International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Artificial Intelligence based Chess Solver

Rohit Verma[1], Sanchit Verma[2],Saransh Maddhesia[3], Shipra Srivastava[4]

[4]Assistant Professor of Information Technology Engineering, Greater Noida Institute of Technology
[1,2,3] Student, Department of Information Technology Engineering, Greater Noida Institute of Technology

---

Abstract –

Numerous published studies have revealed that various researchers have tried to create a system that learns to play cognitive games, given little or no information about the law of the game. A standard chess machine carefully scans the moving possibilities from the chessboard configuration to choose which best move can be made. The vicious search method used by the Deep Blue chess engine has had a huge impact on the world of artificial intelligence, but it is still found hungry for resources. This paper, with the concept of Artificial Neural Networks introduces a simple and effective way to develop a clever chess engine that can help and demonstrate possible movement within the game using an evolutionary and flexible computer learning system for human grandmasters.

Keywords

Artificial Neural Network, Championship game, legal move, Octavius.

1. INTRODUCTION

Chess, known as the game of "perfect information", since both players are conscious about the entire situation of the game at all time, just by looking at the board, has since been considered as standard of testing intelligence. Artificially intelligent programs have been developed by such computing majors as IBM, specifically for the purpose of playing chess, and after Deep Blue, (Mid 1990's) [1], defeated the world chess champion Garry Kasparov, machines have been consistently beating man in the rapid version of the game. The kind of victory of a machine over a human grand master has led to make chess one of the most used games to promote artificial intelligence and Computer Science. Nevertheless, chess engines are not able to strategically plan moves or to explain why they compute such a combination of moves. This limitation in the power of the machine exists due to the fact that the decision tree based hardware and software takes too much time to search the exhaustive option set for the correct move.

It has been found that Neural Networks are rarely used to solve board games. The main focus is given only on the game-ending stages but a few uses of Neural Network in the computer game chess as a whole. The ability to play top human chess players is due to their ability to plan strategies, look forward, understand and create. The ability to play computers, however, is largely based on their speed. It would be great to find out how a machine can benefit from the way a human player organizes a game. There has been little published work looking at Artificial Neural Network studies where external (personal) information is applied to decisive problems with large country spaces. The project is interested in investigating the site by examining whether the Artificial Neural Network is able to access any useful information in a very small area using the purist's method of presenting information. To achieve this goal the board game chess has been selected as the source of the problem.

2. SURVEY

The ancient methods of playing chess with machines have evolved over the years. Nicolas Lassabe et al [7] pointed out in their work that, the first approach was to test and finalize, in each chessboard configuration the best action you can play. This simple and logical method requires extensive comparisons and analysis. Von Neumann and Morgenstern (1944) propose minimax as a means of determining what action should be performed in chess.

[8] A two-player game like chess can be represented by a tree. Each node is a position on the board, starting at the root and starting point of the game. The vertices become a possible movement, leading to the following nodes (positions). The minimax algorithm is a basic search tool used in many game programs. Minimax works by building a depth-first, left-to-right tree, and alternate search for Max and Min's movement. The only method based on the Artificial Neural Network for building a chess engine is reported in the literature under the name Octavius. Octavius is trying to gain an understanding of chess through a brief analysis of the main games and the grandmaster. Its training is based on the assumption that any position achieved during such games must be above any other position available within that game.[5]

3. PROBLEM DEFINITION

To generate a function f, which take as input a chess board position denoted by [13],

$\wedge\ (\ \wedge(X_iB_{kj}),\wedge(Y_iB_{kj})\ )$

Such that,

$$f\Big(\wedge\ \big(\wedge\ (X_iB_{kj}),\wedge(Y_iB_{kj})\big)\Big) = X_iB_{kj} \text{ or } Y_iB_{kj}; \quad (1)$$

Here, X and Y denotes white and black pieces respectively, i

= 1,2,…,16 represents total number pieces for each side,k=1,2,…,8 denotes files and j=1,2,…,8 denotes ranks,Xi Bkj/YiBkj represents the current position of X (white) and Y (black) pieces accordingly.

