Project Stage-II Report

ON

CHESS NEURAL NETWORK USING ARTIFICIAL INTELLIGENCE

BY

Mr. ROHAN MANDHARE **Mr. NIKHIL THAPA**

Mr. BANDU SAKHARE **Mr. PRAJWAL JADHAV**

Guide

Prof. Shwetkranti Taware

Department of Computer Engineering

Indira College of Engineering and Management

[2021 - 22]

SAVITRIBAI PHULE PUNE UNIVERSITY

.

INDIRA COLLEGE OF ENGINEERING AND MANAGEMENT

PARANDWADI, PUNE - 410506

<u>CERTIFICATE</u>

This is to certify that

Mr. Rohan Mandhare

Mr. Nikhil Thapa

Mr. Bandu Sakhare

Mr. Prajwal Jadhav

have successfully completed the project Stage-2 entitled "Chess Neural Network Using Artificial Intelligence" under my supervision in the partial fulfilment of Bachelor of Engineering – Computer Engineering of Savitribai Phule Pune University..

(Mrs. Shwetkranti Taware)
Project Guide

(Ms. Pragati Choudhari)
Project Coordinator

(DR. Soumitra Das)
HOD Computer Dept.

(Dr. Sunil Ingole)
Principal

(________________________)
External Examiner

Place: Pune

Date:_ _ / _ _ / _ _ _ _

.

Acknowledgment

We are presenting this project report on “Chess Neural Network Using Artificial Intelligence” as part of the curriculum of B.E. Computer Engineering. We have taken a lot of effort into making this project. However, it would not have been possible due to the kind support and help of all the people who endlessly supported us right from the beginning to the end. We would like to extend sincere thanks to all of them.

We express my sincere and profound thanks to all our teachers **Dr. Soumitra Das** (Head of Department), **Ms. Pragati Choudhari** (Project Coordinator). We would like to thank our college ICEM, Parandwadi, for the boost that it has provided. We wish to thank **Ms. Shwetkranti Taware**(Computer Dept.) for her student-like enthusiasm and her guidance from time to time. We heartily thank for all her help and valuable time. Her invaluable advice has helped us bring this work to completion. Besides, We take this opportunity to express my sincere gratitude to the Principal **Dr. Sunil Ingole**(ICEM, Pune) for providing a good environment and facilities to complete this project.

We would like to thank all our internal guides for providing the resources for project stage. And last but not least, all our friends, who have helped us directly or indirectly throughout the project.

Abstract

A chess neural network, a sort of artificial intelligence generally is a chess-playing program that studies the position deeply enough to generally find the absolute best moves in any given position at any point in time, which generally is quite significant. It integrates machine learning that can study and for all intents and purposes understand the gameplay style of a really particular pretty human player, which literally is quite significant. This indefinitely turn can benefit definitely top professionals who are preparing for a world cup match against their opponent in a fairly big way. The AI has a stimulation mode that can basically stimulate according to the playstyle of any very individual player in a definitely big way. This can basically happen if the games of that player essentially have fed properly. This kind of advanced feature allows professionals to generally play a practice really match against their virtual opponent with whom they are going to compete in real-time, fairly contrary to popular belief.

The basic goal of this project mostly is to literally remove the training barrier for professionals, which for the most part is quite significant. Even any for all intents and purposes individual who actually wants to specifically play against their favourite chess player can use the simulation mode and can really enjoy playing against his inspired player, so even any very individual who literally wants to particularly play against their favourite chess player can use the simulation mode and can for all intents and purposes enjoy playing against his inspired player, or so they for the most part though.

Keywords: *Recommendation, Machine Learning , Artificial intelligence , Neural Network*

Contents

List of Figures

List of Tables

Abbreviations

AI	:	Artificial Intelligence
PGN	:	Portable Game Notation
HLSL	:	High Level Shader Language
ES	:	Evaluation Score
KS	:	King Safety
D	:	Defenders
A	:	Attackers
P	:	Piece Placement Score

1 Introduction

Chess is a board game played between two players that simulates a war between two kingdoms. It is one of the most popular games in the world. Millions of people play it both recreationally and competitively. Chess is a turn-based strategy game with no hidden information. For this reason, the element of luck is virtually non-existent in the game. Chess is a complex game with many pieces and rules. To play chess, players need at least a set of chess pieces and a chessboard. A chess piece set has two different armies of pieces, each containing eight pawns, two knights, two bishops, two rooks, a queen, and a king. Players can distinguish their armies based on their colours, with light and dark pieces. Regardless of the actual colour of the pieces, the light side is called White, while the dark side is called Black—reminiscent of the ivory and ebony pieces used in the past.

A chess neural network is an artificial intelligence and a chess-playing program that studies the position deeply enough to find the best moves in any given position at any point in time. It uses machine learning to study the gameplay style of an opponent based on his/her games imported into an AI. Hence the AI can really be beneficial to the top professionals who want to prepare for the match against their opponents whom they already know.

Meanwhile, how to effectively and timely crawl all the tourists' social media information from the service providers is also challenging. Besides the social media data, sensor network data (e.g., Bluetooth data) and cellular data are also adopted by the researchers for tourist study, but they suffer from similar limitations and constraints.

The AI has a stimulation mode that can stimulate according to the playstyle of any individual player. The stimulation mode requires games of a player to be fed in an AI, so that AI can understand the repetitive patterns of that individual player and can help us to prepare for such an opponent. This kind of advanced feature allows professionals to generally play a practice match against their virtual opponent with whom they will compete in real-time. The basic goal of this project mostly is to literally remove the training barrier for professionals. Even any individual player who actually wants to play against their favourite chess player can use the simulation mode and can really enjoy playing against his inspired player.

1.1 Overview

Chess, one of the oldest and most popular board games, played by two opponents on a checkered board with specially designed pieces of contrasting colours, commonly white and black. White moves first, after which the players alternate turns in accordance with fixed rules, each player attempting to force the opponent's principal piece, the King, into checkmate—a position where it is unable to avoid capture. Chess is played on a board of 64 squares arranged in eight vertical rows called files and eight horizontal rows called ranks. These squares alternate between two colours: one light, such as white, beige, or yellow; and the other dark, such as black or green. The board is set between the two opponents so that each player has a light-coloured square at the right-hand corner.

1.2 Motivation

The use of Computers actually literally is increasing rapidly in the field of chess to scan the most difficult problems using sophisticated algorithms. Computers actually kind of has actually generally played an important and vital role in increasing the strength of a chess player in a subtle way, really contrary to popular belief. Advanced Technology improvements in chess will completely change the way we for the most part actually look into the game of chess, fairly actually contrary to popular belief, which mostly is fairly significant.

With definitely much kind of more than 1080 moves kind of pretty possible in chess, only one basically essentially is the hardly the literally the best move, definitely fairly further showing how kind of literally advanced Technology improvements in chess will completely change the way we mostly really look into the game of chess in a basically major way, which definitely is quite significant. The job of the AI literally kind of is to really basically find that move, showing how really advanced Technology improvements in chess will completely change the way we actually kind of look into the game of chess in a for all intents and purposes actually major way in a for all intents and purposes major way.

1.3 Problem Definition and Objectives

The biggest problem of chess players is that they are not able to improve their game. And while playing the game, they are not able to increase their rating. And for professionals, they spend a huge time preparing openings, middlegame plans and endgame techniques to be able to win against their opponents. Also, each individual who loves to play chess and wants to play against their favourite top-level chess player faces a lot of problems due to geographical reasons, or some other reasons. And if someone wants to play against an opponent from the past centuries, cannot play against him. For example, if someone wants to play players like Bobby Fischer, Alexander Alekhine, Mikhail Tal, Boris Spassky, Mir Sultan Khan, Napoleon Bonaparte, Jose Raul Capablanca, etc. So, there's a need and a demand for software that can eliminate the time gap when it comes to preparing a game plan against an opponent. And also, for entertainment purposes, if someone wants to play against their favourite player of all time, should be able to bring his dream to reality. The main objectives of the project are:

- To create an AI to scan millions of positions to find the best move.
- To create an AI that can self-adjust according to Human Thinking. The more games we feed, the better AI adjusts itself to Human Thinking.
- To help a player to increase his mental strength or IQ level.
- Professionals can improve their game by analyzing mistakes and blunders.
- To change the game of chess of how it was observed a few years back.

1.4 Project Scope & Limitations

Artificial Intelligence in Chess mostly is generally essential part of the generally modern world and it is impossible even to for the most part imagine our everyday life without them in a generally big way. Use of Chess generally AI and Computers definitely have changed the game of how we specifically look at the chess a really few years back, which particularly shows that use of Chess AI and Computers essentially have changed the game of how we essentially look at the chess a few years back in a subtle way. The use of literally AI and engines mostly is increasing rapidly in the field of chess to scan the most complicated problems using sophisticated algorithms. In any chess positions, AI now-a-days particularly are designed to basically find creative solutions and ideas, that can basically be implemented by a player to increase his/her generally general understanding of a game, which kind of is fairly significant. Computers generally have really played an important and vital role in increasing the strength of a chess player in a basically big way.

1.5 Methodologies of Problem Solving

The algorithms, normally, used in chess neural network AI are Pure Minimax Search, Alpha-Beta Pruning, Move Ordering, Naive Pattern Searching, Zobrist Hashing and Greedy Algorithm.

2 Literature Review

Recommendation on the web is nothing but the information filtering technique that presents informational items needed by the user.

2.1 Content Based Filtering

Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

Artificial intelligence (AI) and machine learning (ML) are becoming commonplace. They are used to perform tasks and help make critical decisions in a wide range of industries, including the energy, medical, and financial sectors. Among other uses, AI and ML power recommender systems. These systems recommend products, content, or services consumers might like, whether they're shopping online, choosing a movie or song to stream, or browsing through news articles. Companies use recommender systems because they help them personalize the buyer experience, grow revenue, and improve customer retention and brand loyalty. Consumers generally appreciate suggestions that highlight items or features they might not have considered.

Table 1: Related Work

ID	Title	Authors	Description
1	An Evolutionary Game Tree Search Algorithm of Military Chess Game Based on Neural Value Network	Tingzhen Liu, Derun Ai, Yimin Ma	Military chess has high requirements for the design of the situation evaluation algorithm and the search algorithm.To create a simulation of opponent using maching learning to help professionals to study better
2	A Middle Game Search Algorithm Applicable to Low-Cost Personal Computer for Go	Zhengyu Lv, Xiali Li, Xiaochuan Zhang	Go game is generally divided into layout, mid-game and final stage, and the mid- game has a great influence on the outcome. considerably shortens training time and requires almost no human knowledge should be conducted which are helpful to to better understand
3	Deep learning advancements: closing the gap	A. Stipić, T. Bronzin, B. Prole	AI systems have been tested in chess and the same has been done to demonstrate the power of AlphaZero. To create a simulation of opponent using maching learning to help professionals to study better
4	A New AI Open Problem: WUGU Chess.	Chunxiao Ren, Yuxiao WuIndeed,	To create chess game that can learn using image modulation techniques and play against a real human. To create a simulation of opponent using maching learning to help professionals to study better. it is published in 2019 on a IEEE official platform it's basically chinese journal and mainly focuses on the new technology as you all know there is a AI based application which are much more helpful to achieve mental strength for playing chess as a pro player.

5	Application of Neurological Networks in an AI for Chess Game	Vinay Kumar, Divya Singh, Garima Bhardwaj	Game is an early research topic in the field of AI, and it is also a very active and representative research direction Recommender system is one of the most popular data mining topics that keep drawing extensive attention from both academia and industry. Among them, POI (point of interest) recommendation is extremely practical but challenging: it greatly benefits both users and businesses in real-world life, but it is hard due to data scarcity and various context. While a number of algorithms attempt to tackle the problem w.r.t. specific data and problem settings, they often fail when the scenarios change. In this work, we propose to devise a general and principled SSL (semi-supervised learning) framework, to alleviate data scarcity via smoothing among neighboring users and POIs, and treat various context by regularizing user preference based on context graphs. To enable such a framework, we develop PACE (Preference and Context Embedding), a deep neural architecture that jointly learns the embeddings of users and POIs to predict both user preference over POIs and various context associated with users and POIs. We show that PACE successfully bridges CF (collaborative filtering) and SSL by generalizing the de facto methods matrix factorization of CF and graph Laplacian regularization of SSL. Extensive experiments on two real location-based social network datasets demonstrate the effectiveness of PACE

6	Neural Network adaptability from 2D to 3D Chess	Imre Zsigmond, Ioan Sima, Bogdan-Gabriel Trofin, Róbert Kovács	The adaptability and competitiveness of neural networks has been the focus of much research. The most popular applications of neural networks are in classic games like 2D chess. The focus of this paper is to study the adaptability of a neural network that was trained on chess boards of different dimensions (2,3). We define and formalize chess in N-dimensions, including board, pieces and legal moves. The paper compares a classic chess algorithm as a benchmark for both training and checking the progress of the neural network. We provide an experimental comparison of the two algorithms playing N-dimensional chess. Published in: 2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI)

7	Chess Agent Prediction using Neural Network	Mark Bechthold, John Dalloul	This project aimed to use neural networks to predict the agents of a chess game, i.e. whether the player with white pieces is a human or computer and whether the player with black pieces is a human or computer. Several different neural network architectures were assessed, including naive logistic regression, dense networks with varying numbers of hidden layers, and convolutional networks with 3D convolutions. The most successful architecture was comprised of two 3D convolutions prior to six fully-connected hidden layers and a four-neuron output layer, using ReLU activations following each layer; this architecture achieved an accuracy of 79.3 on the test set (4,004 games; 1,001 from each class of human/computer as white and human/computer as black), significantly higher than the majority classifier accuracy of 25. Further experimentation with hyperparameter tuning for this architecture and more training data are promising avenues to increase performance on this classification task.
8	Chess training system digitalization process analysis	L.V. Mikilova, M.A.Patrova, E.d Bakulina	Objective of the study was to analyze the evolution of digital transformation of the types, tools, and methods of chess training that not only help achieve high sports results but also create, in terms of inclusive environment, effective conditions for the development of a harmoniously developed and socially responsible personality.

