. The inspiration for this project came from my love of the game of chess and AI. I wanted to make a project on what I loved the most and to teach myself more about AI. I want to make a user interface so that any user can play against my developed AI. The intended audience for this application is any age, I want to show people what AI is capable of and how it will affect our future lives.

## 2 Literature Review and Background Research

The idea of creating a chess-playing machine has been around for centuries. Although early automatons, such as the notorious Mechanical Turk built-in 1770, were merely hoaxes, genuine chess algorithms were available as early as 1948.

### 2.1 Mechanical Turk Chess AI

The Turk, also known as the Mechanical Turk or Automaton Chess Player, was a late-eighteenth-century chess-playing machine.

It was displayed as an automaton by various owners from 1770 until its destruction by fire in 1854, though it was ultimately revealed to be an elaborate hoax.

Wolfgang von Kempelen (1734–1804) designed and unveiled the mechanism in 1770 to impress Empress Maria Theresa of Austria. It appeared to be capable of playing a good game of chess against a human opponent, as well as performing the knight's tour, a puzzle that allows the player to move a knight to occupy every square of a chessboard exactly once. [16]



### 2.2 1950s Chess AI

Dietrich Prinz, Turing's colleague, wrote the first true, automated chess programme in 1951. Prinz's software ran on Manchester University's new Ferranti Mark I machine, and although it couldn't play a full game of chess due to memory and computational limitations, it was able to solve the "mate-in-two" problem: find the best move when you're two moves away from checkmate. Seven years later, on the new IBM 704 mainframe, an IBM researcher named Alex Bernstein wrote the first complete, fully automated chess-playing programme.

### 2.3 1970s Chess AI

Chess-playing programmes began to evolve in the 1970s, thanks to a combination of more efficient hardware and improved software.

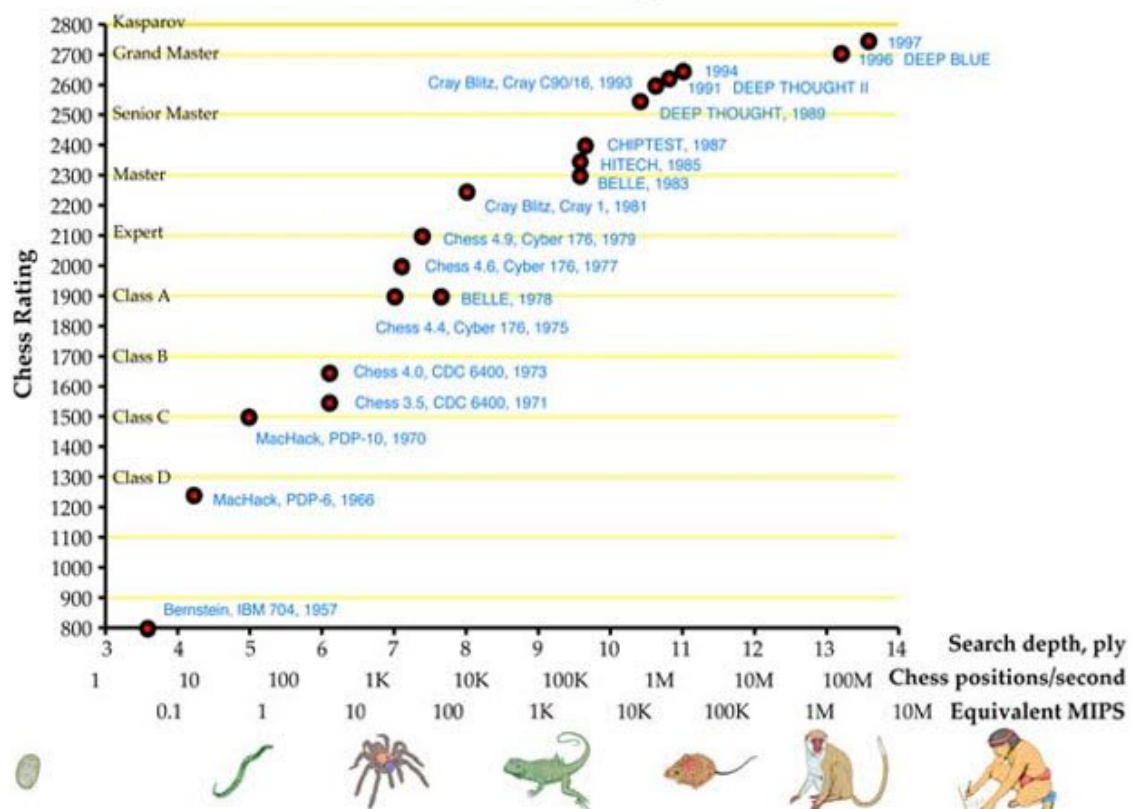
To supplement existing systems, effective heuristics were introduced, as well as creative search enhancements such as "iterative deepening."

Rather than setting a fixed depth, MiniMax used iterative deepening to progressively increase the depth of the search.

With the limited time available during the game, the technique allowed chess programmes to optimise their search strategies each turn. [16]

Year	Computer Chess Rating (best fit)	Human Percentile
1950		0%
1955		0%
1960	1201	49%
1965	1400	61%
1970	1599	74%
1975	1797	87%
1980	1996	95%
1985	2194	98%
1990	2393	100%
1995	2592	100%
2000	2790	100%
2005	2988	100%
2010	3187	100%

### Chess Machine Performance versus Processing Power



[16]



## 2.4 1980s Chess AI

The personal computer era began in the 1980s.

Amateur programmers started experimenting with and playing against chess programmes in their living rooms on their own time when the Apple II, TRS-80, and Commodore PET all arrived on the personal computer market.

Chess-playing systems were also marketed to the general public as stand-alone products.

The Fidelity Electronics Chess Challenger only played amateur chess, but it was a common consumer product at the time.

Nobody knew what computers could do in the end, but by the 1980s, it was clear that they could play chess very well.

## 2.5 Deep Blue vs Garry Kasparov - 1990s Chess AI

With the sole goal of beating the reigning World Chess Champion, Garry Kasparov, IBM purchased Deep Thought and formed a chess programming group headed by the original Carnegie Team who founded it. In 1989, Kasparov easily defeated Deep Thought. The Deep Thought system (now called Deep Blue after IBM's nickname) made its return in 1997, with updated software and major hardware upgrades. Deep Blue could evaluate 200 million chess positions per second, which was unheard of at the time.

## 2.6 Modern Day Chess AI

Today, too, much is known about computer chess play, as hundreds, if not thousands, of previous matches, have been analysed, and so-called endgame databases containing moves and counter-moves in various endgame positions of prior matches, as well as various search and heuristic techniques proven effective in various systems over the years, are also available. Scientists like Claude Shannon and Herbert Simon, who invented Minimax and Alpha-Beta Pruning to make computer chess possible in the first place, are still important today, as modern developers continue to use the basic Minimax architecture with Alpha-Beta pruning, albeit with hardware that was unimaginable in Shannon's time.

