

Machine Learning Comparative analysis of min-max algorithm with alpha-beta pruning optimization for chess engine.

Amit Yadav,

Department of computer science and engineering,
Chandigarh University, India

Sahil Verma,

Uttaranchal University, Dehradun, India Email: sahilverma@ieee.org

Mohammad Shuaib Khan,

Department of computer science and engineering,
Chandigarh University, India Email: shuaibkhan.it@gmail.com

Bhupendra Singh Rawat

Uttaranchal University, Dehradun, India

Abstract—The chess game engine is introduced which makes use of a Minmax algorithm for searching game trees and makes use of alpha-beta pruning algorithm to reduce the search space of game states in the game state tree. The first part of this paper is an expository presentation that demonstrates and explains the Minmax algorithm. The paper also proves that the alpha-beta pruning method is shown to make the chess engine algorithm optimal, and bounds are obtained for its running time with various randomized data.

I.

INTRODUCTION

Computer programs for playing the chess game usually search a large number of states from the game state tree where each node is a state that represents the state in the game. An algorithm called the “Minmax algorithm” is used to select the best possible state, which may or may not be correct (guarantees a checkmate) but is the best optimal state from the current state to all possible future states. Additionally, the Alpha-beta pruning algorithm is used to reduce the number of iterations in the Minmax algorithm.

The purpose of this paper is to create and analyze the performance of a chess game engine that makes use of this algorithm. Additionally, we also analyze the performance of the game engine that reduces the search space for the Minmax algorithm using the Alpha-beta pruning algorithm.

Section I of this document describes the literature review for the Minmax algorithm and the Alpha-beta pruning algorithm. Section II describes the proposed methodology to create

the AI chess game engine. Section III contains the results and a discussion on recording the results. Section IV gives the comparative analysis of the game engine with and without the Alpha-beta pruning function. Finally, Section V gives the conclusion derived from the results.

II. LITERATURE REVIEW

A. Minmax Algorithm

The Minmax algorithm [1], [2] is a simple recursive algorithm that determines the next state by analyzing the different states in the game tree.

The maximizing player tries to maximize the score of the next state whereas the minimizing player tries to minimize the score of the next state. However, since the Minmax algorithm cannot search the entire game tree to find the best next state, it uses a fixed depth search to determine the next state [21].

The Minmax algorithm works by generating a game tree up to a certain depth with the current state as the root node. Then we generate

a score value for each leaf node using the evaluation function. Depending on the type of parent node (maximizing or minimizing), we select the leaf node's value as the parent node's value [22]. So, if the parent node is a maximizing player, it will always select a state with the highest score value from its child nodes.

We repeat this process until we reach our root node at depth 0, and we can determine the optimal path that maximizes (or minimizes) our state value.

Fig. 1. Considering the example in Fig. 1, where each node is the state is the game of chess. A white state represents a maximizing player and a black state represents a minimizing player [23], [30-33]. The leaf nodes are D, E, F, and G which have been assigned score values according to the evaluation function. The current state is the root node A.

Now, considering state B and its child nodes D and E, node B is a minimizing node and hence it will be assigned the minimum value from its child nodes. Hence, node B will be assigned the value 3. Similarly, for the nodes, C, F, and G, node C will be assigned the value 2 [24]. Node A is a maximizing node and hence it will select the maximum score value from its child nodes. Hence, node A will be assigned the value 3.

Looking at the above example we can see that the next best state from the current state A is node B. We can assume that the opposing player plays optimally and selects node E [25], [26].