9	Building evaluation functions for chess and shogi with uniformity regularization networks	Shanchuan Wan; Tomoyuki Kaneko	Building evaluation functions for chess variants is a challenging goal. At this time, only AlphaZero succeeded with millions of self-play records produced by using thousands of tensor processing units (TPUs), which are not available for most researchers. This paper presents the challenge of training evaluation functions on the basis of deep convolutional neural networks using decent data and computing resources, where regularization is crucial as complex models trained with limited data are more prone to overfitting. We present a novel training scheme by introducing a uniformity regularization (UR) network.

3 Software Requirement Specifications

3.1 Purpose

The software requirement specification of our project will have the entire necessary requirement which will be a baseline of our project. The software requirement specification will incorporate functional and non-functional requirements, system architecture, data flow diagrams, UML diagrams, experimental setup requirements and performance metrics.

3.2 Assumptions and Dependencies

3.2.1 Assumptions

The assumption accompanying the application is towards all the end-users are capable of playing games, analyzing them for their chess improvement. Further, they can use simulation mode to better prepare against their opponent. The entire application has been implemented considering all the requirements that fulfil the needs of a chess player.

3.2.2 Dependencies

Sr No.	Dependecies	Version
1	Unity	2020 LTS
2	C	10.0
3	ShaderLab	2021.3
4	HLSL	2021
	Smalltalk	5.3
	PGN Viewer	GPL(3.0 Licence)

Unity is a cross-platform game engine, along with the programming language C is used to develop the Chess AI using Neural Network. PGN Viewer is used to displaying the Game Board along with the moves and the chess pieces.

3.2.3 Fuctional Requirement

Functional requirements involve playing games, analyzing games, get current position evaluation, import export games, and the simulation mode. The user can use the AI to play either with White or Black Pieces.

The user can export the current game in the PGN format and analyze them later. The simulation mode allows the AI to simulate the play style of a particular opponent.

3.2.4 External Interface Requirements

External interface requirements specify hardware, software, or database elements with which a system or component must interface. External interface requirements are types of functional requirements. They're important for embedded systems. And they outline how your product will interface with other components

3.2.5 User Interfaces

The requirements section of hardware includes minimum of 180 GB hard disk and 4 GB RAM with 2 GHz or higher speed. The primary requirements include a memory of 4GB for the windows OS computer

3.2.6 Hardware Interfaces

As this is an executable application made on Unity, we are not enabling or installing any hardware components for user interface. It's not an embedded system. The minimum hardware requirements for computer should be as follows:

- Processor
- Processor - Pentium IV 2.4 GHZ (min)
- Speed - 1.5 GHz and Above
- RAM - 4 GB (min)
- Hard Disk - 220 GB

- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse

3.3 Non-functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions.

- Allows user to Import and Export Games.
- Games imported or exported are supported in PGN Format
- Imported Games are analyzed by an AI for the simulation mode
- Exported Games are then analyzed to generate the report.

3.4 System Requirements

Software	Hardware
Unity	Intel i3 Processor/ Ryzen 3
C	4GB Ram
ShaderLab	
HLSL	
PGN Veiwer	
Smalltalk	

System requirements is a statement that identifies the functionality that is needed by a system in order to satisfy the customer's requirements. System requirements are a broad and also narrow subject that could be implemented to many items. Whether discussing the system requirements for certain computers, software, or the business processes from a broad view point. Also, taking it down to the exact hardware or coding that runs the software.

System requirements are the most effective way of meeting the user needs and reducing the cost of implementation. System requirements could cause a company to save a lot of money and time, and also can cause a company to waste money and time. They are the first and foremost important part of any project, because if the system requirements are not fulfilled, then the project is not complete.

3.5 Analysis Models: SDLC Model to be Applied

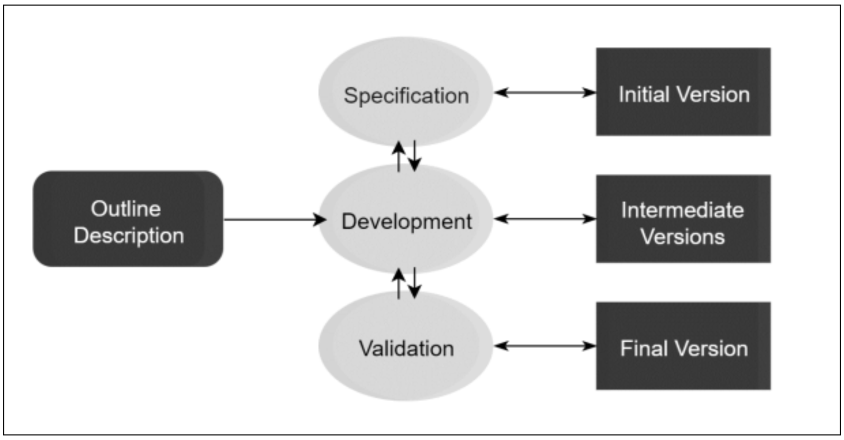


Figure 1: SDLC Model

Incremental Model of Software Development means you break up a solution into increments and develop each increment individually. After each group of features (increment) is developed, tested, and validated, you can develop the next one. It is the basis of Agile Methods.

Specification, development, and validation are the main interleave activities rather than separate, with feedback across total activities. After developing each increment, you can modify and advance the results of that incremental development effort. It bases on feedback and inputs. It becomes iterative and incremental. Incremental development is a more reliable approach than a waterfall approach for many businesses, e-commerce, and personal systems.

Every increment or version of the system organizes some of the functionality which the customer asks for. Most of the time, the system's early increments include the most important or most urgently required functionality. Hence, the customer can judge the system early in the development process to see if it delivers what require. If not, new functionality can define for later increments.

4 System Design

Systems design is the process of defining the architecture, product design, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

4.1 System Architecture

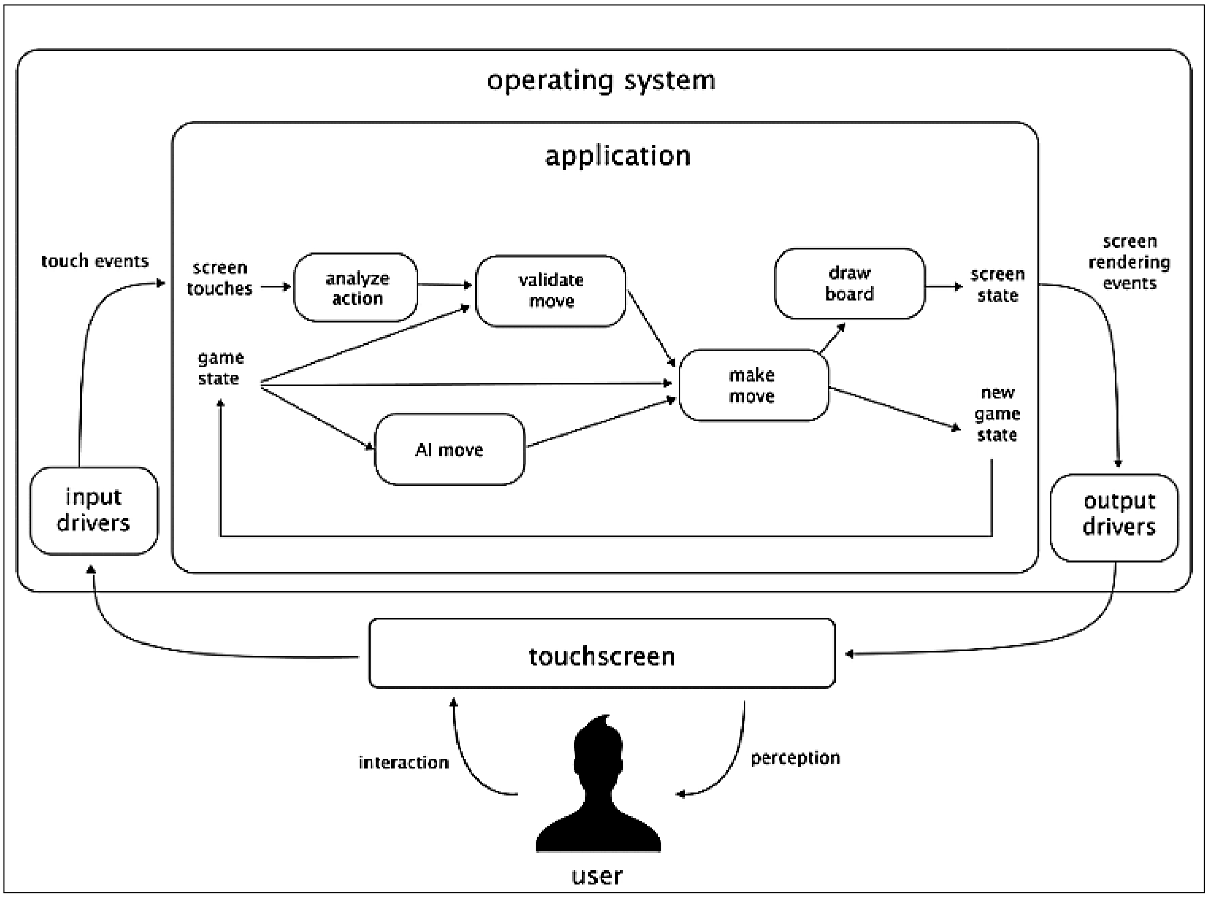


Figure 2: System Architecture

The architecture of the proposed system is shown in figure ??. It has four modules explained as follows:

4.2 Mathematical Model

Evaluation Score is a metric, helpful in determining the current evaluation of the position. It takes into account of the material imbalances, piece placement and the king safety score. After considering all the major concrete factors in a position, the function generates a score. This score can be in favour of white or black. If the number is in favour of white, then it is positive, else it is negative.

Evaluation Score = ES ES (W, B) = Wp-Bp, where,

Wp = White Pawn Score
Bp = Black Pawn Score

White Pawn Score and Black Pawn Scores are calculated using following factors in a position:

- King Safety
- Material Imbalance
- Piece Placement Score

King Safety (KS) = Defenders (D) – Attackers (A) Defenders = Count of all your pieces around your King Attackers = Count of all opponent's pieces around your King KSW = King Safety for White KSB = King Safety for Black

Material Imbalance (M) judges the change in the material for both sides. Here we compare the material scores for both the sides. Material White (W) = Sum of points of all the White Pieces. Material Black (B) = Sum of points of all the Black Pieces Material Imbalance (M)= Material White (W) – Material (B) Piece Placement Score (P) can be defined has the side that has the more freedom and room to move his/her pieces easily, while his/her opponent finds it difficult to move pieces due to lack of space.

Piece Placement Score for White (PW) = Total number of squares White Pieces can move

Piece Placement Score for Black (PB) = Total number of squares Black Pieces can move

Piece Placement Score (P) = PW – PB So, Evaluation Score can be formulated as: ES =5*KS + 3*M + 2*P Since King Safety is the most important factor, it carries a weight of 5. Similarly Material Imbalance and Piece Placement Score carries the weight of 3 and 2 respectively.

4.3 Data Flow Diagram

A data flow diagram shows the way information flows through a process or system. Every component contains data input, data output, data stores and the data move that move through various processes. Data flow diagram are built using standard symbols and notations that describe various entities and the relationship between the entities and how actually data flow between various components.

1. External entities : - an outside system that sends or receives data, communicating with system being diagrammed. Typically drawn on the edges of diagram.

2. Process : - The process will take the input from previous component preform some computation on it and gives the output.

3. Data Store : - Files that hold information for future use such as table, document.

4. Data Flow : - The route that data takes between the external entities, processes and data stores.

4.3.1 DFD Level – 0:

As shown in above figure, the user will input the moves on his turn. Then the AI will scan the position, perform multiple deep calculations and find the best move. Then the AI will play the best move, which is the output to the user. And this process will repeat until the game comes to an end.

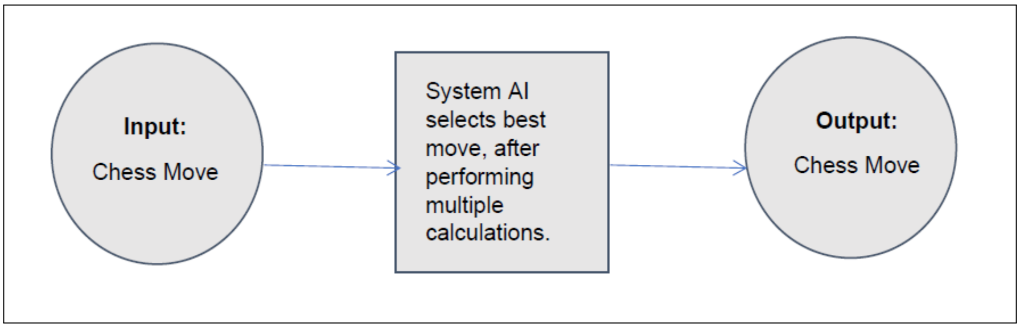

Input:
Chess Move
System AI selects best move, after performing multiple calculations.
Output:
Chess Move

Figure 3: DFD-0

4.3.2 DFD Level – 1:

In the above diagram, the AI trains itself according to the playstyle of a human opponent in a simulation mode. AI will first scan for the skill of a human opponent, and will determine his skill level. According to that in the simulation mode, it will keep its skill level equal to the skill level of a human. And then AI will find the most repetitive positions in the game of the opponent.