4. METHODOLOGY

Figure 1 shows the flow sequence during the training phase where the input process taken from the grandmaster game is encrypted using the board-code system system discussed later. The coded movement is then integrated into a neural network designed where the system is trained to produce output. The system learns patterns (movements and techniques) produced by a human grandmother during their heroic games. The system is trained until it produces the result with a small error. Figure 02 shows the flow of the system designed during the test phases. Here the trained system is tested with the moves that are not trained earlier i.e. user fed unknown moves in unknown environment, where the system behaves as a move generator.

The moves so generated are validated to check whether they are the valid moves or not. It also checks the suggested move to meet the rules of the game.[13]

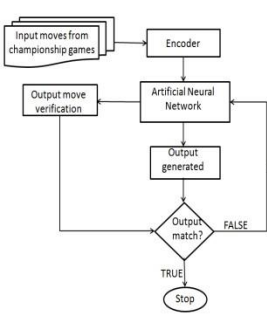


Fig 1: Training Phase

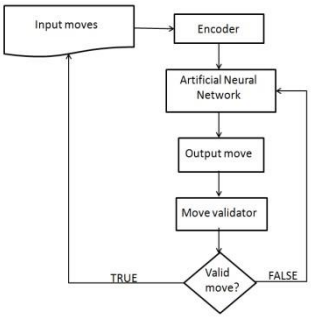


Fig 2: Testing Phase

5. IMPLEMENTATION

Algebraic notation is a method for recording and describing the moves in a game of chess. Each square of the chessboard is identified by a unique coordinate pair, a letter and a number. Each square has a unique identification of file letter followed by rank number. [14] Considering the fact that the Artificial Neural Network only understand the numeric input values those algebraic notations are to be preprocessed. Firstly, the ANN understandable board is to be obtained where each square in the chessboard is represented with the unique ANN understandable numerals, the idea here is the use of numerical notational technique of the board as shown in the figure 3,

8	18	28	38	48	58	68	78	88
7	17	27	37	47	57	67	77	87
6	16	26	36	46	56	66	76	86
5	15	25	35	45	55	65	75	85
4	14	24	34	44	54	64	74	84
3	13	23	33	43	53	63	73	83
2	12	22	32	42	52	62	72	82
1	11	21	31	41	51	61	71	81
	1	2	3	4	5	6	7	8

Fig 3: ANN understandable chessboard[13]

Table 1. Encoding scheme for ANN representation

White Piece		Black Piece	
Algebraic	Numeric	Algebraic	Numeric
e4	5254	d5	4745
Nf6	7163	Nf3	7866

In the numeric notation all the squares are numbered with a unique two-digit numbers. In this simple numeric coordinate system the file and the rank of the chess board is represented by the numerals 1 to 8 starting from left to right (white side) and bottom to top respectively. Hence, the individual square of the chess board can now be denoted by a uniquely identifiable co-ordinate value where first digit describes the file and the second one the rank. Hence, the move that would be written is as shown in the Figure 5.2.The encoded moves depict the combination of the current position of the piece as well as the next position to be moved together within the board. As shown in the Figure 4, the move e4 is represented as 5254 indicating that a piece is to be moved from position 52 to next position 54.The work tries to train the system by means of the patterns generated by the master and grandmaster during the championship games. [13]

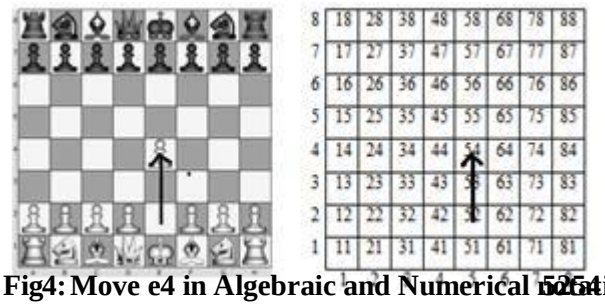


Fig4: Move e4 in Algebraic and Numerical notation

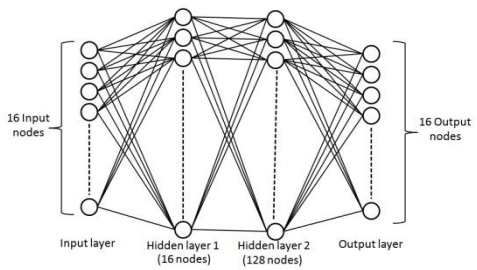


Fig 5: Proposed Neural Network Structure[13]

