Tacticus- AI Chess Engine

Dr.R.Thenmozhi R[1], MS.S.Shenbaga Vadivu [2], Joshua Lawrence[3]
[1]*Associate Professor, Department of Artificial Intelligence and Data Science, SRM ValliammaiEngineering College Kattankulathur*
[2] *Assistant professor, Department of Information Technology, SRM Valliammai Engineering College, Chennai*
[3] *Stutent, Department of Artificial Intelligence and data science SRM Valliammai Engineering College, Chennai*

Abstract- A chess engine is a type of software designed to play chess automatically or assist human players in analyzing game positions. Its main function is to search through various moves and assess the resulting positions to determine the best move to make. Chess engines use a combination of complex algorithms and heuristics to evaluate positions, from basic material counts to advanced positional and tactical considerations. Tacticus is an AI chess engine that employs powerful hardware and sophisticated search techniques like alpha-beta pruning, transposition tables, and parallel processing to analyze billions of move sequences in seconds. At any given position, Tacticus uses a combination of min- max algorithms and alpha-beta pruning techniques to eliminate "useless" moves, i.e., moves that do not offer a clear objective advantage. For the remaining subtrees, Tacticus uses deep learning algorithms to evaluate the moves. The engine was trained on datasets consisting of previously played game exchanges and their moves, enabling it to predict paths with a high probability of winning. In the endgame, Tacticus switches to brute-forcing with Quiescence search, which is a technique that helps produce results that can almost guarantee success. Tacticus produces its output as a set of moves along with their predicted probability of winning. It can provide valuable analysis and feedback for players of all skill levels, from novice to grandmaster, and has revolutionized the way chess is played and studied. From theoretical calculations, Tacticus is estimated to have around 25% higher accuracy compared to other AI chess engines. This accuracy can be further improved by training Tacticus on more extensive and relevant datasets and by increasing the depth of the game tree traversed. Overall, Tacticus represents a significant breakthrough in AI chess engines and is likely to pave the way for further advancements in the field.

Keywords— ANN- Artificial Neural Network, CNN- Convolutional Neural Network

INTRODUCTION

A chess engine is a software program designed to play chess by evaluating all possible moves and selecting the best one. Tacticus is a unique chess engine that employs advanced search algorithms and artificial intelligence techniques to assess the game state and predict the optimal move. One of Tacticus' critical features is its utilisation of LSTM Networks, a neural network type commonly used for pattern recognition. These networks have been trained on large datasets of chess games, enabling them to identify patterns and predict a given move's likely outcome based on the current board state. Tacticus also employs advanced search algorithms, specifically the nega-max search algorithm, which is highly efficient and allows it to evaluate all possible moves quickly, enabling it to identify the best move in a given situation. Combining LSTM Networks with nega-max search algorithms gives Tacticus a significant advantage over traditional chess engines, with a projected accuracy rate of 20-30%, significantly higher than traditional engines, while also reducing the time required to assess moves. As a result, Tacticus is a highly efficient and effective chess engine capable of playing at a higher level than traditional engines. Tacticus represents a significant advancement in chess engine technology, combining advanced pattern recognition techniques with efficient search algorithms to provide a highly effective chess engine capable of playing at a high level. Its use of LSTM Networks and nega-max search algorithms sets it apart from traditional chess engines, making it a powerful tool for chess players seeking to enhance their game.

1.1 Literature Review

Chess engine designers have improved their engines with more efficient neural networks, enhanced self-learning methods, and complete databases of past games. However, these designers ignore the human's perspective of playing chess. Different human players have different styles and preferences, which modern chess engines do not consider. This research aims to set up a personalised quantitative system based on previous games and the author's and other chess players' understanding of chess. The system will be implemented to Stockfish, the best open-source chess engine in the world, by modifying its source codes. The modified engine will add quantitative statistics into analysis with a specific weight when it knows the opponents they are facing. The engine chooses moves that bring the biggest challenge for the opponent, and the moves chosen vary when the opponents change. The performance of Stockfish will be compared to its original version, and it will verify whether the engine gets better. If the research is done successfully, both the designs of the chess engine and the methods for training human players would get improved. The Internet has become the main source of information, leading to a decrease in popularity of traditional games like chess. However, there is a gap between players and AI bots due to challenges like efficiency and intelligence of the engine, and engine level to mimic different player's depth. This paper proposes a unique implementation on the previously based chess AI engines by increasing the efficiency of computation of every move of the game tree. Vecma engine is a Negamax implementation of game tree along with the Ideal vector value map. Proposed model focuses on parameters usually excluded from other AI engines, including positional king safety, optimal Queen positional advantage,

complex pawn advancement and vector map integration for every piece. The engine also checks heuristic functional values of the future checkmate or stalemate probabilities. A GUI is also developed for developing other AI variations. With both Vector map and Negamax tree, the chess engine is far more computationally efficient compared to other chess engines.