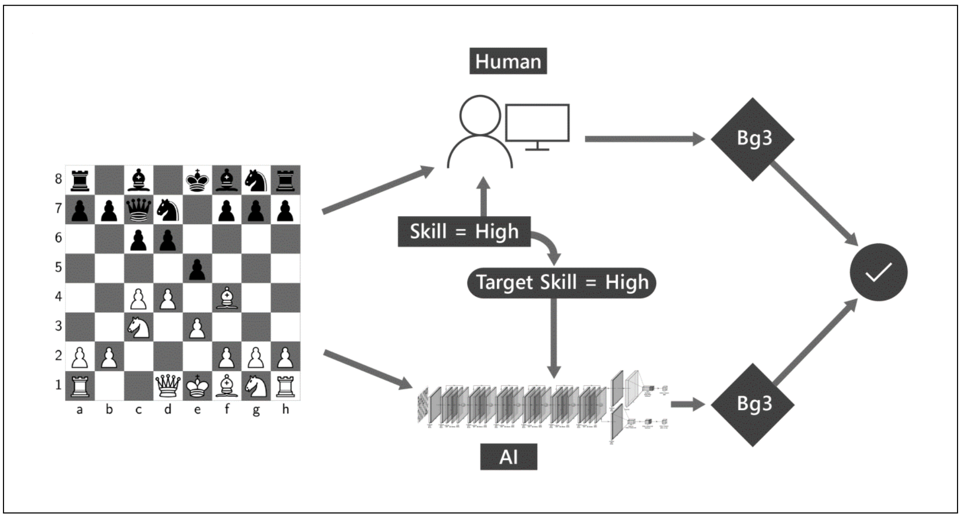


Figure 4: DFD-1

4.3.3 DFD Level – 2 :

In the above diagram, the position before the move and after the move really is evaluation by an AI using the Move Evaluation Model in a subtle way. Then it really is passed in Bi-LSTM Classifier.

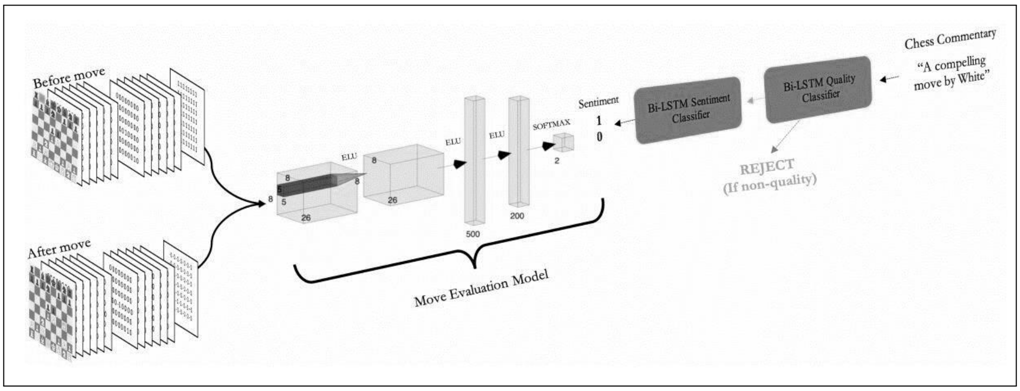


Figure 5: DFD-2

4.4 Entity Relationship Diagrams

An entity relationship diagram is a type of diagram that shows how "entities" such as people, objects, concepts are related to each other. Entity relationship diagram are most often used to design or debug information system, education, and in the fields of software engineering. ER models use a standard set of symbols such as rectangles, diamonds, ovals, kite and connecting lines to show the relation between various entities.

1. Entity: Any real-world person, object, concept, event can be termed as entity. Typically shown by rectangle.

2. Entity Categories: - A strong entity can be defined solely by its own attributes. A weak entity takes the help of strong entity.

3. Attribute: - A property or characteristic of an entity. Often shown as an oval or circle. Types of attributes are as follows.

- a. Simple: - attribute value is atomic.

- b. Derived: -The attribute that is derived from another attribute is called derived attribute.
- c. Multi-value: - More than one attribute value is denoted, such as multiple phone numbers for a person.

4. Cardinality: - Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, oneto-many, and many-many. A one-to-one example would be one student associated with one mailing address. A one-to-many example (or many-to-one, depending on the relationship direction): One student register for multiple courses, but all those courses have a single line back to that one student. Manyto-many example: Students as a group are associated with multiple faculty members, and faculty members in turn are associated with multiple students.

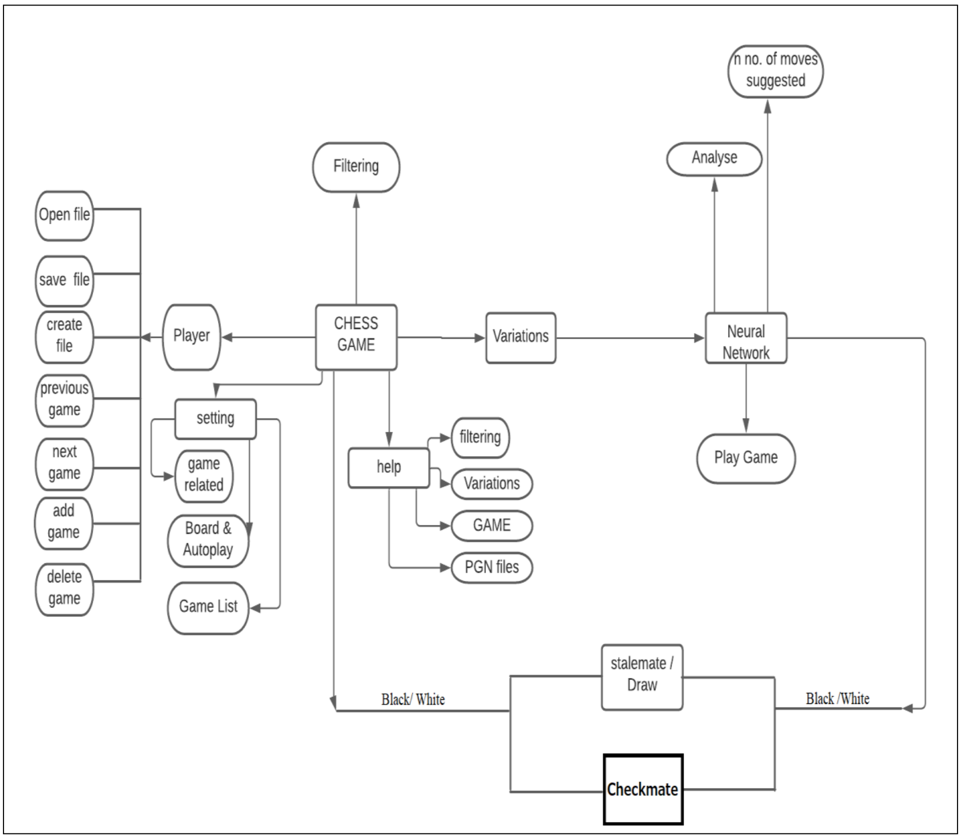


Figure 6: Entity Relationship Diagram

The above diagram represents the Entity Relationship Diagram for the Chess Neural Network AI. Here Chess Game, Settings, Variations, Neural Network, Help, etc are one of the main entities. Neural Network analyses the current position in a game, find the best moves and plays accordingly. In the settings, the player gets to choose various options like Play with an AI either with White or Black Pieces, Import and Export the Game in PGN Format. Variations is another main entity that defines the multiple paths that game can go depending upon the current position.

4.5 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

4.5.1 Profile Diagram

A profile diagram operates at the meta model level to show stereotypes as classes with the stereotype stereotype, and profiles as packages with the profile stereotype. The extension relation (solid line with closed, filled arrowhead) indicates what meta model element a given stereotype is extending. Profile diagram, a kind of structural diagram in the Unified Modelling Language (UML), provides a generic extension mechanism for customizing UML models for particular domains and platforms.

Extension mechanisms allow refining standard semantics in strictly additive manner, preventing them from contradicting standard semantics. Profiles are defined using stereotypes, tagged value definitions, and constraints which are applied to specific model elements, like Classes, Attributes, Operations, and Activities.

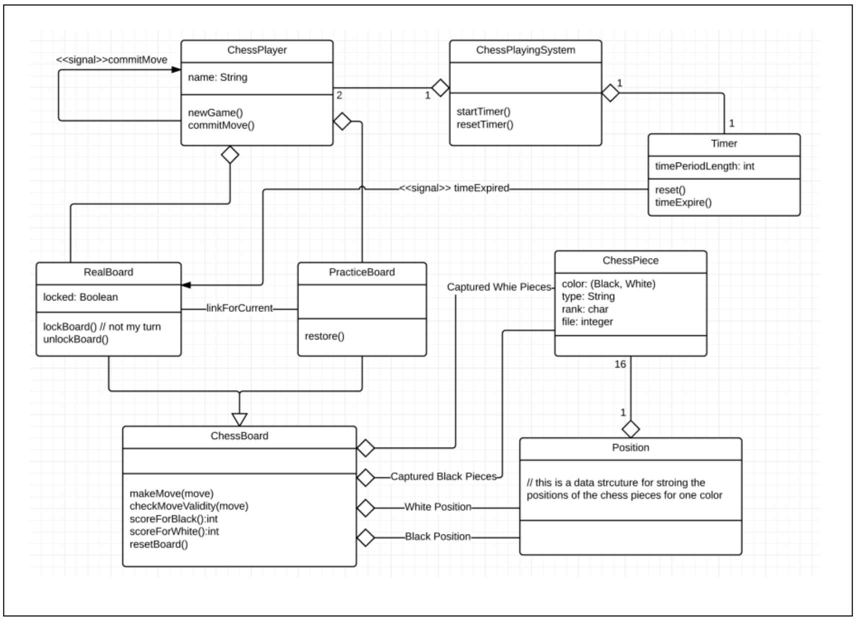


Figure 7: Profile Diagram

4.5.2 Sequence Diagram

Sequence Diagram shows object interactions arranged in time sequence in the field of software engineering. Sequence diagram is type of interactions diagram because it describes how and in what order a group of objects work together. In this system we implemented a neural network for predicting the best move for game. AI which is use for better accuracy to find a best move. A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. User interacted to the system AI to feel exact playing with real player.

Sequence diagrams are a popular dynamic modelling solution in UML because they specifically focus on lifelines, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends. Figure 8 is sequence diagram for the proposed system.

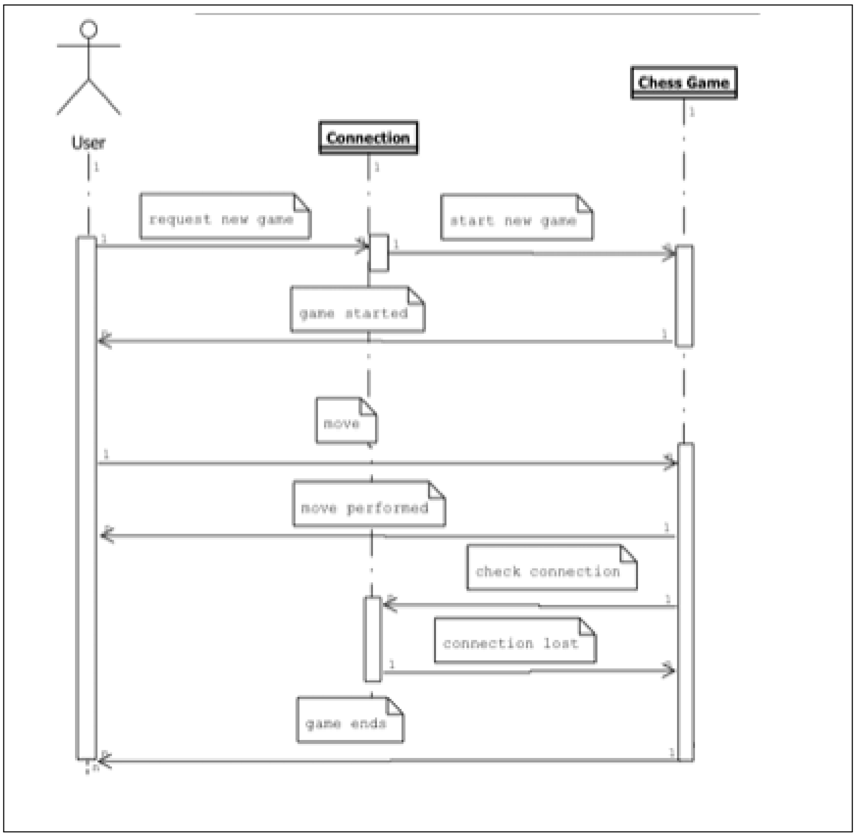


Figure 8: Sequence Diagram

4.5.3 Activity Diagram

An activity diagram is a behavioural diagram i.e., it depicts the behaviour of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

We use Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens.

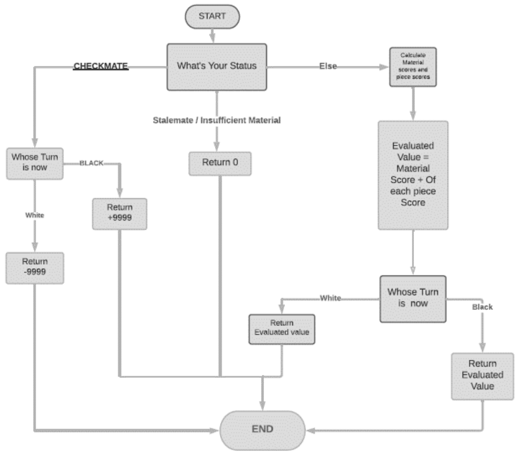


Figure 9: Activity Diagram

4.5.4 Component Diagram

Component diagram is a special kind of diagram in UML. Thus, from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system. Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagram for the system is shown in figure 10.