The use of adaptive Neural Network techniques allows us to train the system to construct a winning position, as well as to develop its pieces which result with the best possible move. The trained system is then deployed in the testing environment where the result will be examined based on its performance. The network structure is of high importance in such ANN based approaches, and the encoding of the problem into ANN understandable format is essential. The strategy is to have 16 inputs and 16 output system where each input connections signifies a piece of the one player (in order) and each output connection stands for a piece of the other player. Each input will be active for only one input, showing a particular piece movement and have values showing from and to squares in the chess board. The Network Object created is then trained using the encoded data that are actually generated by the human Grandmaster (GM) during Championship games. The training process requires a set of examples of proper network behavior - network inputs and the target outputs. During training the weights and biases of the network are iteratively adjusted to minimize the error of the network. The simulation of the designed system allows checking of performance for the new input moves. The trained network object is simulated in the new unknown environment where with the aid of patterns provided to it. This is to test whether it has learnt to produce an output that is intelligent enough.

6. Results and Discussions

6.1 Test Case I: Scenario (Initial phase of the game)

The trained system when tested in the unknown environment found to behave intelligently and was capable enough to produce the result which drives the chess pieces within the board on account of the learned patterns. The system when tested generated the exact moves as the way grandmasters opens the game during the initial stage of the game. The standard patterns followed by the human grandmasters were clearly imitated by the trained neural network in the early phase of the game.

1) Table 2. Test Data (Scenario 1)

Sl.	Input	Output	Remarks
1	5254	5755	Optimal move
2	2133	7866	Optimal move
3	7163	2836	Optimal move

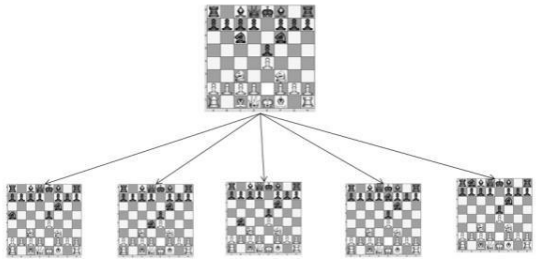
Fig 6: Observation Table 2

6.2 Test Case II: Scenario (Intermediate phase of the game)

After the game progresses, the pieces in the chess board develop resulting on exponential growth in possibility for each of the piece to move within the board. During such scenario the result of the system is found to be very simple and effective as it realizes and hinted the intermediate move of a particular piece on the board to be moved next. When an unknown input move is supplied to the system it is found to be capable of deciding the position from where the piece is to be moved. Since the piece to be moved next was clearly hinted, the tree search technique can help to find out the suitable destination depending upon the situation of the game on the board. The Artificial neural network and the tree based searching technique together in hybrid form is hence found to be effective that reduce the cost to produce move in terms of move searching time. Instead of deriving all the possibility and then searching the entire tree for best possible move, a tree can formed from the position that is suggested by the designed system.

2) Table 3. Testing result (scenario 2)

Sl.	Input	Output	Remarks
1	6143	3639	Move knight from 36
2	6355	4852	Move Queen from 48



3) Fig 7: Searching move using game tree

6.3 Testing of the system in trained environment:

The designed system when tested with the sequence of trained move produced the result as shown below in the Table 4. It is found that system behaved very well for the trained sequence of moves.

4) Table 4. Testing result (trained environment)

Sl.	Input	Output	Remarks
1	4244	7866	Good move
2	3234	7776	Good move
3	6263	3735	Good move
4	4445	4746	Good move
5	5254	6877	Good move
6	7152	5870,8839	Castling, positions hinted
7	5233	6685	Good move
8	3175	7766	Good move
9	7566	5766	Good move
10	4142	6665	Good move
11	5465	3865	Good move
12	7274	6858	Good move
13	5141	6521	Good move