## 3 Design

### 3.1 Chess

#### 3.1.1 What is Chess

Chess is an abstract strategy game in which no detail is secret. It is played on a 64-square square chessboard with an eight-by-eight grid. Each player controls sixteen pieces at the start (one player controls the white pieces, the other player controls the black pieces): one king, one queen, two rooks, two knights, two bishops, and eight pawns. The game aims to checkmate the opponent's king, which is under immediate attack (in "check") and cannot be removed from the attack on the next move. This is done by putting the opponent's king in a position where there is not a possibility of it escaping, either from moving or defending using other pieces. A game can also end in a tie in a variety of ways. [1]

#### 3.1.2 Rules of Chess

##### Pieces

**King** - Moves exactly one square horizontally, vertically, or diagonally. A special move, called castling, is allowed only once per player, per game. If a king moves before castling, that king is no longer allowed to castle. Also, if a rook moves before castling, the king is no longer allowed to castle towards that side.

**Rook** - Moves any number of vacant squares horizontally or vertically. It also is moved when castling.

**Bishop** - Moves any number of vacant squares diagonally.

**Queen** - Moves any number of vacant squares horizontally, vertically, or diagonally.

**Knight** - Moves to the nearest square, not on the same rank, file, or diagonal. A knight move consists of a single horizontal and a single vertical move, forming an "L" shape. The knight is not blocked by other pieces meaning that it jumps to the new location.

##### **Pawns - most complex rules of movement**

A pawn has the most complex rules of movement.

A pawn moves straight forward one square if that square is vacant. If that pawn has not yet been moved, that pawn also has the option of moving two squares straight forward, provided both squares are vacant. Pawns cannot move backwards.

A pawn, unlike other pieces, captures differently from how it moves. A pawn can capture an enemy piece on either of the two squares diagonally in front of the pawn (but cannot move to those squares if they are vacant).

A pawn is also involved in the two special moves en passant and promotion.

[2]

### 3.1.3 Chessboard

A chessboard is a type of gameboard that is used when playing chess. It is made of 8 by 8 squares. Each square of the board can be identified using chess notation, algebraic, or numeric. Each horizontal row of squares is called a rank, a letter from "a" to "h", and each vertical column of squares is called a file, a number from 1 to 8. Each oblique line of squares of the same colour is called a diagonal. A square can be defined with a rank followed by a file, e.g. "a1" or "h8". [3]

### 3.3 Programming Language

Using python as the development language for this project was the most suitable as it includes useful libraries which are helpful for artificial intelligence development.

### 3.4 Chessboard Representation

The chessboard representation is controlled and handled by the python-chess library in the back end. Some of the uses are described below:

Position of each piece.

Which player's turn is it to move.

State of the pieces (en passant, castled, etc).

Returning all legal moves for all pieces.

If a king is in check.

If the king is pinned by another opponent's piece.

The captured piece on after the last move if any.

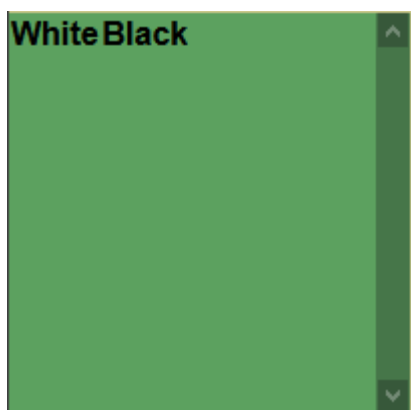
"From Square" and "To Square" on that current board state.

### 3.5 The User Interface

The below image is the chessboard for the project.

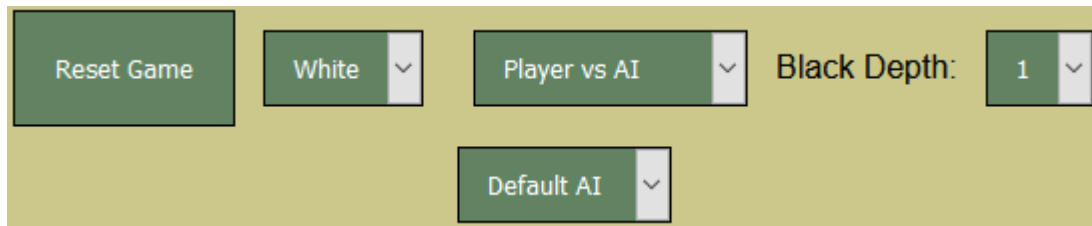


The below image is the list of moves that are made by each player.



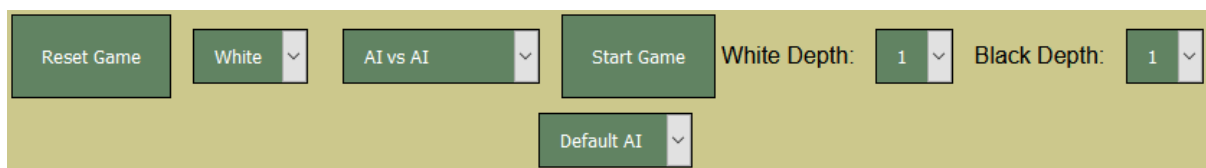
The below images are for the menu of the chessboard. As the user changes the game type they want to play; Player vs AI, AI vs AI and Player vs Player, the menu dynamically changes.

#### Player vs AI



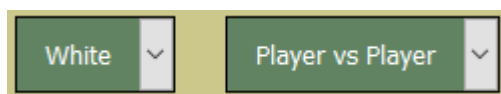
The interface for 'Player vs AI' features a light green background. It includes a 'Reset Game' button on the left. To its right is a 'White' dropdown menu. Further right is a 'Player vs AI' dropdown menu. To the right of this is the text 'Black Depth:' followed by a dropdown menu showing the value '1'. Below these elements is a 'Default AI' dropdown menu.

#### AI vs AI



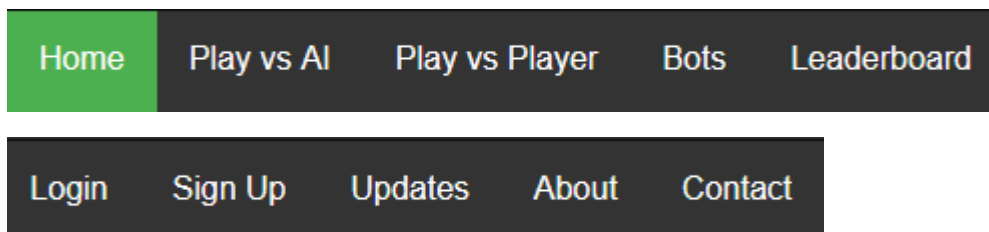
The interface for 'AI vs AI' has a light green background. It contains a 'Reset Game' button, a 'White' dropdown menu, an 'AI vs AI' dropdown menu, and a 'Start Game' button. To the right of the 'Start Game' button are two depth settings: 'White Depth:' with a dropdown showing '1', and 'Black Depth:' with a dropdown showing '1'. A 'Default AI' dropdown menu is positioned below the 'Start Game' button.

#### Player vs Player



The interface for 'Player vs Player' is shown on a light green background. It consists of a 'White' dropdown menu and a 'Player vs Player' dropdown menu.

The navigation bar for the application



The navigation bar is divided into two sections. The top section has a dark background with five links: 'Home' (highlighted in green), 'Play vs AI', 'Play vs Player', 'Bots', and 'Leaderboard'. The bottom section also has a dark background with five links: 'Login', 'Sign Up', 'Updates', 'About', and 'Contact'.