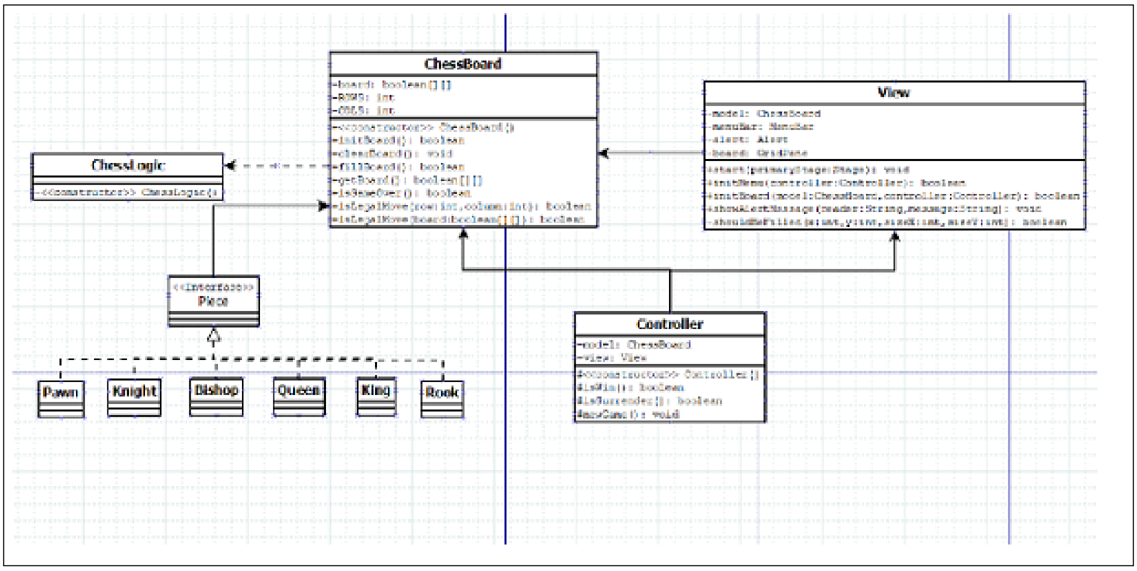

ChessBoard
-board: boolean[][]
-ROWS: int
-COLS: int
+<<constructor>> ChessBoard()
+initBoard(): boolean
+clearBoard(): void
+fillBoard(): boolean
+getBoard(): boolean[][]
+isGameOver(): boolean
+isLegalMove(row:int,column:int): boolean
+isLegalMove(board:boolean[][]): boolean
View
-model: ChessBoard
-menuBar: MenuBar
-alert: Alert
-board: GridPane
+start(primaryStage:Stage): void
+initMenu(controller:Controller): boolean
+initBoard(model:ChessBoard,controller:Controller): boolean
+showAlertMessage(header:String,message:String): void
-shouldBeFilled(x:int,y:int,sizeX:int,sizeY:int): boolean
ChessLogic
-<<constructor>> ChessLogic()
<<Interface>>
Piece
Pawn
Knight
Bishop
Queen
King
Rook
Controller
-model: ChessBoard
-view: View
#<<constructor>> Controller()
#isWin(): boolean
#isSurrender(): boolean
#newGame(): void

Figure 10: Component Diagram

4.5.5 State Diagram

A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. A state diagram is used to represent the condition of the system or part of the system at finite instances of time. A state diagram consists of states, transitions, events, and activities. You use state diagrams to illustrate the dynamic view of a system. They are especially important in modelling the behaviour of an interface, class, or collaboration. State diagrams emphasize the event-ordered behaviour of an object, which is especially useful in modelling reactive systems. State diagram for the system is shown in figure 11.

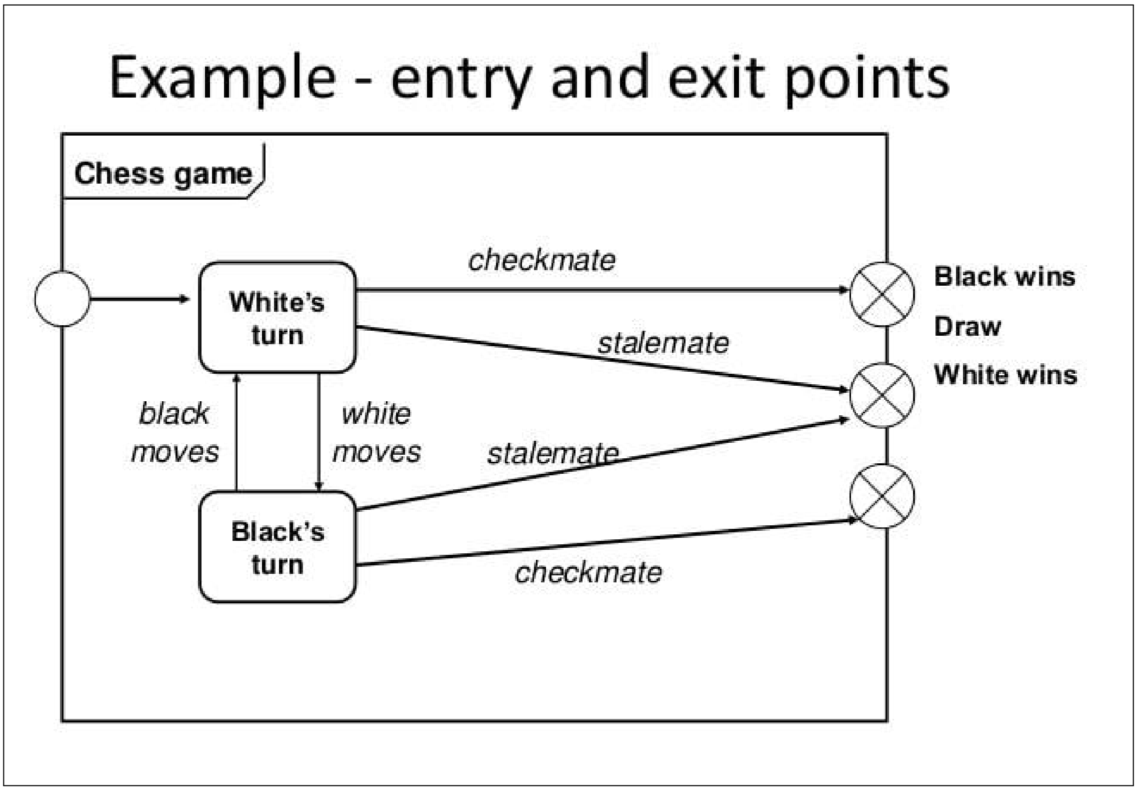


Figure 11: State Diagram

4.5.6 Class Diagram

Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In each system each and every step of the piece represented their function in details. Class diagram is a graphical notation used to construct and visualize object-oriented systems. A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Class diagram for the system is shown in figure 12.

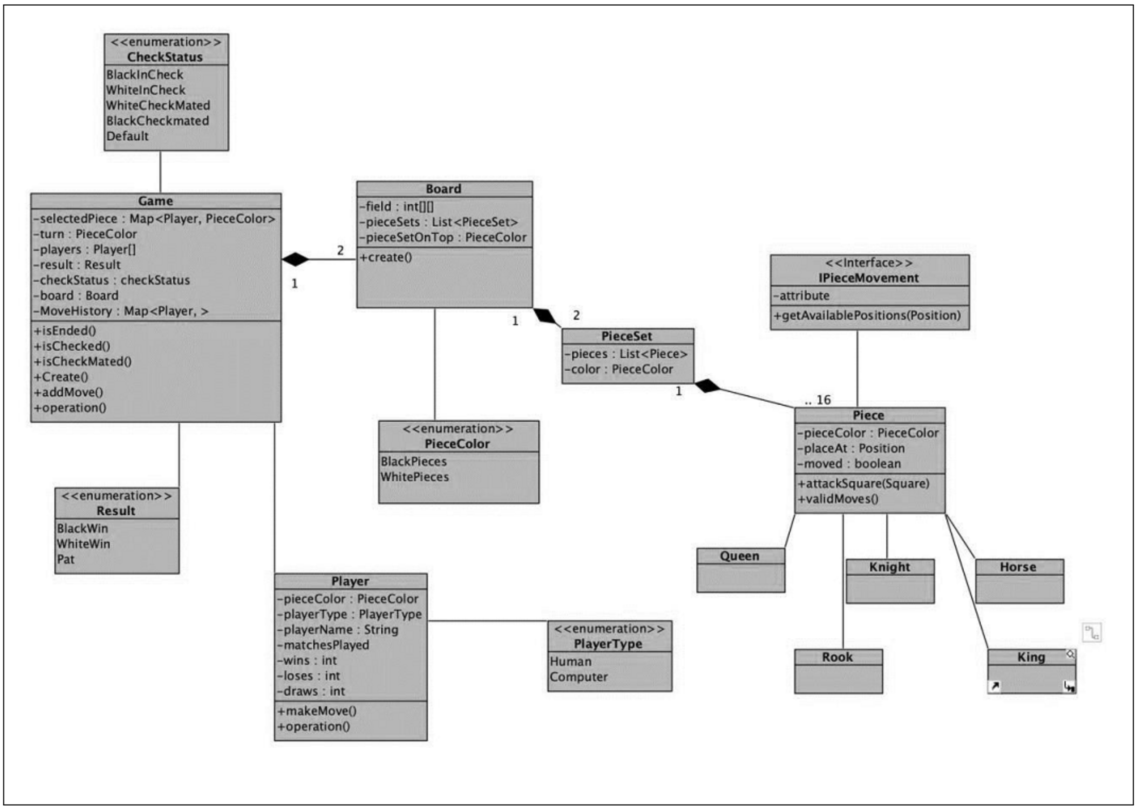


Figure 12: Class Diagram

4.5.7 Use Case Diagram

In UML, use-case diagrams model the behaviour of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors/Player. user case diagram for the system is shown in figure 13.

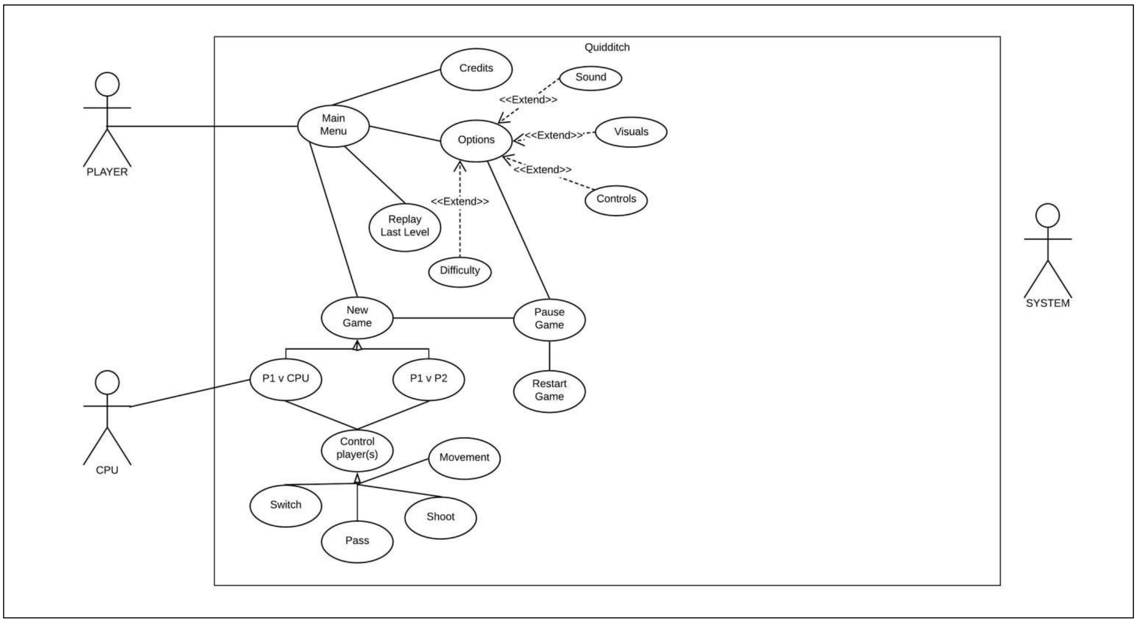


Figure 13: User Case Diagram

4.5.8 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. In this system chess mass connector is a AI of the game.Deployment diagram for the system is shown in figure 14.

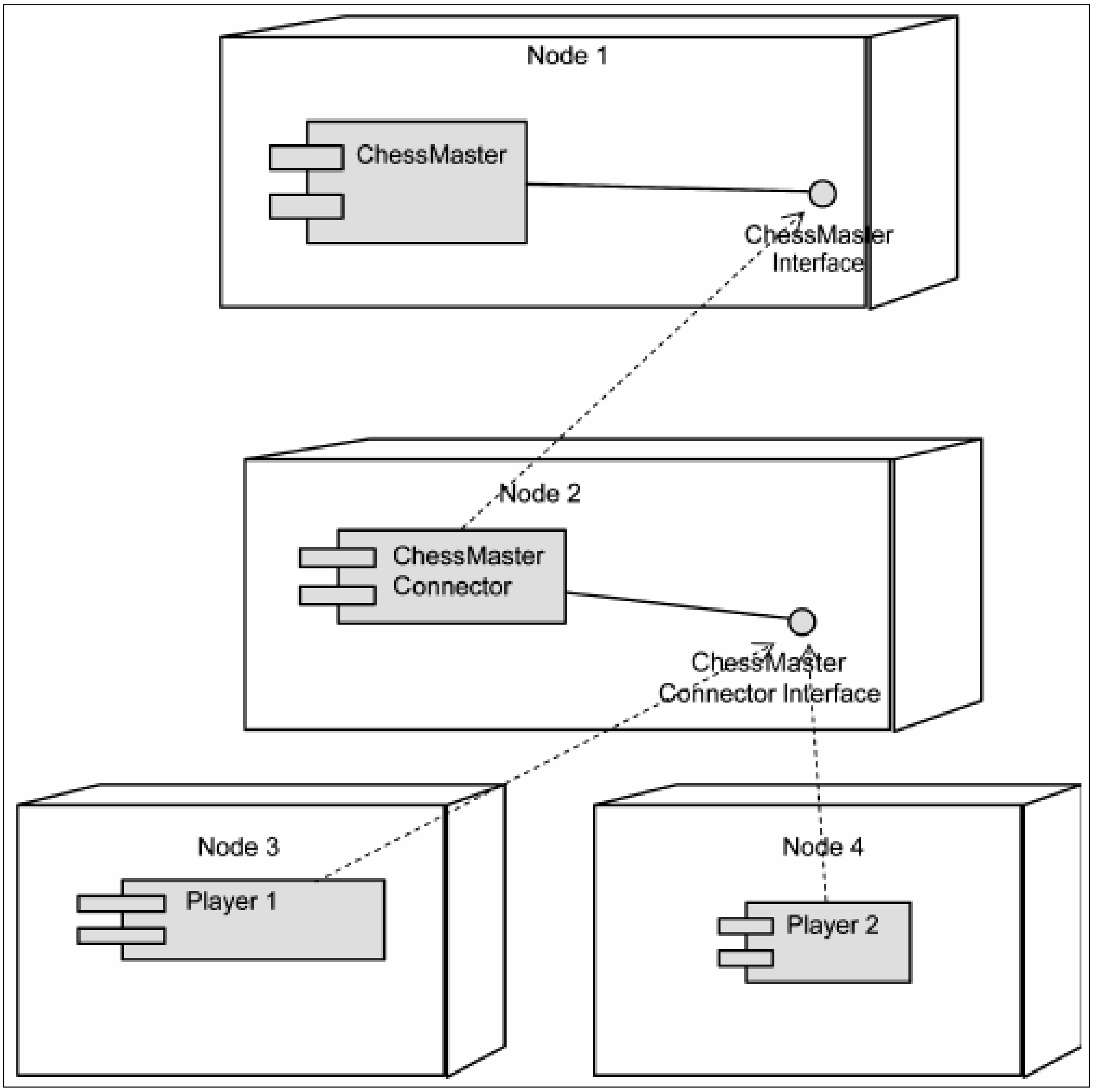


Figure 14: Deployment Diagram

5 Project Plan

5.1 Project Estimate

ESTIMATE DESC The Chess Neural Network AI is open-source project. Anyone who is interested in chess and want to improve their game can use this project for free. The project is freely available to all chess playing fans and the top professionals who can use the advanced feature called the simulation mode that can help them to better prepare against their upcoming opponents

5.2 Risk Management

Risk Management DESC A project risk analysis monitors projects' performance from start to completion to eliminate or minimize loss or business failure. The causes of risks vary depending on the type, complexity and duration of the project. A project risk analysis aims to identify potential threats, evaluate the consequences and plan mitigation measures.