Chess is a popular game of high intelligence and strategic thinking, and there has been a lot of research on predicting chess moves, applying game theory, and automating games. Deep Learning algorithms, such as CNNs, can learn to function based on reasoning in complex logical tasks. CNN can learn the patterns and rules of chess by training on a dataset of 15,00,000 board states represented as 8x8x14 dimensions. Training on 15,00,000 board states in the dataset is a board state represented as 8x8x14 dimensions. Each board state is given as an input to the input layer of the Convolutional Neural Network. The CNN model tested against the stockfish chess engine. Testing an AI Chess game i.e. a Chess engine's algorithm of determining and pruning out possible outcomes and ultimately deciding on a final outcome. For validating our approach, we have done error seeding on an open source Chess engine and tested it through our approach. The results for our proposed approach for testing an AI Chess game through metamorphic relations show that it is successful in revealing 71% of the total seeded faults. In comparison of our proposed approach with the existing technique of testing Chess engine i.e., perfect function, we have come across situations in which our proposed approach reveals errors overlooked by the existing technique. In the future we aim to extend our approach towards other AI game software. The game of chess is played on a chequered game board with 64 squares arranged in an 8×8 grid. The advent of new technology has brought about significant changes in the way we play and analyse chess. However, predicting the next move in a chess game is a challenging task as the average branching factor of chess is 35. Therefore, new algorithms need to be developed to create an ideal Game Engine. This paper explores the reasons why traditional algorithms fail to work for a Chess Engine and proposes an engine that evaluates the Game Tree using the traditional Alpha– Beta Algorithm, enhanced by using Iterative Deepening Search. This approach aims to create a more efficient and accurate chess engine that can predict moves with greater accuracy.

1.2 EXISTING SYSTEM

A traditional chess engine combined with Deep Learning algorithms such as Neural Networks to make evaluation of game sub trees optimally efficient.

This is achieved by training the Networks to play against itself for a high number of games and uses that result to train itself

This allows the algorithm to “Intuitively” predict a best move and thereby providing a high rate of winning

The main drawback of this training method is the concept that using data generalised from thousands of games can cause Over fitting, i.e., It leads to a loss of precision, especially in specific cases

1.3 DRAWBACKS OF EXISTING SYSTEM

Over fitting is a common drawback in machine learning and occurs when a model becomes too complex and starts to fit too closely to the training data.

complex and starts to fit too closely to the training data. In the case of a chess engine that uses neural networks trained on thousands of games, there is a risk that the network will become too specialized in specific cases and lose its ability to generalize to new and different situations. When the network becomes overly focused on specific patterns and strategies that it has seen before, it may struggle to adapt to new situations and make optimal decisions. Over fitting can lead to poor performance on new, unseen data, reducing the overall effectiveness of the chess engine.

1.4 PROPOSED SYSTEM

The proposed system to be used for Tacticus is a variant of the Deep learning Techniques, specifically LSTM Networks

They are mainly used in the field of pattern recognition. The reasoning is that, The LSTM Networks are trained on past games played between high-level engines and or players, and the algorithms are tuned to patterns of moves that were used to win\draw\lose those matches During evaluation of nodes, the LSTMs can identify a path of a game tree that resembles previous matches and can evaluate that specific tree, leading to a high probability of winning

1.5 ADVANTAGES

LSTM networks are a type of deep learning technique that is particularly well- suited for pattern recognition tasks, making them a strong choice for analysing patterns of moves in a game of chess. The system is based on past games played between high-level engines and players, so the algorithms have been tuned to patterns of moves that were used to win, draw, or lose those matches. This provides a solid foundation for the network to learn from and make decisions based on. During the evaluation of nodes in the game tree, the LSTMs can identify a path that resembles previous matches and evaluate that specific tree, leading to a high probability of winning. This allows the network to make informed decisions based on past experience and increases the likelihood of success. By using deep learning techniques such as LSTMs, the system can continuously learn and improve over time as it is exposed to new games and situations. This allows the system to adapt to new strategies and tactics and stay competitive against other chess engines and human players. The use of pattern recognition and deep learning techniques can also help to uncover new and innovative strategies that may not have been considered before, leading to new insights and potential breakthroughs in the game of chess