The user interface for the project is a website that is hosted online. The deployment service I use is Heroku.

### 3.6 Data Flow

Data flows through the system by using JavaScript, HTML, Python and Flask. JavaScript is the frontend and Python is the backend. The user input that has been done on the website is sent to Python with the aid of Flask and then all the validation is done in python. The response is sent back to JavaScript so the user can see the updated state.

### 3.7 Algorithms

In the implementation of Artificial Intelligence, using Negamax was the most approachable option, as well as, Transposition Table with Zobrist Hashing.

### 3.8 Methods of Increasing Playing Strength

There were a few options that could be chosen to increase the AI Player playing strength. It was either use a neural network with machine learning or use a Negamax with a lot of efficiency methods.

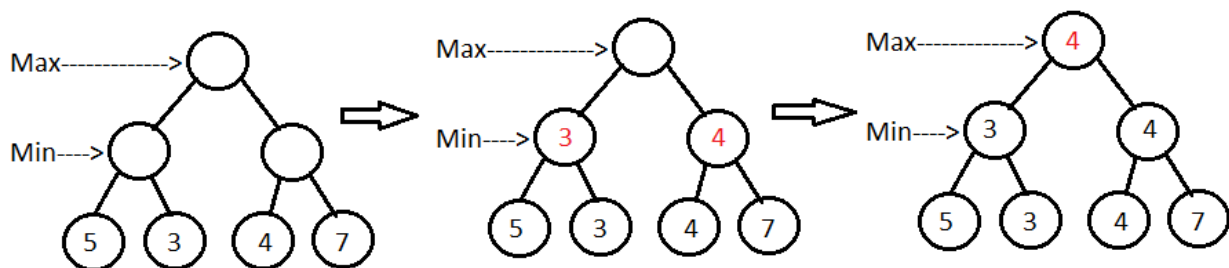
## 4 Implementation and Experiments

### 4.1 Minimax

Minimax is a decision rule that is used in artificial intelligence, decision theory, game theory, statistics, and philosophy to minimise the maximum loss in a worst-case scenario. When dealing with gains, the term "maximin" refers to maximising the smallest gain possible.

It has now been applied to more complex games and general decision-making in the face of ambiguity. It was originally developed for n-player zero-sum game theory, covering both the situations where players take alternative moves and those where they make simultaneous moves. [4]

A simple tree looks like the picture below.



[15]

### 4.2 Alpha-Beta Pruning

Alpha-beta pruning is a search algorithm that aims to reduce the number of nodes in its search tree that are evaluated by the minimax algorithm. It's an adversarial search algorithm that's widely used to play two-player games by machines (Tic-tac-toe, Chess, Go, etc.). It stops assessing a move when it discovers at least one possibility that proves it is worse than a previously considered move. Such actions do not need to be re-evaluated. When applied to a regular minimax tree, it produces the same result as the minimax, but prunes out branches that have no bearing on the final decision. [5]

Alpha is the best value that the maximiser currently can guarantee at that level or above. Beta is the best value that the minimizer currently can guarantee at that level or above.

#### 4.3 Iterative Deepening

Negamax is the iterative deepened version of minimax. Usually, minimax uses recursion to look through depths, however, iterative deepening, also known as Negamax, uses iteration. Recursion meaning that it calls the function inside the function. Iterative meaning that it uses a for loop or a while loop.

#### 4.4 Negamax

Negamax search is a form of minimax search that takes advantage of a two-player game's zero-sum property. This algorithm relies on the fact that  $\text{max}(a, b) = -\text{min}(-a, -b)$  to make the minimax algorithm easier to implement. In such a game, the value of a place to player A is the inverse of the value to player B. As a result, the player on the move looks for a move that maximises the negation of the value resulting from the move: the opponent must have valued this successor position by definition.

If A or B is moving, the logic in the previous sentence holds. This implies that both positions can be valued using a single method. This is a coding simplification over minimax, which allows A to choose the move with the highest-valued successor and B to choose the move with the lowest-valued successor. [6]

In the implementation of Negamax for this project, the current board state is the node that is used in the searching and the evaluation process.

#### 4.5 Transposition Table

Transposition Table is a way of remembering past board states which speed up the searching and evaluation process for Negamax. It uses entries as nodes and Zobrist hashing to hash the board states and save them into a dictionary.

In a game tree created by a computer game playing programme, a transposition table is a cache of previously seen positions and associated evaluations. If a position recurs after a different series of moves, the value of the position is retrieved from the table instead of re-searching the game tree below it.

Transposition tables are most useful in games with perfect details (where the entire state of the game is known to all players at all times). Transposition tables are a form of dynamic programming that is basically memoization applied to the tree quest. [11]

Hash tables are often used to implement transposition tables, with the current board position as the hash index. The number of possible positions in a game tree is proportional to the depth of quest, and can range from thousands to millions or even more. As a result, transposition tables can consume the

majority of available system memory and account for the majority of the memory footprint of game-playing programmes. [11]

#### 4.6 Square Values

For the Negamax algorithm, there needs to be some sort of evaluation for each square. Therefore, I have added a value for each square for each piece. This process is of course variable in the neural network since the machine learning process needs to learn from the beginning without any knowledge.

##### **Pawn**

0	0	0	0	0	0	0	0
5	10	10	-20	-20	10	10	5
5	-5	-10	0	0	-10	-5	5
0	0	0	20	20	0	0	0
5	5	10	25	25	10	5	5
10	10	20	30	30	20	10	10
50	50	50	50	50	50	50	50
0	0	0	0	0	0	0	0

##### **Knight**

-50	-40	-30	-30	-30	-30	-40	-50
-40	-20	0	5	5	0	-20	-40
-30	5	10	15	15	10	5	-30
-30	0	15	20	20	15	0	-30
-30	5	15	20	20	15	5	-30
-30	0	10	15	15	10	0	-30
-40	-20	0	0	0	0	-20	-40
-50	-40	-30	-30	-30	-30	-40	-50



### Bishop

-20	-10	-10	-10	-10	-10	-10	-20
-10	5	0	0	0	0	5	-10
-10	10	10	10	10	10	10	-10
-10	0	10	10	10	10	0	-10
-10	5	5	10	10	5	5	-10
-10	0	5	10	10	5	0	-10
-10	0	0	0	0	0	0	-10
-20	-10	-10	-10	-10	-10	-10	-20

### Rook

0	0	0	5	5	0	0	0
-5	0	0	0	0	0	0	-5
-5	0	0	0	0	0	0	-5
-5	0	0	0	0	0	0	-5
-5	0	0	0	0	0	0	-5
-5	0	0	0	0	0	0	-5
5	10	10	10	10	10	10	5
0	0	0	0	0	0	0	0

### Queen

-20	-10	-10	-5	-5	-10	-10	-20
-10	0	5	0	0	0	0	-10
-10	5	5	5	5	5	0	-10
0	0	5	5	5	5	0	-5
-5	0	5	5	5	5	0	-5
-10	0	5	5	5	5	0	-10
-10	0	0	0	0	0	0	-10
-20	-10	-10	-5	-5	-10	-10	-20