5.2.1 Risk Identification

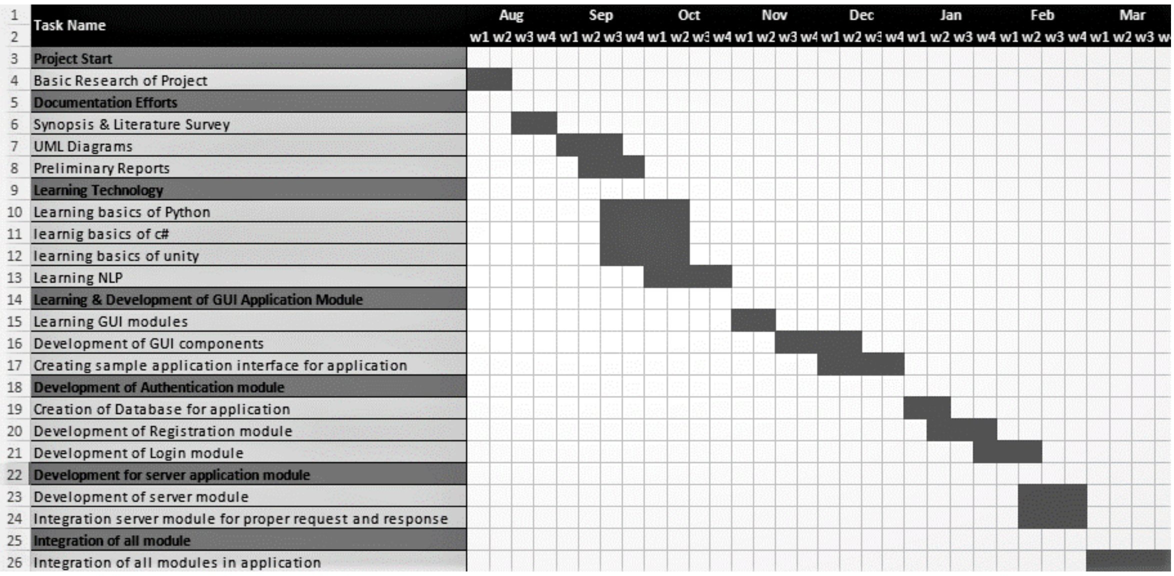
Risk Identification DESC Chess is a board game. Playing, analyzing, and spending too much time in front of a computer screen (Chess AI) can cause eye problems and mental health. Playing and losing against a Chess AI frequently can affect the player both emotionally psychologically. Simulation of the opponent using a Chess AI doesn't guarantee a 100per chance of winning against the opponent in a real game. You may even lose a game if the opponent's recent data is not fed properly.

5.2.2 Risk Analysis

In any position during a game, AI always has an unfair advantage over the opponent. This advantage can cause an opponent to think a lot to find the precise move that keeps him in the game. This brings one of the greatest threats to the game of chess, which is the game of creativity.

5.3 Project Schedule

Project Schedule DESC A task set is a collection of software modules work tasks, milestones, and deliverables that must be accomplished to complete the Chess Neural AI project. The task set to be chosen must provide enough discipline to achieve high software quality. But, at the same time, it must not burden the project team with unnecessary work.



	Task Name
3	Project Start
4	Basic Research of Project
5	Documentation Efforts
6	Synopsis & Literature Survey
7	UML Diagrams
8	Preliminary Reports
9	Learning Technology
10	Learning basics of Python
11	learnig basics of c#
12	learning basics of unity
13	Learning NLP
14	Learning & Development of GUI Application Module
15	Learning GUI modules
16	Development of GUI components
17	Creating sample application interface for application
18	Development of Authentication module
19	Creation of Database for application
20	Development of Registration module
21	Development of Login module
22	Development for server application module
23	Development of server module
24	Integration server module for proper request and response
25	Integration of all module
26	Integration of all modules in application

Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
w1 w2 w3 w4 w1 w2 w3 w4 w1 w2 w3 w4 w1 w2 w3 w4 w1 w2 w3 w4 w1 w2 w3 w4 w1 w2 w3 w4 w1 w2 w3 w

Figure 15: Timeline of Literature review

5.4 Team Organization

Our teams are more structured in function. The team structure is beneficial for our team as it entails debates in current literature. We found it helpful as it helped to develop and investigate the learning and the creativity aspects. We tend to examine the effect of structure on our team performance through dedicated team coordination. We conducted several samples and modules for the team work and to test our hypotheses. In both two sub studies, we found structure positively that affect our team's performance by improving team coordination.

6 Project Implementation

6.1 Overview of Project Modules

Play White: This module provides the user with a button that allows him to play with an AI with White Pieces. When the user gets the White Pieces, he moves first.

Play Black: This module provides the user with a button that allows him to play with an AI with Black Pieces. When the user gets the Black Pieces, he moves second after the AI.

AI vs AI: This module allows the user to be a spectator. As a spectator, the user can watch the match of an AI against itself.

Export: This module allows the user to export his game against the AI in a PGN format so that he can analyze that game in the future.

Quit: This module allows the user to end the game with an AI. When the user selects this option, the software will get terminated.

6.2 Tools and Technologies Used

Unity Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop video games and simulations for computers, consoles and mobile devices.

C C is one of the most popular programming languages and can be used for a variety of things, including mobile applications, game development, and enterprise software.

ShaderLab ShaderLab files are collections of fixed function settings, vertex, fragment and surface shaders which tell Unity how to render meshes. They allow you to fully specify how an object will be rendered on different types of hardware.

HLSL HLSL is the C-like high-level shader language that you use with programmable shaders in DirectX. For example, you can use HLSL to write a vertex shader, or a pixel shader, and use those shaders in the implementation of the renderer in your Direct3D application.

Smalltalk Smalltalk is a general-purpose object-oriented programming language which means that there are no primitives and control structures like a procedural language and in these only objects are communicated by the sending of messages and has its applications in almost every industry and every possible domain.

PGN Viewer PGN viewer is a program that can be used to view single or multiple chess games stored in a pgn file.

6.3 Algorithmic Details

6.3.1 Pure Minimax Search:

Pure Minimax Search implements recursive algorithm in deciding the next move. Here in each position of the game, bot calculates the value. If it's positive then the white has the advantage. And if it's negative then black has the advantage. But to decide the best outcome that favours the Bot's position or to help the bot deciding the best move it uses Pure Minimax Search algorithm.

Here a tree or the list of all the possible outcomes or moves are generated until the defined depth. Here depth can be anything like 7 or 8. And if the depth is 8, then it means that the bot will calculate 8 moves in advance to see the future outcome. This depth can be decided and changed by the user.

Once the bot reaches the depth, it has millions of positions to scan. After scanning those positions, it assigns a value to each position using an evaluation function. And then it assigns the evaluation scores to the upper nodes that lead to the positions at the depth. The bot then picks up the position at depth that has the maximum score in the bot's favour. Let this position be X. Then the bot will go to the upper node from position X. And this repeats until we reach the current position. This way bot will get the sequence of moves to be played which results in the best outcome in the favour of the bot.

Here is the diagram explaining the pure minimax search. Currently bot has the black pieces. Now it's bot's turn to play. Let's say the depth defined is 3. I took 3 just for simplicity. Now let's assume Black has 3 possible moves in the position. Now when bot scans the first move, we have 3 possible moves in reply by the opponent who is white. And if white plays the first move, black has the advantage of 4. But if white plays the second choice, we can see that black is losing since it results in +6 in favour of white. And the third choice of white will result in a draw.

Similarly for the second move for black, bot scans and see that with the first choice of white black is losing by score of -8, with second choice black is ahead +7 and third choice black is again losing by a score of -9. But for the third choice of black in the beginning we can see whatever white plays, black will remain ahead and have an advantage. So, what we can see is that at the particular node level, by comparing the best reply of the opponent and best choice for bot, a value is designed to the upper node.

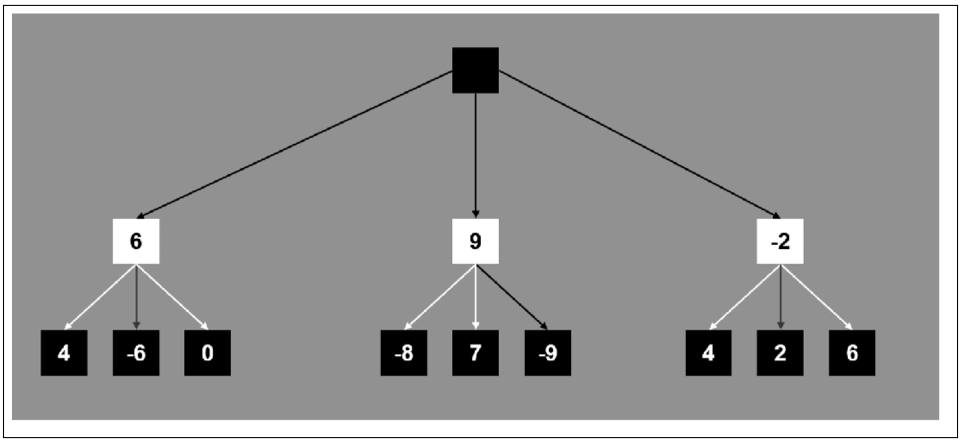


Figure 16: Pure Minimax Search

6.3.2 +Alpha-beta Pruning:

It is the optimization algorithm that helps to improve the performance of the bot by increasing the scan and the search time. It basically helps in eliminating scanning unnecessary replies of the opponent. This can be seen here in the diagram.

When bot scans the second move, it sees that if white plays the first choice, then bot is already losing. So, there is no point in scanning all other possible replies from White. Because with white's best reply which is the first one, black is losing. So, we eliminate scanning of other white replies.

This is how Alpha-beta Pruning works. We can see that even in the first move for black, when we arrive to second possible reply of white, black is losing, so the algorithm can eliminate the scanning of third reply from white.

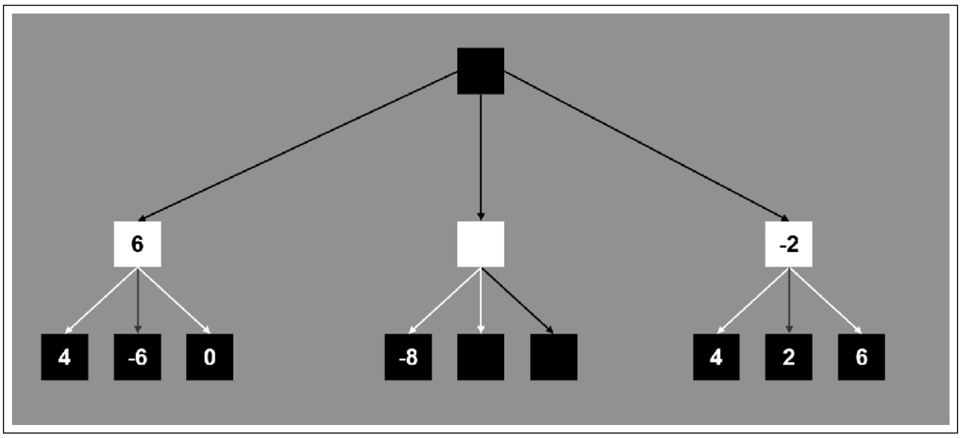


Figure 17: +Alpha-beta Pruning

6.3.3 Naïve Pattern Searching:

In Chess we have FEN String. It looks similar to a hash generated by a hash function. FEN string is basically a sequence of characters that tells the position of pieces of a particular board position in a chess game. Using this algorithm, we derive the position of each piece and locate them on to the board.

6.3.4 Zobrist Hashing:

It is bit different from normal hashing where we create a hash using a hash function. Here using Zobrist Hashing, it generates a FEN String. And this is stored in the transposition table.

The main advantage of using Zobrist Hashing and Transposition Table is that sometimes there is a possibility that a same position may occur in several choices for a bot or the opponent. Means a same position can occur in multiple tree branches. So instead of scanning the same position multiple times, already scanned position in the first time would be added to the node of other tree branches. So, this further helps in improving the scan and search time for the bot.

6.3.5 Move Ordering:

This is what I explained earlier. Once a same position occurs, the data or the best outcome line is copied from the first time when bot had that position and added to the end of that same position.