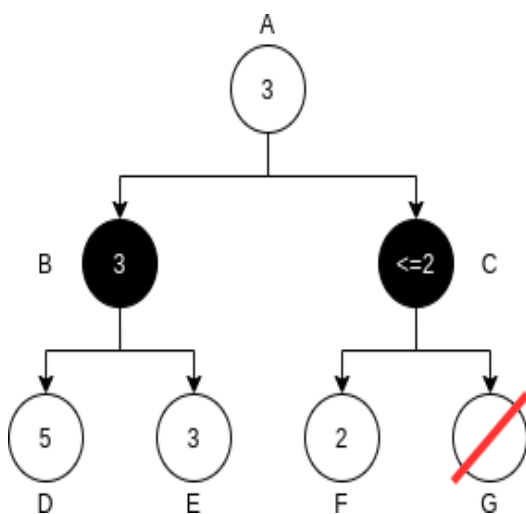


Fig. 1. Ex. Game tree

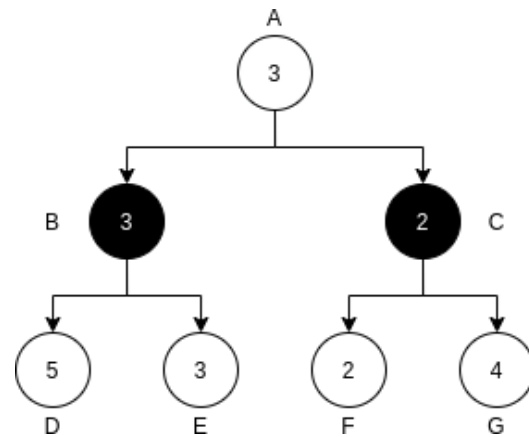


Fig. 2. Ex. Game tree with $-\beta$ pruning

B. Alpha-beta pruning algorithm

Considering the example in Fig. 2, we can see that node F has a score value of 2. We know that the parent node C is a minimizing node and hence its score value will be at most equal to 2 (from node F). Now considering the left subtree (node B, D, and E), the value of node B will be equal to

3. We can hence determine that node A being a maximizing node will select node B (value 3) over C (at most 2). Thus, we would never have to calculate the score value of node G, since its parent node will always select the minimum value, which can at most be 2.

This method of reducing the search space in the game tree is known as $-\beta$ pruning [1], [2], [3], where α and β are the upper and lower bounds. With $-\beta$ pruning, we only have to explore the most optimal nodes which can lead to a win (considering the opposing player plays optimally) and we can stop exploring nodes that do not change the score of their parent nodes and β are the best scores that either player can achieve where corresponds to the maximizing player and β corresponds to the minimizing player (Fig 3). Initially, α is assigned the minimum possible score for the maximizing player which is $-\infty$ and β is assigned the maximum possible score for the minimizing player which is ∞ [27].

Now, consider the following example using the α and β variables, we can determine which node should be explored further and which can be

pruned. The condition to terminate the recursive call is when the value of α is less than or equal to β ($\alpha \leq \beta$). For minimizing node B, α would remain the same ($-\infty$) and β would be $\min(\text{node D}, \text{node E})$ which is 3.

Since the left subtree is now explored, the values of α and β for node A will now be ($\alpha = 3$ and $\beta = 3$). The algorithm will now start evaluating the right subtree with modified α and β . Node C is a minimizing node and hence, for node F value of β for node C will be 2 (due to F). As the termination condition is now satisfied the recursive call will be terminated and the current score value will be returned to the parent node [28], [29].

Following is the pseudocode for Minmax with Alpha-beta pruning,

Alpha-beta pruning depends on the order in which the child nodes are visited. If children of a node are visited in the worst possible order, pruning may not happen. For maximizing nodes, we want to visit the child with the highest score value first [30,31]. Similarly, for the minimizing nodes, we want to visit the nodes, we want to visit the child with the lowest score value first.

III. PROPOSED METHODOLOGY

In this paper, we will be comparing the effectiveness of the Alpha-beta pruning technique in improving the performance of the Minmax algorithm in chess. Additionally, we will analyze the performance results to obtain a quantitative measure for the effectiveness of the Alpha-beta pruning technique.

A. Board Representation

The chessboard is represented in the simplest possible manner - as an 8 by 8 matrix, each containing a piece or a blank space. For board representation, we use the chessboard.js library. This library provides the chessboard layout, its animations, and different API calls to interact with the chessboard.

B. Evaluating the next move

The chess engine uses Flask to interact with the chessboard. We use the python-chess library to determine all the legal next moves from the current state. Using the library, we first create a

chessboard object with starting state representing the start of the game. In order to make a move, the chess object can be passed an alphanumeric string which tells the board to move a piece from position A to position B. If the move is illegal, the method call returns an error. The board object also has a method `legal_moves` which returns an iterable list of all possible next moves. These moves are used by the Minmax algorithm to determine the next best state. The board object also implements a method `fen`, which returns a FEN string that represents the state of the board and this string can be used by the chessboard.js to update the state to the user.

C. Heuristic evaluation function

The Minmax score value for each state will be the sum of values of all the pieces present on the chessboard. Hence, the following is the equation to calculate the score value,

$$\text{Score} = (\text{BLACK}) + (\text{WHITE})$$

Before calling the Minmax function for making a move the α and β variables will be initialized to $-\infty$ and ∞ respectively. Once the score is calculated for the child node, it will be assigned to the parent node's α or β depending on whether it is a maximizing node or a minimizing node. The α and β variables are maintained by each state and when the terminating condition is true ($\alpha \leq \beta$), the recursive call in the Minmax algorithm is terminated.