### King

20	30	10	0	0	10	30	20
20	20	0	0	0	0	20	20
-10	-20	-20	-20	-20	-20	-20	-10
-20	-30	-30	-40	-40	-30	-30	-20
-30	-40	-40	-50	-50	-40	-40	-30
-30	-40	-40	-50	-50	-40	-40	-30
-30	-40	-40	-50	-50	-40	-40	-30
-30	-40	-40	-50	-50	-40	-40	-30

### King End Game

-50	-30	-30	-30	-30	-30	-30	-50
-30	-30	0	0	0	0	-30	-30
-30	-10	20	30	30	20	-10	-30
-30	-10	30	40	40	30	-10	-30
-30	-10	30	40	40	30	-10	-30
-30	-10	20	30	30	20	-10	-30
-30	-20	-10	0	0	-10	-20	-30
-50	-40	-30	-20	-20	-30	-40	-50

King end game is different as the king needs to move to the centers to control move squares since there are not much major pieces left on the board such as rooks, bishops, knights and a queen.

#### 4.7 Neural Network

For the Neural Network implementation, I partially applied another developer's code. [14] The parts I have used are the learning file and the config file. Although, making some changes and additions according to the project was needed.

The neural network is stored in a text document. It stores the data, moves, tactics and techniques that is learned from the super grandmaster games to the text file called weights.txt. This process is explained in 4.10.

#### 4.8 Platform

##### 4.8.1 Python

Python was used to develop the application because it had many modules and libraries that could help the project.

##### 4.8.2 JavaScript

JavaScript was the most suitable for the project because of its dynamic usability. The application uses JS to display the board and every action that can be made on the website.

#### 4.8.3 Python-chess

I used python-chess for the main game system and features. The library handles piece movements, castling, en passant, move legality, board position checks such as if the game is over, etc.

#### 4.8.4 Chessboard.js

Chessboard.js is a user interface tool that displays a chessboard with many features such as images and smooth movement of the pieces as well as getting and setting the positions.

#### 4.8.2 JQuery

JQuery was needed for Chessboard.js to work as it uses its functions. This is also a more efficient way to use JavaScript as it reduces the code that needs to be written and improves stability.

#### 4.8.5 Flask

Flask is a web framework that allows the developer to use the REST API to send data between the front-end and the back-end of a system efficiently. I am using Flask to read the users actions on the website, front-end, movement on the chessboard, to send it to Python, back-end. These actions are verified and played on the main chessboard in the back-end and then sends a response back to the front-end.

### 4.9 Online Implementation

#### 4.9.1 Heroku

To publish the application online I have used a service called Heroku. This service allows the compilation of the project which uses Python, JavaScript and Flask into a presentable web application.

The web application can be found on this link:

<https://ce301-chess.herokuapp.com/>

The first time the website loads, after it has not been used for a long time, the user needs to wait for the servers to activate the URL in the backend.

#### 4.9.3 CSS, Layout and Colour Choices

The Layout choices which was made were the most suitable for the game to be hosted online. The largest object being the chessboard and the navigation bar at the top guides the user to play.

#### 4.9.4 User Experience

Making the user experience was as important as developing the AI. The piece movements, AI speed and an application without bugs would lead to a better user experience.

#### 4.9.5 User Interface

The user interface of the website is colour matching. The menu at the top is the navigation bar that guides the user through the website. So far, there are pages for each tab in the navigation bar with manually filled in information. In the future, this would change to a more dynamic and interactive approach that will do what the tab was supposed to do, to begin with.

#### 4.9.8 Human-Computer Interaction

An essential part of this project was the human-computer interaction. Making sure it was as smooth as possible by front-end and back-end validations on each movement, animation on the pieces and the computer playing at the right time for each colour makes a difference.

#### 4.10 Technical Documentation

In this section, I will be explaining what each method and file does in the project.

When the application has started, the user is shown with a chessboard.



The user can change the configuration of the game in the menu below the chessboard. When the user makes a move as the white pieces, the frontend of

the system will send the data such as the move made to the backend for validation.

If the backend of the system validates the move, the user's chessboard is updated to the new chessboard state. Then, the AI Player, depends on the chosen AI, starts evaluating the board state and decides on a move.

This evaluation process time depends on the depth chose. This application is currently maxed and 4 depths. A depth is how far the AI Player will look into the future for all possible game states. After the evaluation, the chessboard in the backend is first updated, then the chessboard in the frontend is updated. The user will see the updated state and decide on the next move.















#### 4.10.1 File Structure

To keep the files organized in the development environment using a structured way of keeping the files as required. Flask's file structure [12] helped create the file arrangement.

```
/home/user/Projects/flask-tutorial
├── flaskr/
│   ├── __init__.py
│   ├── db.py
│   ├── schema.sql
│   ├── auth.py
│   ├── blog.py
│   └── templates/
│       ├── base.html
│       ├── auth/
│       │   ├── login.html
│       │   └── register.html
│       └── blog/
│           ├── create.html
│           ├── index.html
│           └── update.html
├── static/
│   └── style.css
├── tests/
│   ├── conftest.py
│   ├── data.sql
│   ├── test_factory.py
│   ├── test_db.py
│   ├── test_auth.py
│   └── test_blog.py
├── venv/
├── setup.py
└── MANIFEST.in
```










## 4.10.2 Files Folder

The files folder includes all the non-coding files and non-essential files. There are files like academic papers, PowerPoints, Summaries, Helping documents and files for other submissions for this project.

Name	Last commit	Last update
..		
 Assessing Game Balance with Alp...	Organize folder structure.	2 months ago
 CE301 Week 11.pptx	Organize folder structure.	2 months ago
 CE301_Kuzey_Cimen_Abstract.docx	Update Poster and Abstract.	1 month ago
 CE301_Kuzey_Cimen_Final_Repor...	Update Final Report.	6 hours ago
 CE301_Kuzey_Cimen_Poster.pptx	Update Report, update NN writing and readi...	1 month ago
 Challenge Week.pptx	Organize folder structure.	2 months ago
 Mastering Chess and Shogi by Se...	Organize folder structure.	2 months ago
 Mastering Chess and Shogi by Se...	Organize folder structure.	2 months ago
 Research.docx	Organize folder structure.	2 months ago
 Summary.docx	Organize folder structure.	2 months ago
 alphazero summary.pdf	Organize folder structure.	2 months ago
 chess_image.PNG	Organize folder structure.	2 months ago
 project-report-help.pdf	Organize folder structure.	2 months ago
 setup.txt	Organize folder structure.	2 months ago









#### 4.10.3 Games Folder

The games folder includes all the professional games that a super grandmaster has played in their lifetime up to when these files were downloaded. The games are downloaded from this site: <http://www.pgnmentor.com/files.html>

Name	Last commit	Last update
..		
 Aronian.pgn	Add more games.	2 months ago
 Carlsen.pgn	Add PGN games of chess super grandmasters.	2 months ago
 Caruana.pgn	Add more games.	2 months ago
 Ding.pgn	Add more games.	2 months ago
 Fischer.pgn	Add more games.	2 months ago
 Karpov.pgn	Add more games.	2 months ago
 Kasparov.pgn	Add more games.	2 months ago
 Nakamura.pgn	Add more games.	2 months ago
 So.pgn	Add more games.	2 months ago