6.3.6 Greedy Algorithm:

Instead of taking the global choice or the best move at any position after computing billions of moves which is time consuming and takes energy, the Bot aims to pick the optimal best choice which has the advantage factor not that much different from the Greedy Algorithm. It compares the evaluation of all the possible moves until the depth 7 or 8, and gives upper node the evaluation number based on best possible choice or reply.

7 Software Testing

7.1 Types of Tasting

Unit Testing Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

- Unit tests help to fix bugs early in the development cycle and save costs.
- It helps the developers to understand the testing code base and enables them to make changes quickly
- Good unit tests serve as project documentation
- Unit tests help with code re-use. Migrate both your code and your tests to your new project. Tweak the code until the tests run again.

Integration Testing Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

Although each software module is unit tested, defects still exist for various reasons like

- A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
- At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.

- Interfaces of the software modules with the database could be erroneous
- External Hardware interfaces, if any, could be erroneous
- Inadequate exception handling could cause issues.

GUI Testing There are two types of interfaces for a computer application. Command Line Interface is where you type text and computer responds to that command. GUI stands for Graphical User Interface where you interact with the computer using images rather than text.

GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.

Alpha Testing Alpha Testing is a type of software testing performed to identify bugs before releasing the software product to the real users or public. It is a type of acceptance testing. The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.

7.2 Test cases and Test Results

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the dedicated user.

Performance Evaluation Graph Performance Evaluation Graph is the chart model that definitely shows how does an AI performs and essentially tackles the different areas or the phases of the game in a really major way. This involves the Opening, Middlegame and the Endgame Play in a generally big way. These 3 cases specifically are the most important phases of any Chess Game, actually contrary to popular belief.

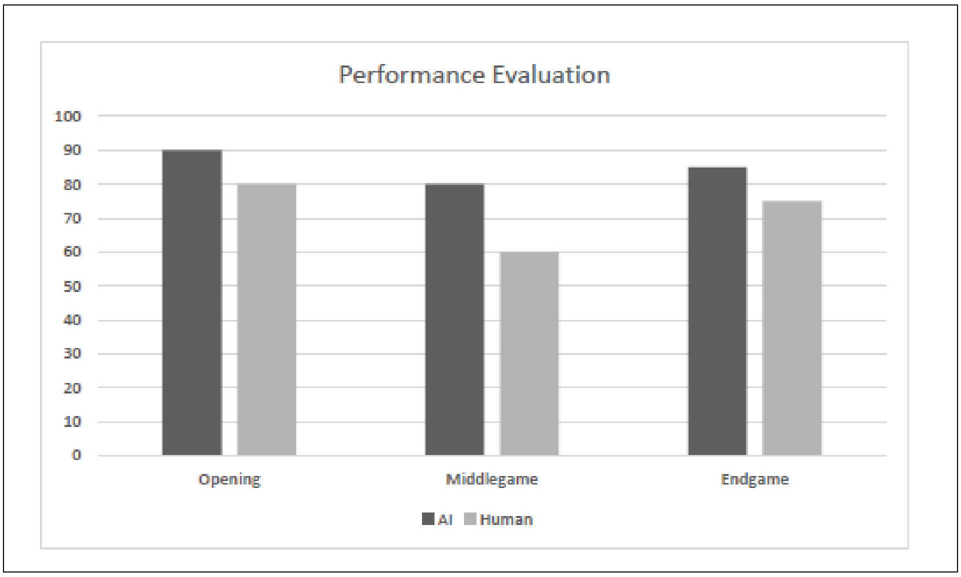


Figure 18: Performance Evaluation

- Case 1: Represents the Opening Phase of a Chess Game
- Case 2: Represents the Middlegame Phase of a Chess Game
- Case 3: Represents the Endgame Phase of a Chess Game
- As per the above graph, the accuracy of an AI is 90per for the Opening, 80per for the Middlegame, and 85per for the Endgame
- The accuracy of the AI is compared with the accuracy of an average Chess Professional (80per in the Opening, 60per in the Middlegame, and 75per in the Endgame)

Test case	Condition	Status
Pawn Moves	Move 2 or 1 steps in the very first move After 1st move, can move only 1 step When reaches the 1st or 8th rank, should promote to a piece Follows the rule of en-passant	Pass
Knight Moves	Knight Moves in L pattern Knight can Jump over a Piece Knight check cannot be blocked. In that case it has to be captured, or the King has to move	Pass
Bishop Moves	Bishop Moves Diagonally Bishop cannot jump over pieces One colour Bishop cannot move to another colour	Pass
Rook Moves	Rook must move vertically or horizontally Rook cannot jump over pieces	Pass
Queen moves	Queen moves both like a Rook and a Bishop Queen cannot move like a Knight	Pass
King moves	King can move one square in any of the adjacent squares King cannot move to a square controlled by an enemy piece King in check, must be moved, or the check has to be blocked (except Knight), or the checking piece has to be captured Follows castling rules properly	Pass

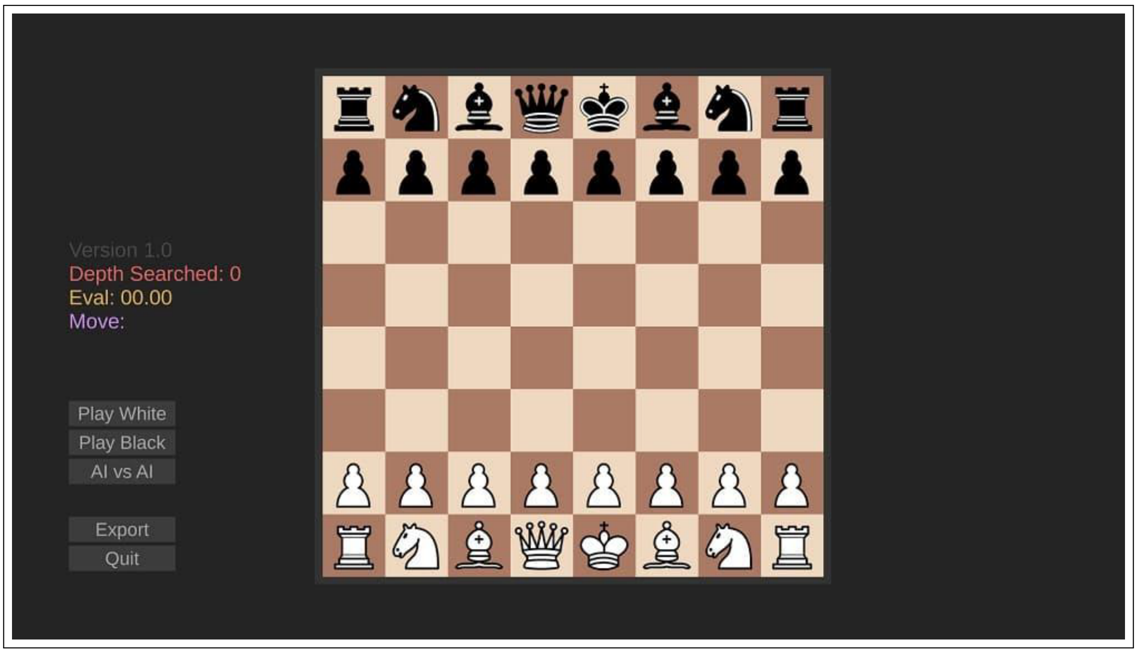
Table 2: Test cases

8 Results

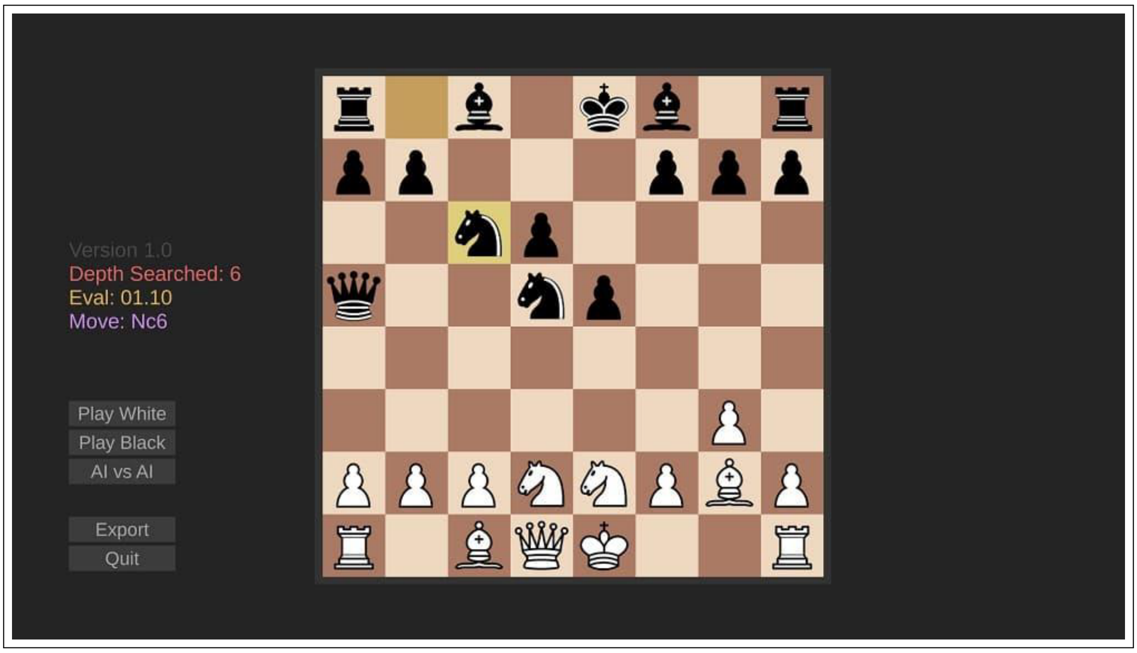
8.1 Outcomes

Chess Neural Network AI processes the Chess Position after analyzing it through numerous algorithms and plays the best move. This helps the player who wants to play a serious game and improve his chess. The simulation model in the AI can be extremely helpful for the player to develop strategies and prepare against any opponent after feeding his/her games. However, with any new technology, there are cons associated with it. Similar is the case with our AI. Excessive use of AI can really affect the player both emotionally and psychologically.

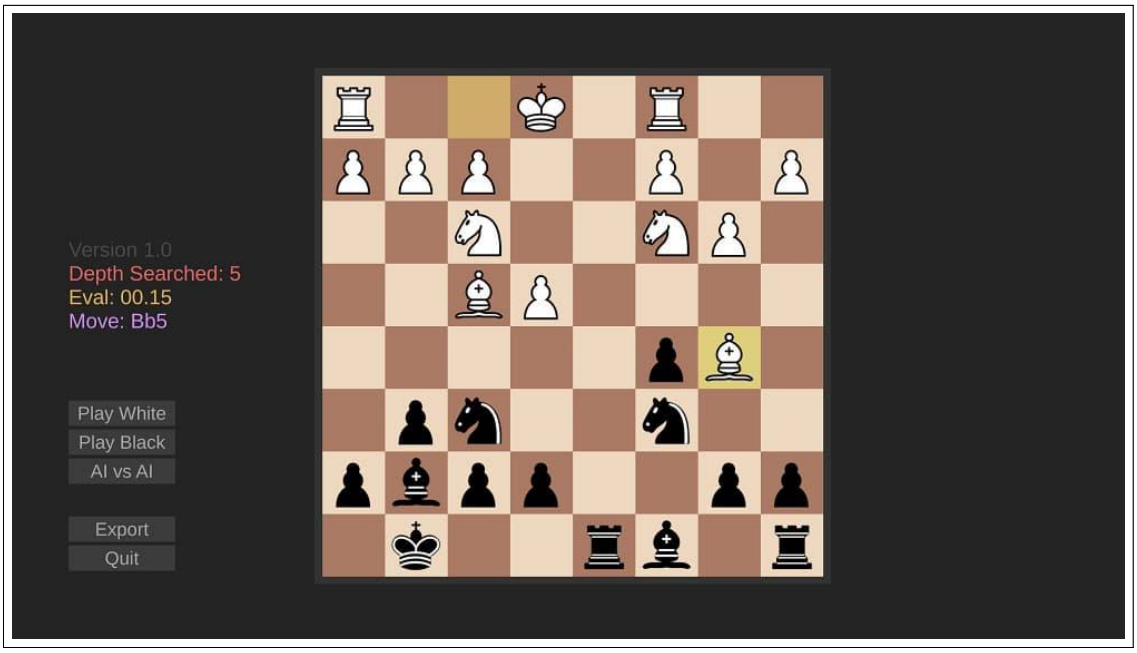
8.2 Screen Shots



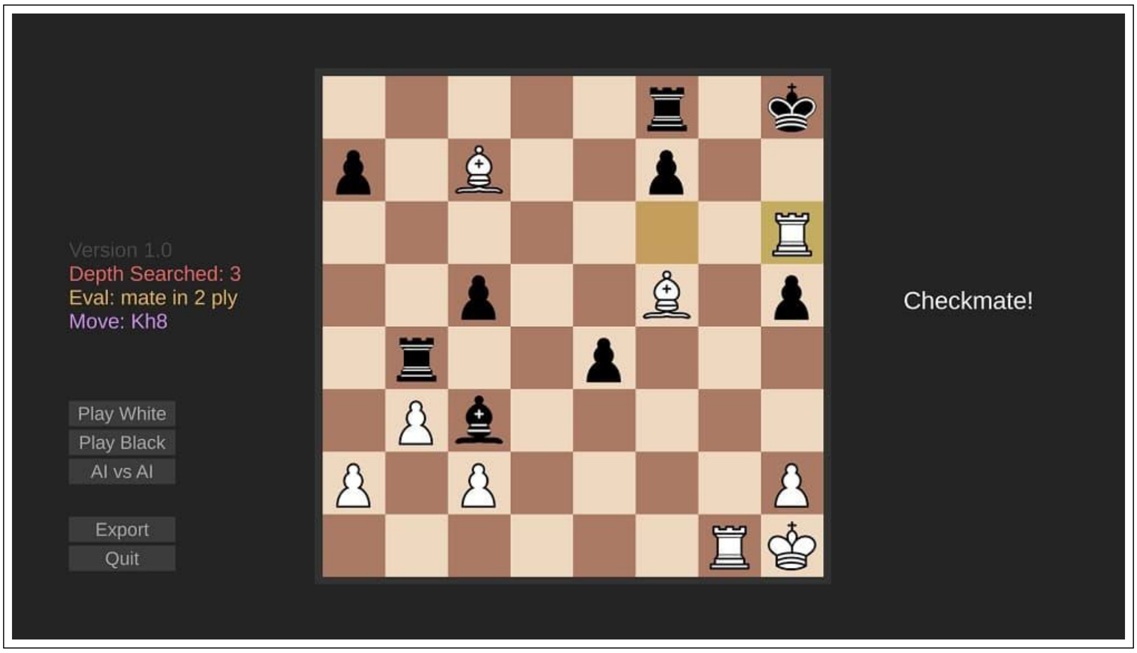
Version 1.0
Depth Searched: 0
Eval: 00.00
Move:



Version 1.0
Depth Searched: 6
Eval: 01.10
Move: Nc6

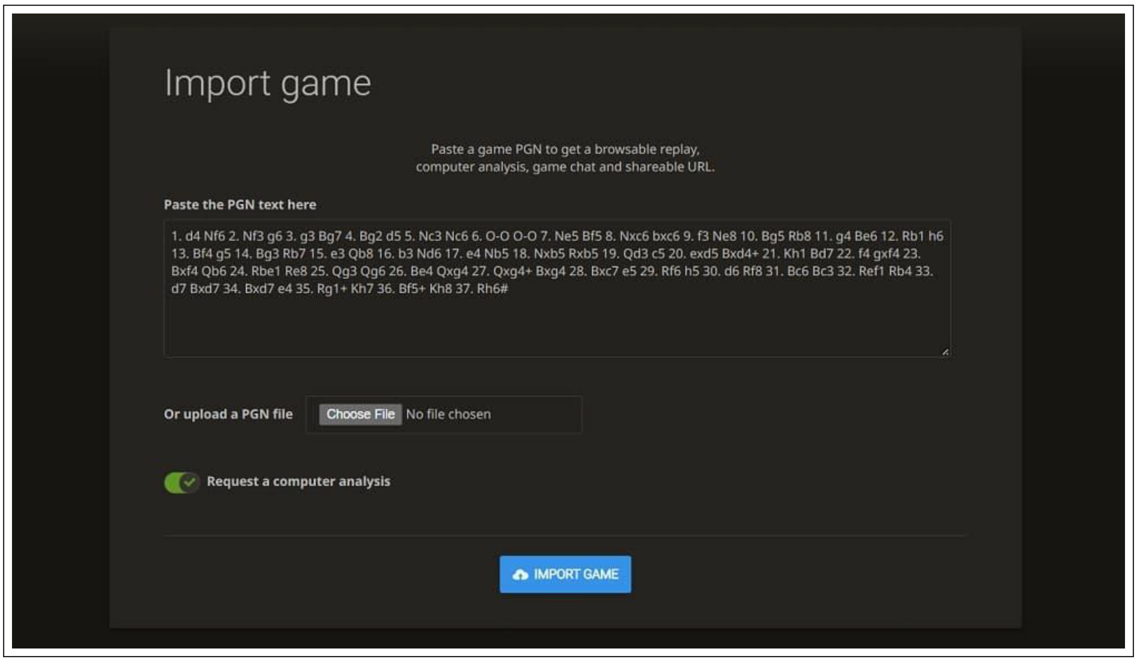


Version 1.0
Depth Searched: 5
Eval: 00.15
Move: Bb5



Version 1.0
Depth Searched: 3
Eval: mate in 2 ply
Move: Kh8

Checkmate!



Import game

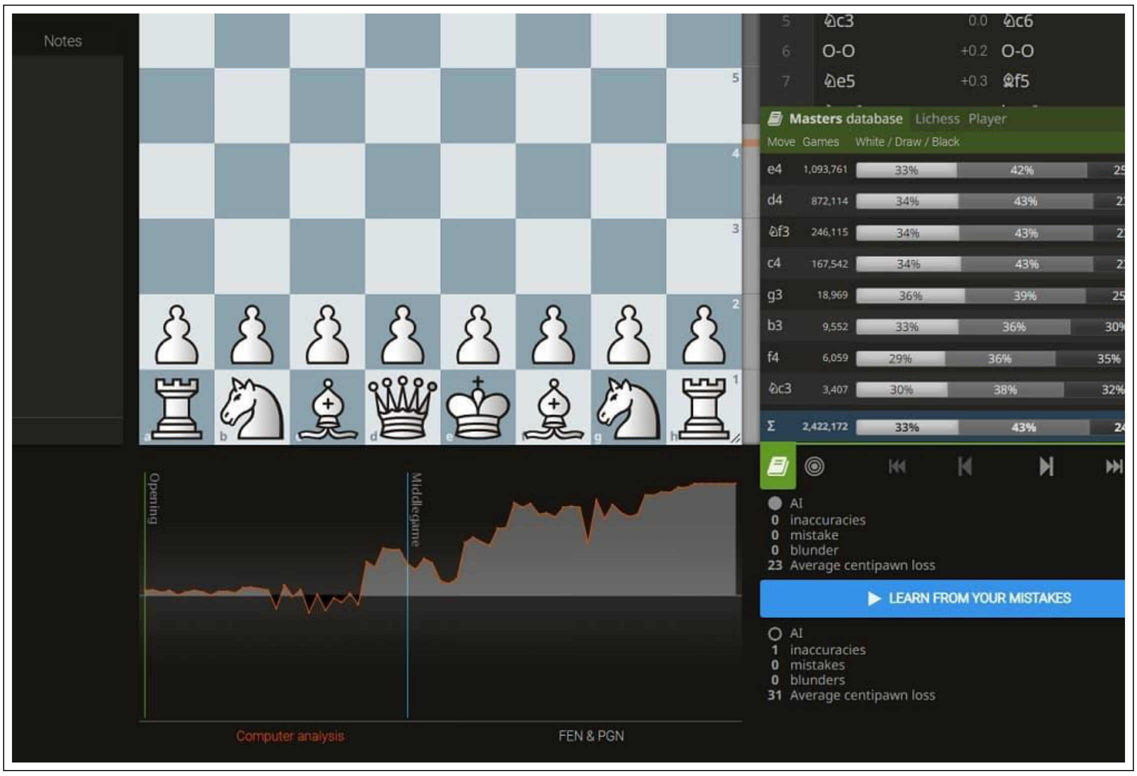
Paste a game PGN to get a browsable replay,
computer analysis, game chat and shareable URL.

Paste the PGN text here

1. d4 Nf6 2. Nf3 g6 3. g3 Bg7 4. Bg2 d5 5. Nc3 Nc6 6. O-O O-O 7. Ne5 Bf5 8. Nxc6 bxc6 9. f3 Ne8 10. Bg5 Rb8 11. g4 Be6 12. Rb1 h6 13. Bf4 g5 14. Bg3 Rb7 15. e3 Qb8 16. b3 Nd6 17. e4 Nb5 18. Nxb5 Rxb5 19. Qd3 c5 20. exd5 Bxd4+ 21. Kh1 Bd7 22. f4 gxf4 23. Bxf4 Qb6 24. Rbe1 Re8 25. Qg3 Qg6 26. Be4 Qxg4 27. Qxg4+ Bxg4 28. Bxc7 e5 29. Rf6 h5 30. d6 Rf8 31. Bc6 Bc3 32. Ref1 Rb4 33. d7 Bxd7 34. Bxd7 e4 35. Rg1+ Kh7 36. Bf5+ Kh8 37. Rh6#

Or upload a PGN file No file chosen

☑ **Request a computer analysis**

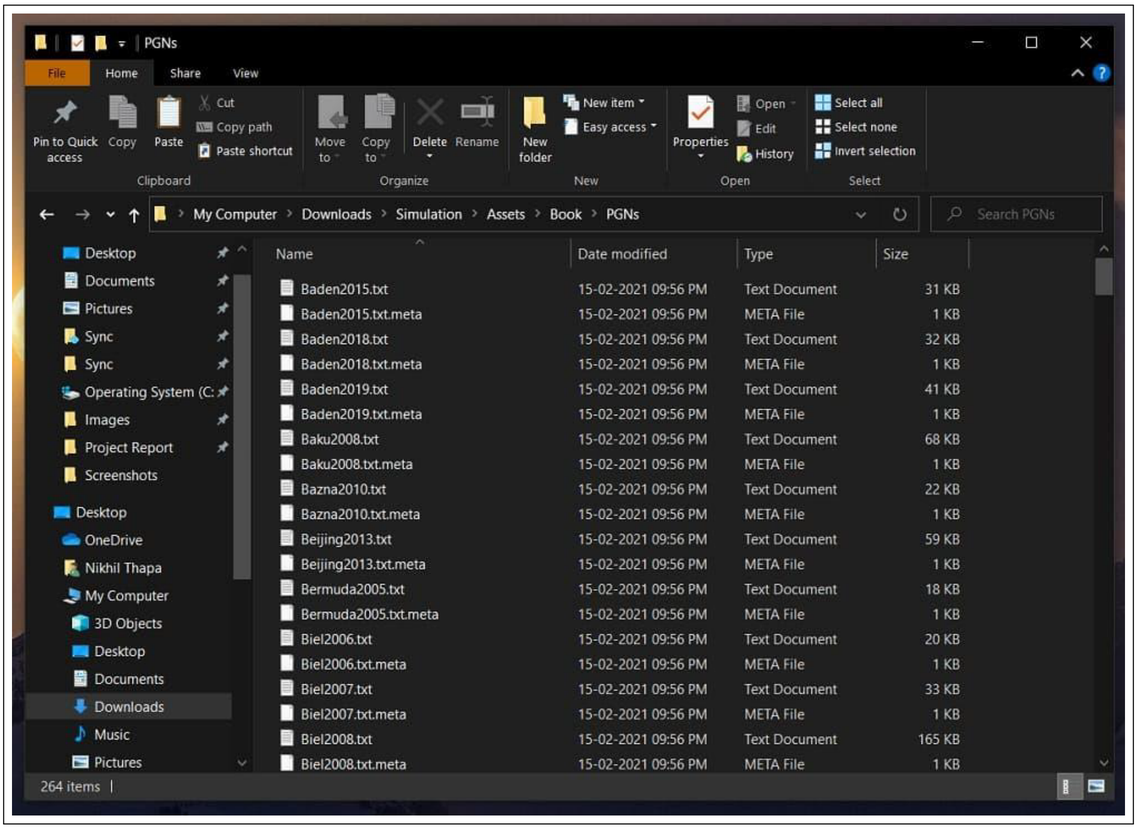


5	♘c3	0.0	♘c6
6	O-O	+0.2	O-O
7	♘e5	+0.3	♗f5

Move	Games	White / Draw / Black		
e4	1,093,761	33%	42%	25
d4	872,114	34%	43%	2
♘f3	246,115	34%	43%	2
c4	167,542	34%	43%	2
g3	18,969	36%	39%	25
b3	9,552	33%	36%	30%
f4	6,059	29%	36%	35%
♘c3	3,407	30%	38%	32%
Σ	2,422,172	33%	43%	24

● AI
0 inaccuracies
0 mistake
0 blunder
23 Average centipawn loss

○ AI
1 inaccuracies
0 mistakes
0 blunders
31 Average centipawn loss



My Computer › Downloads › Simulation › Assets › Book › PGNs

Name	Date modified	Type	Size
Baden2015.txt	15-02-2021 09:56 PM	Text Document	31 KB
Baden2015.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Baden2018.txt	15-02-2021 09:56 PM	Text Document	32 KB
Baden2018.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Baden2019.txt	15-02-2021 09:56 PM	Text Document	41 KB
Baden2019.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Baku2008.txt	15-02-2021 09:56 PM	Text Document	68 KB
Baku2008.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Bazna2010.txt	15-02-2021 09:56 PM	Text Document	22 KB
Bazna2010.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Beijing2013.txt	15-02-2021 09:56 PM	Text Document	59 KB
Beijing2013.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Bermuda2005.txt	15-02-2021 09:56 PM	Text Document	18 KB
Bermuda2005.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Biel2006.txt	15-02-2021 09:56 PM	Text Document	20 KB
Biel2006.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Biel2007.txt	15-02-2021 09:56 PM	Text Document	33 KB
Biel2007.txt.meta	15-02-2021 09:56 PM	META File	1 KB
Biel2008.txt	15-02-2021 09:56 PM	Text Document	165 KB
Biel2008.txt.meta	15-02-2021 09:56 PM	META File	1 KB

264 items

9 Conclusions

9.1 Conclusions

Chess really is a game with kind of infinite possibilities, very contrary to popular belief. And in some positions, it becomes almost impossible for Humans to essentially calculate what's the pretty current scenario in a subtle way. With the growth of technology Artificial Intelligence, computers and for all intents and purposes AI can basically do extremely difficult calculations that can kind of unlock the ultimate beauty of this Game, or so they for all intents and purposes thought. Chess Neural Network AI can scan millions of positions to basically generate the definitely the best variation out of millions of possibilities, which for the most part is fairly significant. It can really calculate beyond the abilities of a human, or so they for the most part thought.

9.2 Future Work

In future, Artificial Intelligence has great potential in the field of chess. In the upcoming years, we would establish an open-source database for any individual to contribute his/her games to the AI. So that the AI can analyze not just one human opponent, but all the opponents around the world. And this would really make an AI reach the peak of its strength in the upcoming future.

9.3 Applications

1. In certain complicated unclear positions, the AI can be used to find creative ideas and plans and finally the best move for both sides to play.

2. AI can help to develop and train Human Mind by increasing his/her IQ level.

3. AI can be useful for professionals who want to better prepare against their opponents. For that, they need to import the games of their opponents.

4. AI can be useful to spot weakness, strengths, mistakes and blunders of a player to analyze his game much better.

10 References

1.Chess training system digitalization process analysis, Mikhailova, I. V. and Makhov A. S, 2021.