D. Game over condition

The method `is_game_over` provided by the python-chess library is called before every move to determine if the game is over if any of the following conditions happen:

Checkmate.

Stalemate (Not in check state but no legal moves are available).

Insufficient material.

Other conditions for the game to end are not considered (resignation, timeout, repetition, agreement).

IV. RESULTS

To obtain performance analysis for the Minmax algorithm, we considered the time required to make a single move in a game of chess in milliseconds. The Minmax algorithm used by the chess engine evaluates the game tree with a

depth of 3. The chess engine is played against the Stockfish 14 game engine. The Minmax algorithm plays 10 games with the Stockfish 14 game engine and additionally, 10 games are played with the alpha-beta algorithm.

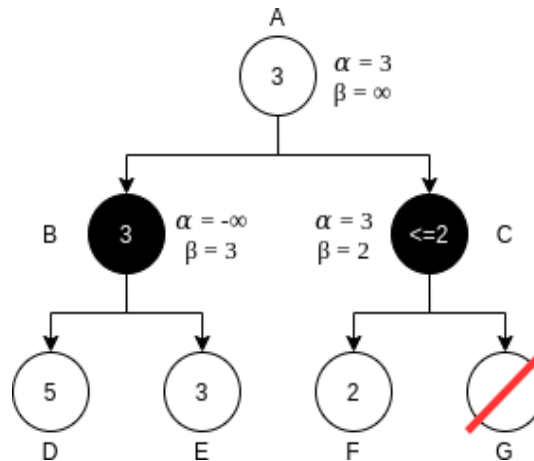


Fig. 3. Ex. $-\beta$ pruning

TABLE I

HEURISTIC EVALUATION TABLE

BLACK	SCORE	WHITE	SCORE
Pawn	-10	Pawn	10
Bishop	-30	Bishop	30
Knight	-30	Knight	30
Rook	-50	Rook	50
Queen	-90	Queen	90
King	-900	King	900

A.

Minmax algorithm

Following is the bar graph for 1 game played against the Stockfish game engine using the Minmax algorithm, as shown in Fig 4.

Following is the line graph for a set of 10 games played against the Stockfish game engine using the Minmax algorithm as shown in Fig 5.

Following is the bar graph for the number of moves played each game for a set of 10 games against the Stockfish game engine using the Minmax algorithms shown in Fig 6.

B. Minmax algorithm with alpha-beta pruning

Following is the bar graph for 1 game played against the Stockfish game engine using the Minmax algorithm with alpha-beta pruning as shown in Fig 7.

Following is the line graph for a set of 10 games played against the Stockfish game engine using the Minmax algorithm with alpha-beta pruning as shown in Fig 8.

Following is the bar graph for the number of moves played each game for a set of 10 games against the Stockfish game engine using the Minmax algorithm with alpha-beta pruning as shown in Fig 9.

C. Analysis

Hence, looking at the above results, we can conclude that the Minmax algorithm takes more time to calculate each move since it looks at all the states in the game state tree for a certain depth. This results in higher time required for each move as compared to the alpha-beta pruning optimization which does not have to look at all the states.

V. CONCLUSION

In this paper, we implemented the Minmax algorithm to create a chess engine and also