#### 4.10.4 Static Folder











The static folder includes all the static files that are being used in this project. Such as chessboard.js files, CSS files, JQuery files,

Name	Last commit	Last update
..		
 dist	Implement AI vs AI and fix bugs.	4 months ago
 img/pieces	Use flask, make a webpage, integrate chessb...	7 months ago
 chessboard.css	Use flask, make a webpage, integrate chessb...	7 months ago
 chessboard.js	Use flask, make a webpage, integrate chessb...	7 months ago
 jquery.min.js	Use flask, make a webpage, integrate chessb...	7 months ago
 js.cookie.mjs	Add cookies so chessboard is not lost when ...	7 months ago
 script.js	Convert history of moves to a table, fix upda...	1 month ago
 style.css	Fix minor CSS bug in move history table.	1 month ago



#### 4.10.5 Templates Folder

This folder is necessary for flask to work and function. It includes all the html files that the web application will use.

Name	Last commit	Last update
..		
 about.html	Fix small html bug.	1 month ago
 ai.html	Fix small html bug.	1 month ago
 bots.html	Fix small html bug.	1 month ago
 contact.html	Fix small html bug.	1 month ago
 index.html	Convert history of moves to a table, fix upda...	1 month ago
 leaderboard.html	Fix small html bug.	1 month ago
 login.html	Fix small html bug.	1 month ago
 player.html	Fix small html bug.	1 month ago
 register.html	Fix small html bug.	1 month ago
 updates.html	Fix small html bug.	1 month ago

#### 4.10.6 .gitignore

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/.gitignore](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/.gitignore)

This file is used in Git. File paths written in this file are ignored by git and will not be pushed to the git repository.

#### 4.10.7 Procfile

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/c7186ab807e871904b7181a04357d37679a63aa8/Procfile](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/c7186ab807e871904b7181a04357d37679a63aa8/Procfile)

Procfile is a file that is used by Heroku to publish the application online. It includes which python file to run for the application to work.

#### 4.10.8 README.md

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/README.md](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/README.md)

README file is used in my project to display what libraries my project uses, prerequisites, installation, deployment, online website, usage of the website, roadmap, authors, acknowledgements and contact information. There is a table of contents at the top of the README file to guide the reader.

4.10.9 app.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/app.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/app.py)

This python file is where the main application is run at the start. It uses Flask to create functions which will be used by both the backend and the frontend. An example function:

**@app.route('/')**

**index():**

This function is used display the home page of the application. It is displayed when the website is visited.

**@app.route('/about')**

**about():**

This function is used to return the about page to the frontend so the about html file is shown when the user clicks the about tab.

This webpage holds information about the project, AI and the web application.

**@app.route('/ai')**

**ai():**

This function is used to return the ai page to the frontend so the ai html file is shown when the user clicks the ai tab.

This webpage holds information about the AI types used in this game.

**@app.route('/contact')**

**contact():**

This function is used to return the contact page to the frontend so the contact html file is shown when the user clicks the contact tab.

This webpage holds information about the contact information of the university and me.

**@app.route('/bots')**

**bots():**

This function is used to return the bots page to the frontend so the bots html file is shown when the user clicks the bots tab.

### **@app.route('/leaderboard')**

#### **leaderboard():**

This function is used to return the leaderboard page to the frontend so the leaderboard html file is shown when the user clicks the leaderboard tab.

This webpage holds placeholders for the leaderboard which will be implemented in the future.

### **@app.route('/player')**

#### **player():**

This function is used to return the player page to the frontend so the player html file is shown when the user clicks the player tab.

This webpage holds placeholders for playing against a player in realtime which will be implemented in the future.

### **@app.route('/login')**

#### **login():**

This function is used to return the login page to the frontend so the login html file is shown when the user clicks the login tab.

This webpage holds placeholders for logging in to the account which will hold information about the games played, ranking and account information.

### **@app.route('/register')**

#### **register():**

This function is used to return the register page to the frontend so the register html file is shown when the user clicks the register tab.

This webpage holds placeholders for registering an account which will hold information about the games played, ranking and account information.

### **@app.route('/updates')**

#### **updates():**

This function is used to return the updates page to the frontend so the updates html file is shown when the user clicks the updates tab.

This webpage will hold information about the updates made and will be made to the game.

### **@app.route("/reset")**

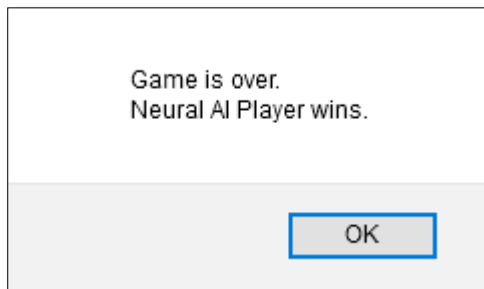
#### **reset():**

This function resets the board. It is used when the page is refreshed, a configuration in the menu is changed or the user pressed the reset button.

### **@app.route('/is\_game\_over')**

#### **is\_game\_over():**

This board checks if the game is over and returns the result to the front end. If the game is over the user is shown with a game over screen and then the user is no longer able to make a move as the pieces are locked.



### **@app.route('/current\_board\_state')**

#### **current\_board\_state():**

This function returns the current board state to the front end so the user can see the updated state of the board live.

### **@app.route('/past\_moves')**

#### **past\_moves():**

This function is used to return the past moves that are made by each player to the front end for it to be displayed on the moves list which is next to the chessboard.

### **@app.route('/move')**

#### **def move():**

This function is used to read the user's move and apply the move onto the backend. If the move is validated, the computer move function is called. The board is then updated to show the user the updated chessboard state.

**@app.route('/selfplay')**

**def self\_play\_move():**

This function is used to make the two AI Players play against each other. It could be Neural AI Player vs Neural AI Player, Negamax AI Player vs Negamax AI Player or Neural AI Player vs Negamax AI Player.

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

This if statement creates a new flask application.

This function uses @app.route to forwards the past moves that have been made in that specific chess game to the front end for it to be displayed on the past move list.

4.10.10 board.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/board.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/board.py)

This python file is used to create the chessboard on which the game will be played on. It also has a function for the computer move which calls the evaluation function in the evaluator class in evaluate.py.

The initializer function " \_\_init\_\_()" for the class is used then the class is called and an object is created. After this happens the class checks, using its parameters, if there is an existing chessboard in the backend. If there is an existing chessboard, the function uses that chessboard. If not, It will create a new chessboard for the game to be played on.

The "comp\_move()" function is called when it is computers turn to make a move. This function will call the Negamax function in the evaluator class, get the final decided move and return it for it to be displayed on the frontend.

4.10.11 config.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/config.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/config.py)

This file is used to configure the parameters of the neural network.

The parameters are below with their description.

MAX\_ITER\_MTD - Maximum number of iterations in MTD.

MAX\_DEPTH - Maximum depth of Negamax for searching.

PAWN\_SCORE - Material score of a single pawn.

MAX\_POSITION\_SCORE - Maximum score awarded in piece square tables.