14	1121	4866	Good move
15	7485	6663	Good move
16	4132	6381	Queen thrown

Fig 8: Observation Table 4

6.4 Testing of the system in unknown environment

The designed system when tested in the new environment produced the results that are very much acceptable during initial phase of the game by hinting some of the correct position of the piece which is to be moved next (eg. 5755). When the bishop was moved from position 61 to 34, the system hinted to move pawn from position 67, but is found to be blocked by knight positioned on 66. Even though it hinted the invalid move but the system was intelligent enough to suggest the piece to be move next in the unknown environment. The system performed even more intelligently as it hinted to move the knight from position 36 to 44 as because 44 is found to be a good position supported by one of the pawn residing in the position 55. But, when a move 6344 is tested it hinted to move a bishop from position 38, which is found to be not a good choice. Instead, it could have captured that knight by its pawn residing in the position 55.

The designed system when tested in the untaught environment produced the following results:

5) Table 5. Testing result (unknown environment 1)

Sl.	Input	Output	Observation
1	5254	5755	Good move
2	2133	7866	Good move
3	7163	2836	Good move
4	6134	6705	Move pawn(knight blocks the move)
5	4243	3644	Move knight (very strong move)
6	4142	2755	Move pawn (destination not defined)
7	6344	3845	Move bishop (weak move)

Fig 9: Observation Table 5

6)

The result of the game shown above in the Table 5 was readjusted using the concept of game tree technique when required. The network is then further trained with the input and the correct output so that the system adapts such scenario. Again, after training the system with the moves where system found difficult to produce good result, the system produced results as shown in the Table 6.

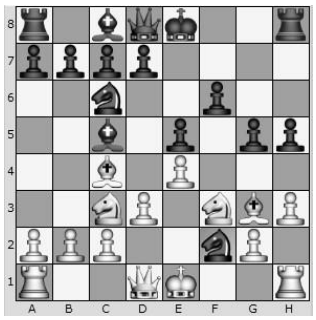


Fig 10: Observation Table 6

7) Table 6. Test data (unknown environment 2)

Sl.	Input	Output	Observation	Remarks
1	5254	5755	Move pawn from 57	Optimal move
2	6134	7866	Move knight from 78	Optimal move
3	4243	2836	Move knight from 28	Optimal move

4	7163	6829	Move bishop from 68	Good move
5	2133	6671	Move knight from 66	Good move
6	3175	6705	Move pawn from 67	Good move
7	7584	7759	Move pawn from 77	Good move
8	8283	8765	Move pawn from 87	Good move
9	8473	7460	Move knight from 74	Optimal move

7. CONCLUSION

The ANN's can be trained to learn moves as complex as those of chess games have been long debated upon. Apart from Octavian there has been no mention of ANN applications in chess playing. Even Octavius reports degradation in performance. Our work brings out the fact that ANN's can indeed be used for playing, teaching or helping humans in playing chess. A true minimax search of a game tree may be expensive since every leaf node must be visited. Since, for a uniform tree with exactly W moves at each node, there are

W^D nodes at the layer of the tree that is D ply from the root. Because of the depth of the game tree for chess was potentially too deep to be successful when using a pure approach, it was found if Artificial Neural Network and tree searching technique used in hybrid form can substantially reduce the move searching cost in terms of time and space. If sufficient number of training patterns is presented to ANN, it was found that the machine cleverly hints the suitable position from where the move can be made. This position can be used to derive a tree which tends to produce effective moves.

In the simplest case the best algorithm is the one that visits fewest nodes when determining the true value of a tree. It was found that the scope of chess as a problem domain was too broad for minimax approach because of the fact that every leaf node is to be reached for best move. Therefore, the problem was narrowed by the way ANN performed. A lot of time can be saved by the use of this smarter approach as because there is no requirement to visit all leaves. This technique can be used to device a chess engine that can play suggest moves, and to improvise its play, game after game, improving the understanding gained by the human players during the play.

REFERENCES

- https://www.chess.com/terms/alphazero-chess-engine
- https://www.shredderchess.com/online/endgame-database.html
- https://www.chessprogramming.org/Material https://en.wikipedia.org/w/index.php?title=Monte_Carlo_tree_search&oldid=1026010523
- https://www.chessprogramming.org/Nalimov_Tablebases http://neuralnetworksanddeeplearning.com
- Chess.com website: https://www.chess.com/article/view/computerchess-engines