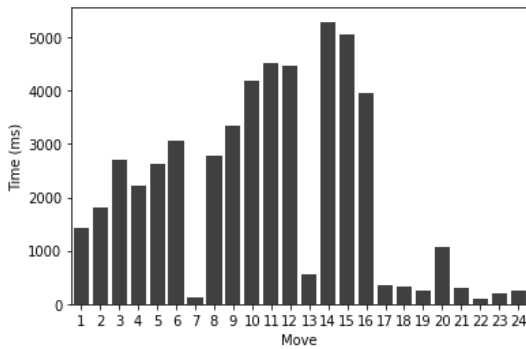


Fig. 4. Minmax bar graph for 1 game

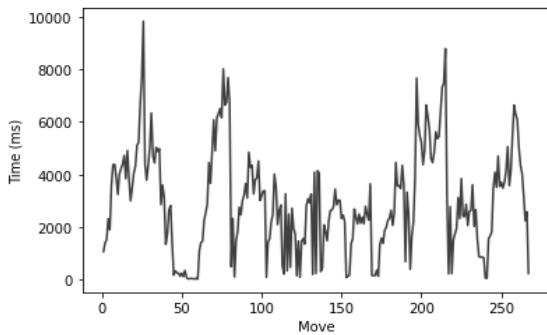
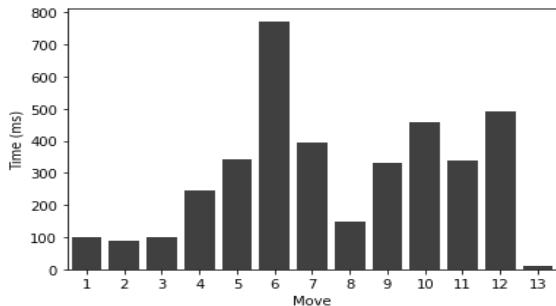


Fig. 5. Minmax line graph plot for 10 games



compared its performance with the Alpha-beta optimization. The Minmax algorithm generated the game tree with a depth of 3. The chess engine was tested against the Stockfish 14 chess engine. The results show that the Alpha-beta optimization significantly reduces the time required to determine the next move.

Thus, the alpha-beta pruning algorithm optimizes the Minmax algorithm and is 90.7455% faster than the Minmax algorithms

Minmax algorithm			
No.	Property	Result	Unit
1	Average time taken per move for 10 games	2965.0524	ms
2	Average no. of moves for each game	27.7	-

TABLE III

II. MINMAX ALGORITHM WITH ALPHA-BETA

Minmax algorithm with alpha-beta			
No.	Property	Result	Unit
1	Average time taken per move for 10 games	274.4000	ms
2	Average no. of moves for each game	16.0	-

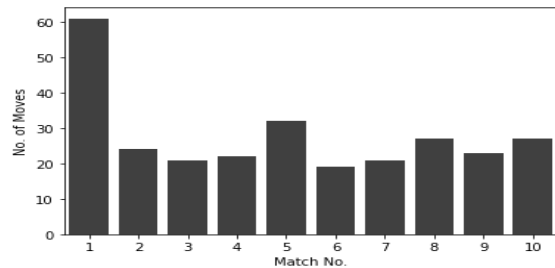


Fig. 6. 3.Minmax bar graph plot for moves each game

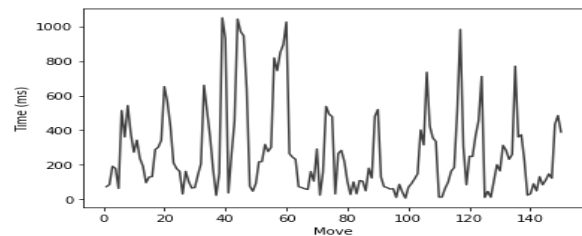


Fig. 7. 4.Minmax with alpha-beta bar graph plot for 1 game

TABLE II

I. MINMAX ALGORITHM

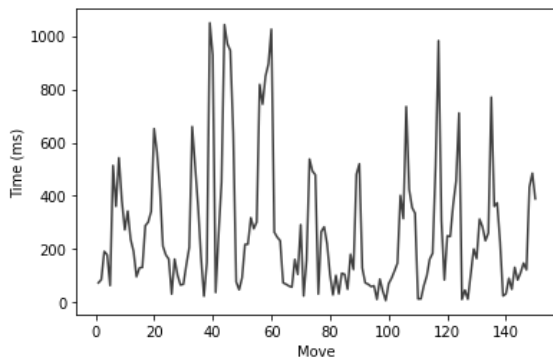


Fig. 8. 5.Minmax line graph plot for 10 games

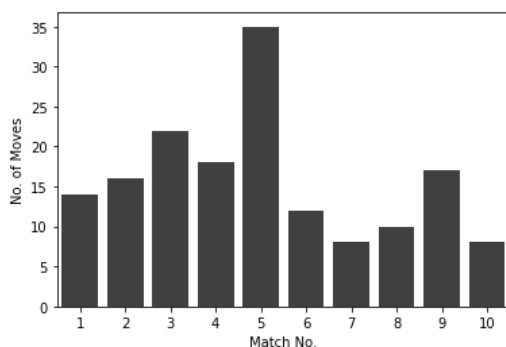


Fig. 9. 6.Minmax with alpha-beta wbar graph plot for moves each game

REFERENCES

[1] J. Liu et al., "A Comprehensive Privacy-Preserving Federated Learning Scheme with Secure Authentication and Aggregation for Internet of Medical Things," in *IEEE Journal of Biomedical and Health Informatics*, doi: 10.1109/JBHI.2023.3304361.