MAX\_SCORE = PAWN\_SCORE \* 1 - Maximum score that is awarded in evaluation.

ALPHA\_INIT = 0.001 - Initial learning rate.

ALPHA\_DEC\_FACTOR = 1.0001 - Learning rate gets divided by this after every game.

LAMBDA = 0.7 - Temporal decay factor.

MAX\_GAMES = 10 - Number of games to learn from.

GAMES\_FILE\_NAME - Filename to import the games for training.

4.10.12 entry.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/entry.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/entry.py)

This file contains a class that is used as an object in the transposition table.

This object contains useful information such as the value of the current board state that is returned by the evaluation function, the flag -----, the depth of the board from the starting position and the move that is made to get to the current board state. These useful parameters help the AI player improve the efficiency of deciding on making a move.

It is called when there needs to be a new entry to the transposition table, a new chessboard state has been seen.

4.10.13 evaluate.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/evaluate.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/evaluate.py)

This file is the class for the evaluation of a position in the given board state.

This class is responsible for the main evaluation for the AI Player. It will include neural network calculations and algorithms such as Negamax.

```
__init__(self, max_iterations=config.MAX_ITER_MTD, max_search_depth=config.MAX_DEPTH, max_score=config.MAX_SCORE, evaluator="negamax"):
```

This function is the initializer for the evaluator class. This will decide which evaluator function to run. It creates the transposition table, and the square values, heat maps, for each piece, discussed in 4.6.

**negamax(self, board, depth, alpha, beta, move, evaluator):**

This function is the negamax function. It is used for the Negamax AI Player.

**\_mtd(self, board, depth, firstGuess):**

This function is the wrapper for the neural network evaluator function. It starts the evaluation process for the neural network.

**selectMove(self, board):**

This function is used to select a move from the board and it will return the score of that move and the state of the board after that move.

**def clearTranspositionTable(self):**

This function is used for clearing the transposition table. It helps the neural network start a new transposition table after each game.

**negamax\_evaluate(self, board):**

This function is the evaluation function for the negamax AI Player.

**nn\_evaluate(self, board):**

This function is the evaluation function for the neural network AI Player.

**findFeatures(self, board, color):**

This function detects and calculates how good that move is after the neural network chooses a move.

4.10.14 learn.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/learn.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/learn.py)

This file is used in the learning process for the neural network part of the application.

main():

This function is the wrapper for the learning process of the neural network. It starts the games, counts moves, updates weights text file by calling the weights handler class. After the training is done, number of games decided by the config file, it closes the weights handler.

**learn(wRawInit, wRawFin, fInit, fFinal, J, alpha, lambdaDecay, clampVal):**

This function is used in the learning process of the neural network. Unrolling parameters into vector, calculate update amount (with sign) for parameters, update parameters, rolling parameter vector and return final weights.

4.10.15 Requirements.txt

This text file contains all the needed modules and libraries which is used in the application.

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/requirements.txt](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/requirements.txt)

4.10.16 Runtime.txt

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/runtime.txt](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/runtime.txt)

This text file is for Heroku. It tells Heroku which python version to run for the application to work online.

4.10.17 script.bat

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/script.bat](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/script.bat)

This file is used as an initializer for the project for windows operating systems. It is used for creating the virtual environment, activating it, and installing the required modules and libraries which are written in requirements.txt.

4.10.18 script.js

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/script.js](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/script.js)

This file contains the script for the application. It has many functions such as getting the board orientation, history of the piece moves by each player, board position of the chessboard, if the game is over or not and creating the game board.

Additional functions include on piece drop action and on piece drag start.

4.10.19 script.sh

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/script.sh](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/script.sh)



This file is used as an initializer for the project for Linux operating systems. It is used for creating the virtual environment, activating it, and installing the required modules and libraries which are written in requirements.txt.

4.10.20 transposition\_table.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/transposition\\_table.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/transposition_table.py)

This file contains a class with functions for the transposition table.

- Storing and looking up entries for the current board state.
- Generating a Zobrist hash for a board state that has not been stored.
- Zobrist hash function for creating the hash for a board state.

After a user makes a move the AI Player starts the searching and evaluation process. In that searching and evaluation process, every new chessboard state it sees is saved into the dictionary in the Transposition Table class.

#### **\_\_init\_\_(self):**

This function is the initializer for the transposition table class.

A “key” dictionary is initialized by calling the gen\_zobrist\_keys().

A “table” dictionary is initialized every time a new chessboard state has been seen.

#### **store(self, state, value, flag, depth, move):**

This is the store function for the transposition table. The saving process is done by looking at each square, getting the pieces key-value from the generated keys integer and getting the key according to that piece on that square.

#### **lookup(self, state):**

This is the lookup function for the transposition table class. This function looks up if the chessboard state is in the table dictionary. If the AI player encounters the same chessboard state, it gathers its Entry object in the dictionary value, which was generated in the storing process of that chessboard state, which includes the depth, the move made, value (evaluation score) and the flag.

#### **gen\_zobrist\_keys(self):**

A Transposition Table is initialized with integer keys generated between 0 and maximum integer (9223372036854775807) for each piece for each square. Therefore, generating a long code with a length of 768 digits. This function

returns the long code to the class variable for it to be used later. This process is only done once when the transposition table is initialized.

### **zobrist\_hash(self, state):**

After all the keys are gathered in that chessboard state. Those individual keys are then XORed together to generate a unique has for that specific chessboard state.

4.10.21 weights.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/weights.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/weights.py)

This file contains the weights of the neural network nodes. This data is then used by the game to recall the best moves in that current board position.

4.10.22 weightHandler.py

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/weightsHandler.py](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/weightsHandler.py)

This class is responsible for editing the weights python file. It includes setters and getters.

### **\_\_init\_\_(self, file\_name):**

This function is the initializer for the weights handler class. If the text document for the neural network exists, it opens it and sets the local variables same as the text file.

This is for if I wanted to train the AI at a further time, it can save its current game position and learned data so it does not have to begin from the start again.

If the text document does not exist, this initializer function will create an empty neural network.

### **writeWeightsToFile(self):**

This function writes the initial position, final position and the game into a text file for it to be read later.

### **closeWeightsFile(self):**

This function closes the weights file.

**setGameNum(self, game):**

This function sets the game number for the current game that is being played by the trainer.

**getWeights(self):**

This function gets the weights from the the class variables.

**getGameNum(self):**

This function returns the game number that is being learned by the trained

4.10.23 style.css

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/static/style.css](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/bc3bf6bba83c8b4401764f34598d31a93bf5082b/static/style.css)

This file is the only CSS file for the application. It is used to make the website look better to increase user experience.

About.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/about.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/about.html)

AI.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/ai.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/ai.html)

Bots.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/bots.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/bots.html)

Contact.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/contact.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/contact.html)

Index.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/index.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/index.html)

Leaderboard.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/leaderboard.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/leaderboard.html)

Login.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/login.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/login.html)

Player.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/player.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/player.html)

Register.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/register.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/register.html)

Updates.html

[https://cseegit.essex.ac.uk/ce301\\_2020/ce301\\_cimen\\_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/updates.html](https://cseegit.essex.ac.uk/ce301_2020/ce301_cimen_kuzey/-/blob/6c60612f5633c6efe8d8f104b242abe760abbeac/templates/updates.html)

## 5 Testing and Evaluation

To make sure that the developed chess AI player was good enough, no bugs or risks, testing and evaluation was needed. The Negamax AI will be called the "Default AI Player" and the neural network AI will be called "Neural AI Player".