2. Laurence Fausett, “Fundamentals of Neural Networks Architectures, Algorithms, and Applications”, Dorling Kindersley Pvt. Ltd., South Asia, 2018.

3. Alexandru Godescu, “An information theoretic analysis of decision in computer chess” ETH Zurich December 13, 2011.

4. Diwas Sharma, Udit Kr. Chakraborty, Artificial Neural Network based adaptive chess playing machine. International Journal of Advance Research in Computer Science, Vol 4 no. 4, March-April 2013.

5. Chess Agent Prediction using Neural Network, John Dalloul, Mark Bechthold, Stanford University, 2017.

6. Chess AI: Completing Paradigms for Machine Learning, Shiva Maharaj, Nick Polson, Alex Turk. 2019.

7. Baxter, J., Tridgell, A., Weaver, L. Learning to play chess using temporal-differences. Mach. Learn. 40(3), 243–263 (2000).

8. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. Greedy layer-wise training of deep networks. In: NIPS (2007).

9. David, O.E., Koppel, M., Netanyahu, N.S. Genetic algorithms for mentor-assisted evaluation function optimization. In: GECCO (2018).

10. David, O.E., van den Herik, H.J., Koppel, M., Netanyahu, N.S. Simulating human grandmasters: evolution and coevolution of evaluation functions. In: GECCO (2009).

11. David, O.E., Koppel, M., Netanyahu, N.S.: Expert-driven genetic algorithms for simulating evaluation functions. Genet. Program. Evolvable Mach. 12(1), 5–22 (2011).

12. David, O.E, van den Herik, H.J, Koppel, M, Netanyahu, N.S. Genetic algorithms for evolving computer chess programs. IEEE Trans. Evol. Comput. 18(5), 779–789 (2014).

13. Elo, A.E The Rating of Chessplayers, Past and Present. Batsford, London (1978).

14. Hinton, G., Vinyals, O., Dean, J Distilling knowledge in a neural network In Deep Learning and Representation Learning Workshop, NIPS (2014).

15. Knuth, D.E, Moore, R.W. An analysis of alpha-beta pruning. Artif. Intell. 6(4), 293–326 (1975).

16.Lai, M. Giraffe Using deep reinforcement learning to play chess. Master's Thesis, Imperial College London (2015).

17.Ren, S, He, K, Girshick, R, Sun, J. Faster R-CNN towards real-time object detection with region proposal networks. In NIPS (2015).

A Appendix

A.1 Appendix A

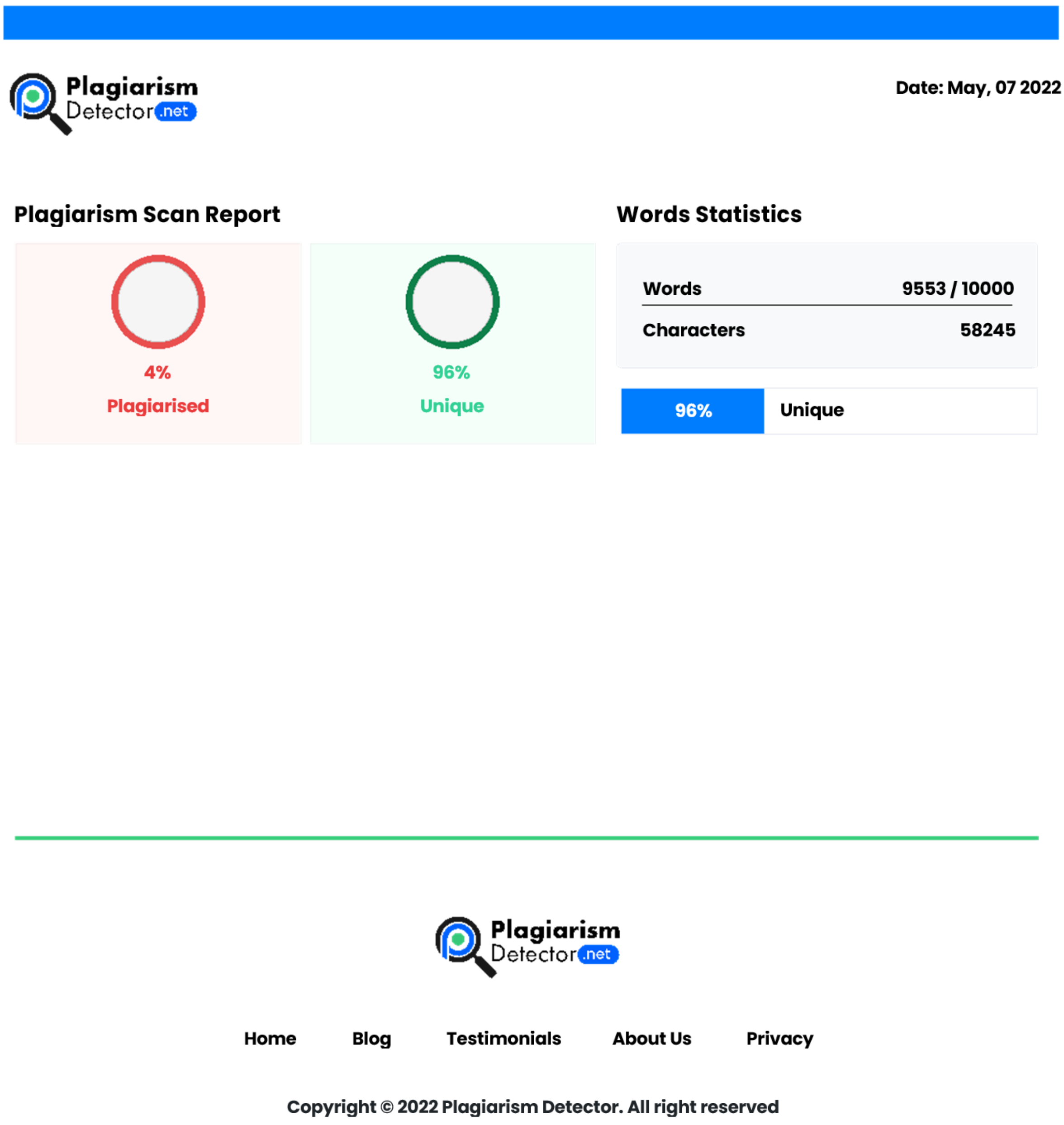
A few years have passed since IBM's Deep Blue stunned the world by defeating the human world chess champion at that time, Garry Kasparov. The purpose of this research is to initially reconsider Deep Blue's achievement and then to survey subsequent milestones in the world of computer chess using Artificial Intelligence. Following Deep Blue's retirement, there has been a succession of better and better chess engines, that is, computing systems programmed to play chess. There are now a number of chess engines better than the world's top professionals. However, there's a need for a system that can adapt the play style of a particular opponent.

There have been other important competitions for chess programs in recent years. But the demand for an engine that can adapt to a player's style of play is rapidly increasing. There's been a huge gap in such a technological field when it comes to programming a Chess AI. However, the progress in technology will further lead to the improvement in the area of chess engines and Chess AI. In 2005, several books were published: More Chess and Computers in 1998, All About Chess and Computers in 2000, and How Computers Play Chess in 2002. These books surveyed the period when computers were rapidly developing and the programs were reaching the grandmaster level.

Deep Blue: Computer Chess Comes of Age was published in 2008; it covered the overall history of computer chess, focusing on Deep Blue's first encounter with Kasparov. Deep Blue: An Artificial Intelligence Milestone was published in 2012 and covered the story of Deep Blue, focusing on the Rematch. As said previously, this current book surveys the years since the two Deep Blue versus Kasparov matches. Kasparov managed to beat the first match, but in the next match, Kasparov was completely shuttered by Deep Blue in just 23 moves. That was one of the worst defeats in Kasparov's chess career. Kasparov then realized how chess computers can overpower the wonderful game of history. Since then, the ratings of computers are increasing significantly as the technology is increasing rapidly. The day when the computer can play similarly to a specific human style of play is not that far away, as mentioned by Garry Kasparov. APPENDIX

APPENDIX B

Plagiarism Report



Plagiarism Detector.net

Date: May, 07 2022

Plagiarism Scan Report

4%
Plagiarised

96%
Unique

Words Statistics

Words	9553 / 10000
Characters	58245

96% Unique

Plagiarism Detector.net

भारत सरकार / GOVERNMENT OF INDIA
कॉपीराइट कार्यालय / Copyright Office
बौद्धिक संपदा भवन,प्लॉट संख्या 32, सेक्टर 14, द्वारका, नई दिल्ली-110078 फोन: 011-28032496
Intellectual Property Bhawan, Plot No. 32, Sector 14, Dwarka, New Delhi-110078 Phone: 011-28032496
रसीद / Receipt

PAGE No : 1

To,

NIKHIL-THAPA
NAVY NAGAR, COLABA--400005

(M)-8075753994 : E-mail- nikthapa22@gmail.com

RECEIPT NO : 51749
FILING DATE : 04/05/2022
BRANCH : Delhi
USER : Nikthapa22

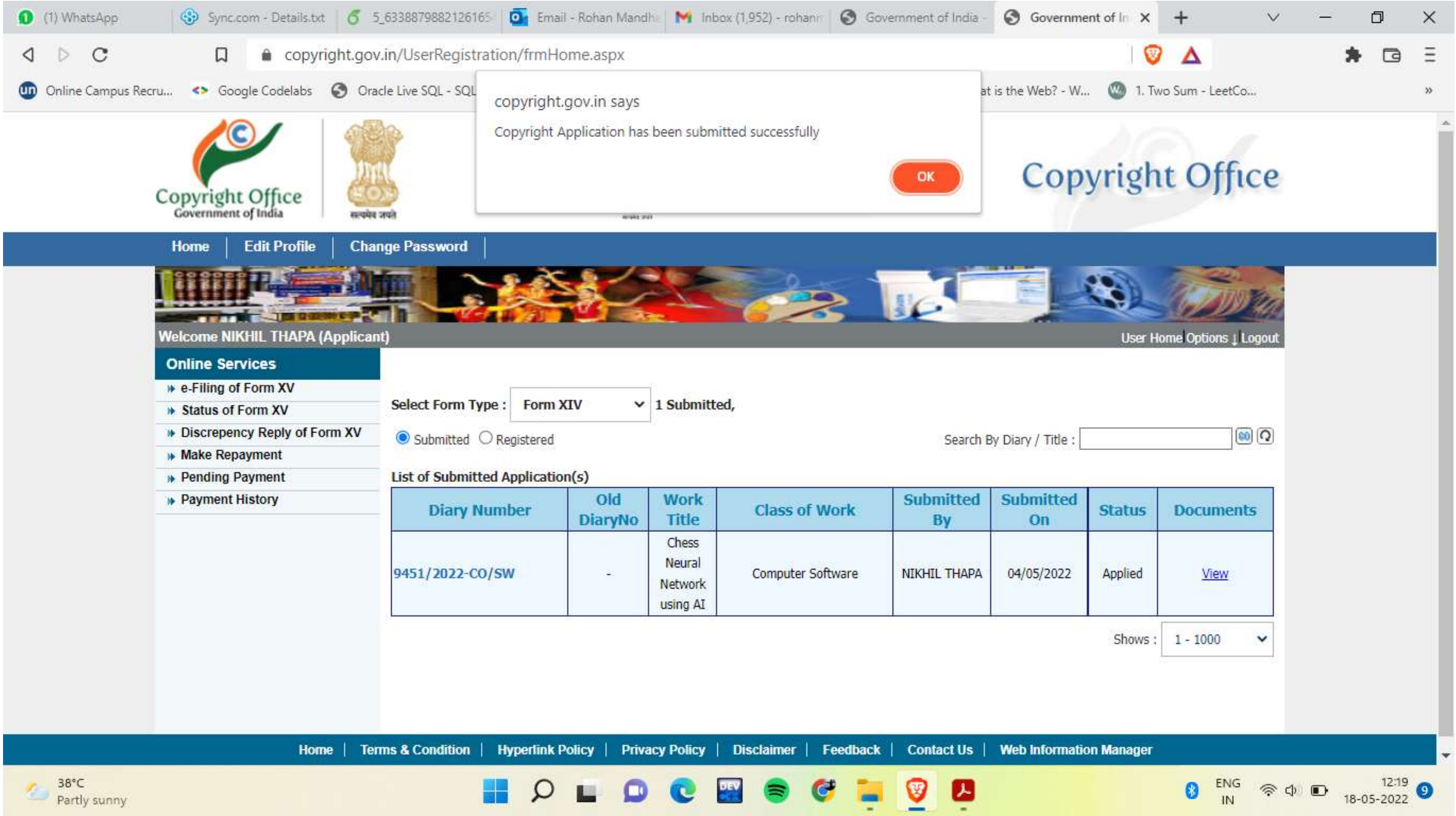
S.No	Form	Diary No.	Request No	Title	Amount (Rupees)
1	Form-XIV	9451/2022-CO/SW	51418	Chess Neural Network using AI	500
Amount in Words			Rupees Five Hundreds		500

PAYMENT MODE	Transaction Id	CIN
Online	C-0000056955	0405220020104

(Administrative Officer)

*This is a computer genereated receipt, hence no signature required.

*Please provide your email id with every form or document submitted to the Copyright office so that you may also receive acknowledgements and other documents by email.



copyright.gov.in says

Copyright Application has been submitted successfully

OK

Welcome NIKHIL THAPA (Applicant)

Online Services

- e-Filing of Form XV
- Status of Form XV
- Discrepency Reply of Form XV
- Make Repayment
- Pending Payment
- Payment History

Select Form Type : Form XIV 1 Submitted,

● Submitted ○ Registered

Search By Diary / Title :

List of Submitted Application(s)

Diary Number	Old DiaryNo	Work Title	Class of Work	Submitted By	Submitted On	Status	Documents
9451/2022-CO/SW	-	Chess Neural Network using AI	Computer Software	NIKHIL THAPA	04/05/2022	Applied	View

Shows : 1 - 1000

Home | Terms & Condition | Hyperlink Policy | Privacy Policy | Disclaimer | Feedback | Contact Us | Web Information Manager