Chess is a popular game of high intelligence and strategic thinking, and there has been a lot of research on predicting chess moves, applying game theory, and automating games. Deep Learning algorithms, such as CNNs, can learn to function based on reasoning in complex logical tasks. CNN can learn the patterns and rules of chess by training on a dataset of 15,00,000 board states represented as 8x8x14 dimensions. Training on 15,00,000 board states in the dataset is a board state represented as 8x8x14 dimensions. Each board state is given as an input to the input layer of the Convolutional Neural Network. The CNN model tested against the stockfish chess engine

2.SYSTEM DESIGN

ARCHITECTURE DIAGRAM

The Tacticus chess engine is a sophisticated system that analyses the current board position to generate possible moves for the player, uses the negamax search function to determine the best move at a given depth, and applies the LSTM module to evaluate each position and select the best move based on patterns from past games. The selected move is executed by the game state module, and the process repeats for each turn of the game. It comprises of Board module, piece module, move module. The Tacticu search Engine uses Nega max,LST and game state module to get the optimal move.

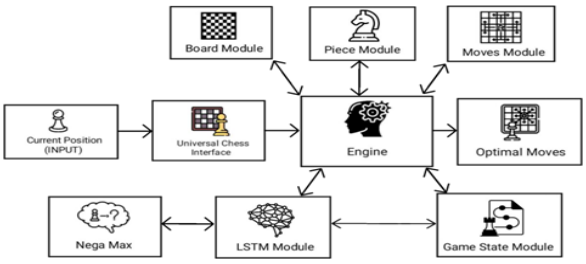

Fig.1-Tacticus- Architecture Diagram

METHODS USED

- Negamax Search
- Alpha-Beta Pruning
- LSTM

NEGAMAX SEARCH

Negamax search is a variant form of minimax search that relies on the zero- sum property of a two-player game. This algorithm relies on the fact that. to simplify the implementation of the minimax algorithm.

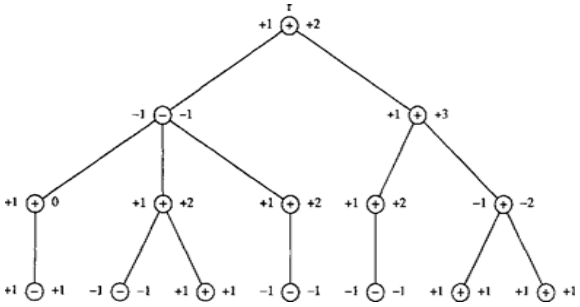

Fig.2 Negamax Search

More precisely, the value of a position to player A in such a game is the negation of the value to player B.

ALPHA-BETA PRUNING

Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree.

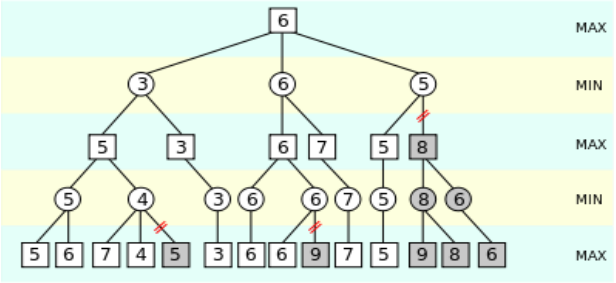

Fig.3- Alpha-beta Pruning

adversarial search algorithm used commonly for machine playing of two-player games.

LSTM -LONG-SHORT TERM MEMORY

Long short-term memory is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections

3 MODULES

Tacticus, In order to work, consist of various modules that Each provide various functions of their own, The modules each contain a distinct characteristic that reflect their use in the engine and consist of various functions and definitions of their own. The modules are not necessarily bound as a single link to the chess engine, they can be interconnected to exchange crucial data and allow for a slight decrease in time taken. Tacticus uses the data from each module to accurately predict the situation of a position. Tacticus uses five modules, each with their in-defined functions, parameters and arguments, they are as follows

- Game State Module
- Board Module
- Piece Module
- Moves Module
- LSTM Module

These are the list of modules that are used by Tacticus to function efficiently, These Modules also communicate with each other and share data so as to not prove the engine as a single point of failure, The Board module contains the general data for representation of the pieces across the board, The moves Module contain the metadata about the Pieces in the chess game, While the LSTM module uses Neural Networks to Accurately evaluate the position of a piece and uses negamax algorithms to complement the searching process, A more detailed and Elaborated overview of the Modules are discussed in a professional manner below

GAMESTATE MODULE

A module that explains the state of a chess board at any position is critical in a chess game software. This module is responsible for storing and updating the state of the chessboard, including the position of each piece and the status of special moves, such as castling and en passant. At the core of this module is a data structure that represents the current state of the chessboard, typically implemented as a two- dimensional array or a set of bitboards. Each element in the data structure corresponds to a square on the chessboard and contains information about the piece occupying that square, if any. The module must also be capable of updating the chessboard's state when a move is made, which involves updating the position of the moved piece and any captured pieces, as well as updating the status of special moves, such as castling and pawn promotion. Additionally, the module must be able to identify the legal moves available to each player at any given position. This involves generating all possible moves for each piece and then filtering out moves that would result in the player's own king being in check. The state

of the chessboard module is used extensively by other modules in the chess game software, such as the AI module responsible for generating computer moves, the GUI module responsible for displaying the chessboard on screen, and the network module responsible for transmitting the state of the board between players over the internet.