### 5.1 Human Player vs Default AI Player Tournaments

A total of 10 games are played. In 5 of the games, the Human Player was the white pieces and the other 5 of the games it was the black pieces.

### 5.2 Human Player vs Neural AI Player Tournaments

A total of 10 games are played. In 5 of the games, the Human Player was the white pieces and the other 5 of the games it was the black pieces.

### 5.3 Neural AI Player vs Default AI Player Tournaments

A total of 10 games are played. In 5 of the games, the Neural AI Player was the white pieces and the other 5 of the games it was the black pieces.

### 5.4 Test Results

#### **Test 1 - Human Player vs Default AI Player Tournaments**

The human player won 8/10 games against the Default AI Player.

The human player was better than the Default AI Player in these games. A larger sample of games and a larger sample of human players were needed to get a better conclusion out of this test.

#### **Test 2 - Human Player vs Neural AI Player Tournaments**

The human player won 6/10 games against the Neural AI Player.

The human player was arguably better than the Neural AI Player. The Neural AI Player needs more time, computing power and more games to beat an average human player confidently. Neural AI Player scored better than the Default AI Player.

#### **Test 3 - Neural AI Player vs Default AI Player Tournaments**

The Neural AI Player won 4/10 games against the Default AI Player.

The Default AI player was arguably better than the Neural AI Player. This is largely because there were not enough time to train the Neural AI Player. Default AI player will choose the best move in that position.

## 6 Findings and Technical Achievements

### 6.1 Quantity of Work Done

I believe that the quantity of work done in this project during the given time line was excellent. I accomplished my project goals and aims, although there is still area for improvement for the game and the AI Player.

### 6.2 Quality of Work Done

I believe that the quality of work done in this project was welldone. The application is hosted online, it uses front end and back end systems and two AI Players were developed with their own algorithms.

## 7 Project Planning and Management






















### 7.1 Agile Development

At the start of this project, I planned before starting any coding. Selected the papers I wanted to read [13], how I wanted to start the project and how I will manage my time on further implementations.

I used Kanban to organise my work. Creating tasks, updating them and selecting them as done was useful to make sure the project stayed on track.

### 7.2 Gitlab

Gitlab was an essential part of this project. Using it as a cloud saving, backup and tracking changes to the project was helpful.

Name	Last commit	Last update
 files	Update Final Report.	20 hours ago
 games	Add more games.	2 months ago
 static	Fix minor CSS bug in move history table.	1 month ago
 templates	Convert history of moves to a table, fix upda...	1 month ago
 .gitignore	Update evaluation function and update requi...	5 months ago
 Profile	app.py	1 month ago
 README.md	Update README.md	19 hours ago
 app.py	Update file formats.	4 days ago
 board.py	Update file formats.	4 days ago
 config.py	Update file formats.	4 days ago
 entry.py	Implement negamax using transposition tabl...	4 months ago
 evaluate.py	Update Final Report.	1 day ago
 learn.py	Update file formats.	4 days ago
 requirements.txt	Numpy	1 month ago
 runtime.txt	Heroku first commit	1 month ago
 script.bat	Make a script file for CERES/linux and update...	5 months ago
 script.js	Fix all menu features (selecting game types), ...	1 month ago
 script.sh	Make a script file for CERES/linux.	5 months ago
 transposition_table.py	Export transposition table and update pip m...	2 months ago
 weights.py	Update Report, update NN writing and readi...	1 month ago
 weightsHandler.py	Update file formats.	4 days ago

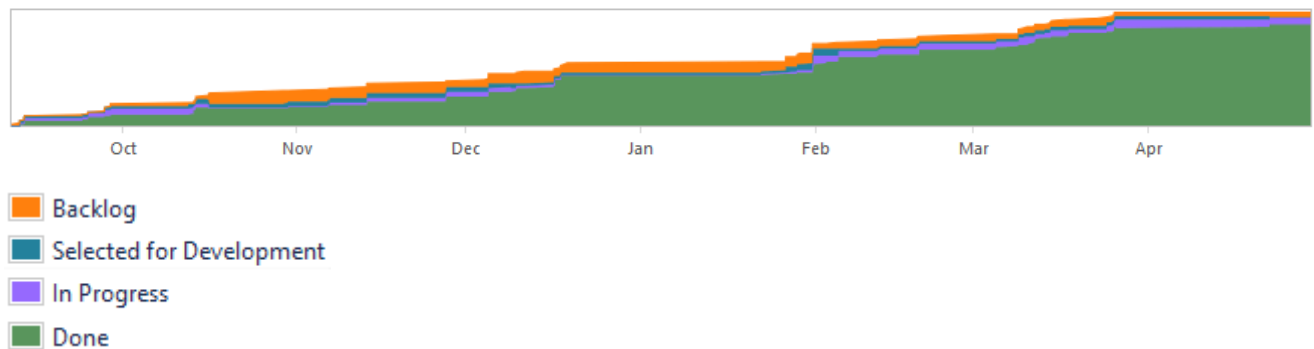
### 7.3 Momentum

Maintaining the momentum and adapting to change in this project was crucial in achieving a satisfying final product. Every week I added my tasks on Jira that I will complete during that week. This helped me stay on track and sustained my motivation to develop this project.

### 7.4 Adapting to Change and Progress

There were many changes to the system during the development. An example of this would be that I initially wanted to implement Monte Carlo Tree Search for the heuristic searching algorithm of the system. However, I decided not to do that because of how new it was and not enough resources were available on the internet. Therefore, I decided to continue with improving minimax with efficiency and accuracy methods.

### 7.5 Cumulative Flow Diagram



This diagram shows the tasks in the backlog, selected for development, in progress and done on Jira. It displays the cumulative progression of my project.

### 7.6 Bugs & Risks

At some moments in the project, there were issues with bugs and risks. An example of this would be that the frontend and the backend timing was slow, and the user could move the pieces when the AI player was calculating its move. This was fixed by delaying the user's movement in the backend of the application.

There are still existing bugs in the system. An example of this would be

Risk in the system could be that a user encounters a bug in the application that I did not know of.



## 8 Conclusion and Reflection

In this project, I have implemented Minimax, Negamax, iterative deepening, transposition table, Zobrist hashing, Neural Network, web architecture, front end and back end systems. I have developed a website that the user can interact with to play the game of chess against an AI. The user also has the choices to play vs a player on the same computer or make the two AI players, Negamax AI Player and Neural Network AI Player with each other. At the beginning I was planning on developing just an AI Player without a user interface, however I found that a website would be the most suitable.

### 8.1 Future Work

In the future, for this project, adding new features such as user accounts, a leaderboard, multiple Bots and a ranking system would increase the quality of the application and improve the satisfaction of the user.

#### 8.1.1 Accounts

I want the user to be able to create an account to save played games and add their name to a leaderboard.