[2] Ramisetty, Sowjanya; Anand, Divya; Kavita; Verma, Sahil; Jhanjhi, N. Z.; Masud, Mehedi; Baz, Mohammed, *Computer Systems Science and Engineering* 2023, 45(2), 1971-1983.
<https://doi.org/10.32604/csse.2022.021924>

[3] Kaur, Ramanpreet, Divya Anand, Upinder Kaur, Sahil Verma, Kavita, Seok-Wook Park, A. S. M. Sanwar Hosen, and In-Ho Ra. 2023. "An Advanced Job Scheduling Algorithmic Architecture to Reduce Energy Consumption and CO2 Emissions in Multi-Cloud" *Electronics* 12, no. 8: 1810.
<https://doi.org/10.3390/electronics12081810>

[4] Upadhyay, Shrikant, Mohit Kumar, Aditi Upadhyay, Sahil Verma, Kavita, A. S. M. Sanwar Hosen, In-Ho Ra, Maninder Kaur, and Satnam

Singh. 2023. "Digital Image Identification and Verification Using Maximum and Preliminary Score Approach with Watermarking for Security and Validation Enhancement" *Electronics* 12, no. 7: 1609.
<https://doi.org/10.3390/electronics12071609>

[5] Xi-Zhao Wang, Yu-Lin He, P. Su, and Wen-Liang Li, "Two-ply iterative deepening in Chinese-chess computer game," 2009 International Conference on Machine Learning and Cybernetics, pp. 2020–2026, 2009.

[6] J. J. Gillogly, "The Technology Chess Program," *Artificial Intelligence*, vol. 3, pp. 145–163, 1972.

[7] Bahuguna, A., Ashraf, A., Kavita, Verma, S., Negi, P. (2023). Brain Tumor Classification from MRI Scans. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) *International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems*, vol 537. Springer, Singapore. https://doi.org/10.1007/978-981-99-3010-4_57

[8] Gupta, H., Kaur, A., Kavita, Verma, S., Rawat, P. (2023). Recognition of Handwritten Digits Using Convolutional Neural Network in Python and Comparison of Performance for Various Hidden Layers. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) *International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems*, vol 537. Springer, Singapore.
https://doi.org/10.1007/978-981-99-3010-4_58

[9] Dhetarwal, M., Ashraf, A., Verma, S., Kavita, Rawat, B. (2023). Employee Turnover Prediction Using Machine Learning. In: Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (eds) *International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems*, vol 537. Springer, Singapore.
https://doi.org/10.1007/978-981-99-3010-4_55

[10] Thind, R., Divya, K., Verma, S., Kavita, Kaur, N., Uniyal, V. (2023). Voice Email for the Visually Disabled. In: Hassanien, A.E., Castillo, O.,

- Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. ICICC 2023. Lecture Notes in Networks and Systems, vol 537. Springer, Singapore. https://doi.org/10.1007/978-981-99-3010-4_60
- [11] Jain, Shikha, Navneet Kaur, Sahil Verma, Kavita, A. S. M. Sanwar Hosen, and Satbir S Sehgal. 2022. "Use of Machine Learning in Air Pollution Research: A Bibliographic Perspective" *Electronics* 11, no. 21: 3621. <https://doi.org/10.3390/electronics11213621>
- [12] Ayanouz, S. A. Abdelhakim, B. Benhmed, and Mohammed., Eds., 2020.
- [13] T. & Lalwani, Bhalotia, & Shashank, A. & Pal, Bisen, S. & Rathod, and Vasundhara, 2018.
- [14] M. & Nadim, Inupakutika, D. & Kaghyan, & Sahak, Akopian, & David, and Gunnam.
- [15] Das, S. R., Jhanjhi, N. Z., Asirvatham, D., Ashfaq, F., & Abdulhussain, Z. N. (2023, February). Proposing a Model to Enhance the IoMT-Based EHR Storage System Security. In *International Conference on Mathematical Modeling and Computational Science* (pp. 503-512). Singapore: Springer Nature Singapore.
- [16] J. J. Gillogly, "The Technology Chess Program," *Artificial Intelligence*, vol. 3, pp. 145-163, 1972.
- [17] C. E. Shannon, "Programming a computer for playing chess," *Philosophical Magazine*, vol. 41, no. 7, pp. 256-275, 1950.
- [18] Ikram, H., Fiza, I., Ashraf, H., Ray, S. K., & Ashfaq, F. (2023, February). Efficient Cluster-Based Routing Protocol in VANET. In *International Conference on Mathematical Modeling and Computational Science* (pp. 165-182). Singapore: Springer Nature Singapore..
- [19] K. Srinivasan, L. Garg, D. Datta, A. A. Alaboudi, N. Z. Jhanjhi, R. Agarwal, and A. G. Thomas, "Performance comparison of deep cnn models for detecting driver's distraction," *Materials & Continua*, vol. 68, no. 3, pp. 4109-4124, 2021.
- [20] A. Almusaylim, Z. Jhanjhi, N. Z. Alhumam, and A, "Detection and mitigation of RPL rank and version number attacks in the internet of things: SRPL-RP," *Sensors*, vol. 20, no. 21, pp. 5997-5997, 2020.
- [21] V. Dogra and A. Singh, "Analyzing DistilBERT for Sentiment Classification of Banking Financial News," in *Intelligent Computing and Innovation on Data Science*, vol. 248. Springer.
- [22] M. Kaur and A. Singh, "FANET: Efficient Routing in Flying Ad Hoc Networks (FANETs) Using Firefly Algorithm," *Lecture Notes in Networks and Systems*, vol. 248.
- K. Kumar, "A Survey of The Design and Security Mechanisms of The Wireless Networks and Mobile Ad-Hoc Networks," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 993, pp. 12 063-12 063, 2020.
- [23] A. Srivastava and S. Verma, "Analysis of Quality of Service in VANET," *Materials Science and Engineering Conference Series*, vol. 993, no. 1, pp. 12 061-12 061, 2020.
- [24] I. A. Shah, Q. Sial, N. Z. Jhanjhi, and L. Gaur, "The Role of the IoT and Digital Twin in the Healthcare Digitalization Process: IoT and Digital Twin in the Healthcare Digitalization Process," *Digital Twins and Healthcare: Trends, Techniques, and Challenges*, pp. 20-34, 2023.
- [25] N. Z. Jhanjhi, S. N. Brohi, N. A. Malik, and M. Humayun, "Proposing a hybrid rpl protocol for rank and wormhole attack mitigation using machine learning," 2020 2nd International Conference on Computer and Information Sciences (ICCIS), pp. 1-6, 2020.
- [26] K. Hussain, S. J. Hussain, N. Jhanjhi, and M. Humayun, "SYN Flood Attack Detection based on Bayes Estimator (SFADBE) For MANET," in 2019 International Conference on Computer and Information Sciences (ICCIS).
- [27] I. A. Shah, Q. Sial, N. Z. Jhanjhi, and L. Gaur, "Use Cases for Digital Twin," *Digital*

Twins and Healthcare: Trends, Techniques, and Challenges, pp. 102–118, 2023.

[28] S. Ghosh and A. Singh, "Svm and knn based cnn architectures for plant classification," *Computers, Materials & Continua*, vol. 71, no. 3, pp. 4257–4274, 2022.

[29] Ibrahim Elhenawy, Salwa H. Mahmoud, Ahmed Moustafa, A Lightweight Privacy Preserving Keyword Search Over Encrypted Data in Cloud Computing, *Journal of Cybersecurity and Information Management*, Vol. 3, No. 2, (2020) : 29-41 (Doi : <https://doi.org/10.54216/JCIM.030201>)

[30] Arvind Mahindru, A.L. Sangal, PARUDroid: Validation of Android Malware Detection Dataset, *Journal of Cybersecurity and Information Management*, Vol. 3, No. 2, (2020) : 42-52 (Doi : <https://doi.org/10.54216/JCIM.030202>)

[31] Mohammed I. Alghamdi, An investigation into the effect of cybersecurity on attack prevention strategies, *Journal of Cybersecurity and Information Management*, Vol. 3, No. 2, (2020) : 53-60 (Doi : <https://doi.org/10.54216/JCIM.030203>)

[32] Rishabh Singh, Rishabh Reddy, Vidhi Kapoor, Prathamesh Churi, K-means Clustering Analysis of Crimes on Indian Women, *Journal of Cybersecurity and Information Management*, Vol. 4, No. 1, (2020) : 05-25 (Doi : <https://doi.org/10.54216/JCIM.040101>)

[33] Miss. Sayali Karmode Yelpale, Security and Privacy Challenges in Cloud Computing: A Review, *Journal of Cybersecurity and Information Management*, Vol. 4, No. 1, (2020) : 36 - 45 (Doi : <https://doi.org/10.54216/JCIM.040102>)