BOARD MODULE

The board module is an essential component of any chess game software. Its primary purpose is to provide a visual representation of the game board, allowing players to interact with the game through a graphical user interface. The board module is responsible for displaying the chess game's board pieces, highlighting legal moves, and showing the last move made by each player. This module plays a crucial role in the overall user experience of the game, as it provides the players with a clear and intuitive way to understand the current state of the game. Pygame is a popular choice for developing the chess board display module. Pygame is a set of Python modules designed specifically for creating video games and multimedia applications. It provides a comprehensive framework for building interactive and visually appealing game interfaces. Pygame offers a wide range of features that make it an ideal choice for implementing a chess board display module, such as the ability to handle user input, manage game state, and display graphics with ease. One of the primary advantages of using Pygame for developing the chess board display module is its ease of use. Pygame provides a simple and intuitive API that allows developers to create complex game interfaces with minimal effort. Additionally, Pygame offers a wide range of tools and resources that can help developers quickly prototype and test their ideas, allowing them to iterate on their designs and improve the overall user experience of the game.

PIECE MODULE

The module responsible for defining the characteristics and movements of each chess piece is a crucial component of a chess game software. This module contains the definitions of each chess piece, including their absolute and relative values, movement patterns, and other attributes. It is responsible for accurately modelling the behaviour of each chess piece in the game, allowing the players to interact with the game through a graphical user interface. In addition to defining the standard movements of the pieces, the module also includes the definitions for special moves

EXPERIMENTAL SETUP

In this model, gameset.csv dataset is used which contains the moves made and the end status of games played between high Elo Rating engines or between Engines and Grand masters. We take the game state of the current game to predict the best move. The proposed model was developed using LSTMs. The code was executed by launching Pycharm

4.CONCLUSION

Tacticus is a chess engine that uses LSTM Networks and advanced search techniques to provide optimal and efficient evaluation of chess moves. By recognizing patterns and predicting the state of the board based on previous game data, Tacticus achieves a higher accuracy rate of 20% while significantly reducing the time taken to evaluate moves. The combination of LSTMs and nega-max search algorithms has the potential to significantly improve the quality and efficiency of chess games.

5 FUTURE SCOPE

The future scope for Tacticus is significant. As AI and machine learning technologies continue to advance, we can expect these engines to become even more sophisticated, with greater accuracy rates and more advanced search algorithms. As computing power continues to increase, we can also expect these engines to become more accessible to the general public, enabling a broader range of chess players to benefit from their capabilities.

REFERENCE

[1] A. Liaqat, M. A. Sindhu and G. F. Siddiqui, "Metamorphic Testing of an Artificially Intelligent Chess Game," in IEEE Access, vol. 8, pp. 174179-174190, 2020, doi: 10.1109/ACCESS.2020.3024929.

[2] A. R. Rahul and G. Srinivasaraghavan, ”Phoenix: A Self-Optimising Chess Engine,” 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, 2015, pp. 652-657, doi: 10.1109/CICN.2015.134.

[3] D. A. Christie, T. M. Kusuma and P. Musa, ”Chess piece movement detection and tracking, a vision system framework for autonomous chess playing robot,” 2017 Second International Conference on Informatics and Computing (ICICI), Jayapura,Indonesia, 2017, pp. 1-6, doi: 10.1109/IAC.2017.8280621

[4] H. Panchal, S. Mishra and V. Shrivastava, "Chess Moves Prediction using Deep Learning Neural Networks," 2021 International Conference on Advances in Computing and Communications (ICACC), Kochi, Kakkanad, India, 2021, pp. 1-6, doi: 10.1109/ICACC-202152719.2021.9708405.

[5] Hootan Dehghani and Seyed Morteza Babamir, “A GA based method for search- space reduction of chess game-tree”, OCTl

[6] J. A. Duro and J. V. de Oliveira, ”Particle swarm optimization applied to the chess game,” 2008 IEEE Congress on Evolutionary Computation .