#### 8.1.2 Leaderboard

I want there to be a leaderboard so that every user can compete against other players. To do this there would first need to be accounts in the application.

#### 8.1.3 Multiple Bots

I want to create multiple difficulty bots that the user can choose to play against to their liking. This will make it so a user can have it at their current level.

#### 8.1.4 Ranking System

I want there to be some level of competitiveness when playing against other players. There would be ranked games where players that play the game seriously can play.

## References

- [1]  
Wikipedia Contributors, “Chess,” *Wikipedia*, Feb. 23, 2019.  
<https://en.wikipedia.org/wiki/Chess>.
- [2]  
Wikipedia Contributors, “Rules of chess,” *Wikipedia*, Nov. 19, 2019.  
[https://en.wikipedia.org/wiki/Rules\\_of\\_chess](https://en.wikipedia.org/wiki/Rules_of_chess).
- [3]  
Wikipedia Contributors, “Chessboard,” *Wikipedia*, Mar. 29, 2021.  
<https://en.wikipedia.org/wiki/Chessboard>.
- [4]  
Wikipedia Contributors, “Minimax,” *Wikipedia*, Sep. 16, 2019.  
<https://en.wikipedia.org/wiki/Minimax>.
- [5]  
Wikipedia Contributors, “Alpha-beta pruning,” *Wikipedia*, Mar. 06, 2021.  
[https://en.wikipedia.org/wiki/Alpha-beta\\_pruning](https://en.wikipedia.org/wiki/Alpha-beta_pruning).
- [6]  
Wikipedia Contributors, “Negamax,” *Wikipedia*, Oct. 20, 2020.  
<https://en.wikipedia.org/wiki/Negamax>.
- [7]  
Flask Team, “Welcome to Flask — Flask Documentation (1.1.x),” *flask.palletsprojects.com*.  
<https://flask.palletsprojects.com/>.
- [8]  
C. Oakman, “chessboardjs.com» Homepage,” *chessboardjs.com*. <https://chessboardjs.com/>.
- [9]  
N. Fiekas, “python-chess: a chess library for Python — python-chess 1.5.0 documentation,” *python-chess.readthedocs.io*. <https://python-chess.readthedocs.io/en/latest/>.
- [10]  
Chessprogramming, “Transposition Table - Chessprogramming wiki,” *www.chessprogramming.org*. [https://www.chessprogramming.org/Transposition\\_Table](https://www.chessprogramming.org/Transposition_Table).
- [11]  
Wikipedia Contributors, “Transposition table,” *Wikipedia*, Apr. 12, 2020.  
[https://en.wikipedia.org/wiki/Transposition\\_table](https://en.wikipedia.org/wiki/Transposition_table).
- [12]  
Flask Team, “Project Layout — Flask Documentation (1.1.x),” *flask.palletsprojects.com*.  
<https://flask.palletsprojects.com/en/1.1.x/tutorial/layout/> (accessed Apr. 21, 2021).
- [13]  
D. Silver *et al.*, “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,” *arXiv.org*, 2017. <https://arxiv.org/abs/1712.01815>.
- [14]  
JHurricane96, “JHurricane96/chessai,” *GitHub*, Apr. 14, 2018.  
<https://github.com/JHurricane96/chessai/tree/09e3ae5eaab76c0a5d70d30e2cd6252b5f24d8aa>  
(accessed Apr. 22, 2021).
- [15]  
baeldung, “Introduction to Minimax Algorithm,” *Baeldung*, Jul. 11, 2017.  
<https://www.baeldung.com/java-minimax-algorithm>.
- [16]  
Staff Writers, “A Brief History of Computer Chess,” *TheBestSchools.org*, Jul. 20, 2015.  
<https://thebestschools.org/magazine/brief-history-of-computer-chess/>.

[17]

Chessprogramming, “Simplified Evaluation Function - Chessprogramming wiki,”

*www.chessprogramming.org*.

[https://www.chessprogramming.org/Simplified\\_Evaluation\\_Function](https://www.chessprogramming.org/Simplified_Evaluation_Function) (accessed Apr. 28, 2021).

## Appendix

### Weekly Work Done

#### **Week 1**

Fresher's Week

No work was done as the project has not yet started.

#### **Week 2**

Challenge Week

In this week I have done background research for my project and planned on my project goals and aims.

#### **Week 3**

<https://cseejira.essex.ac.uk/browse/A301132-28>

In this week I have done background research for my project and planned on my project goals and aims.

#### **Week 4**

<https://cseejira.essex.ac.uk/browse/A301132-29>

In this week I have researched about Monte Carlo Tree Search.

#### **Week 5**

<https://cseejira.essex.ac.uk/browse/A301132-30>

In this week I have implemented Monte Carlo Tree Search.

#### **Week 6**

<https://cseejira.essex.ac.uk/browse/A301132-32>

In this week I succeed in the implementation of Monte Carlo Tree Search however found out it was slower than minimax search with alpha-beta pruning.

#### **Week 7**

In this week I have researched improving the efficiency of the minimax search with alpha-beta pruning.

### **Week 8**

<https://cseejira.essex.ac.uk/browse/A301132-36>

In this week I have tried another implementation of Monte Carlo Tree Search since I did not want to give up on this heuristic search algorithm.

### **Week 9**

<https://cseejira.essex.ac.uk/browse/A301132-38>

This week I tried to find the bug and the fix for the Monte Carlo Tree Search.

### **Week 10**

<https://cseejira.essex.ac.uk/browse/A301132-45>

This week I have stopped implementing Monte Carlo Tree Search. I found out about Negamax and implemented it.

### **Week 11 - Interim**

<https://cseejira.essex.ac.uk/browse/A301132-51>

This week I created the PowerPoint I was going to present and fixed the final bugs.

### **Week 12**

Term Break Week 1

### **Week 13**

Term Break Week 2

### **Week 14**

Term Break Week 3

### **Week 15**

Term Break Week 4

### **Week 16**

<https://cseejira.essex.ac.uk/browse/A301132-53>

In this week I received my feedback and made improvements on those points.

### **Week 17**

<https://cseejira.essex.ac.uk/browse/A301132-60>

In this week I made improvements to the efficiency of Negamax and added heatmaps for the pieces.

### **Week 18**

<https://cseejira.essex.ac.uk/browse/A301132-69>

In this week I made improvements to the user interface for the website

### **Week 19**

### **Week 20**

<https://cseejira.essex.ac.uk/browse/A301132-74>

In this week I researched the implementation of neural networks with machine learning.

### **Week 21**

<https://cseejira.essex.ac.uk/browse/A301132-75>

This week I started the neural network process for the project.

### **Week 22**

<https://cseejira.essex.ac.uk/browse/A301132-76>

In this week I continued my implementation of the neural network.

### **Week 23**

<https://cseejira.essex.ac.uk/browse/A301132-82>

### **Week 24**

<https://cseejira.essex.ac.uk/browse/A301132-88>

**Week 25**

<https://cseejira.essex.ac.uk/browse/A301132-89>

**Week 26**

Term Break Week 1

**Week 27**

Term Break Week 2

**Week 28**

Term Break Week 3

**Week 29**

Term Break Week 4

**Week 30**

Final Week