A Neuro-Evolutionary Algorithm Approach to Play Connect4

Haya Fatima
DSSE
Habib University
sf07503@st.habib.edu.pk

Zainab Haider
DSSE
Habib University
zh07104@st.habib.edu.pk

Dua Batool
DSSE
Habib University
db07098@st.habib.edu.pk

***Abstract*—This study explores an evolutionary approach to evolving neural network weights for training game agents, with a focus on Connect4 gameplay. We encode a multilevel feedforward neural network architecture and utilize a genetic algorithm to optimize the weights, aiming to enhance decision-making processes and overall performance of the game agent. The research investigates the employability of neuroevolutionary techniques in game agent training and evaluates the efficacy of the proposed framework in comparison to traditional methods. Initial experiments, including interactions with random and heuristic-based opponents, revealed challenges in achieving satisfactory performance. However, through iterative refinement and the simplification of the decision-making process, our refined approach yielded promising results, demonstrating substantial improvements in gameplay performance. By leveraging evolutionary algorithms to directly train the neural network, our game agent achieved notable success against both random and heuristic-based opponents. These findings underscore the effectiveness of our evolutionary approach in refining neural network weights and highlight its potential applicability in training game agents for complex tasks.**

***Index Terms*—evolutionary algorithm, game agent, genetic algorithm, minimax algorithm, neural network**

I. Introduction

A. Game Agents in AI

Artificial Intelligence (AI) has made significant progress in recent years, evolving from theoretical concepts to practical applications across diverse domains. Among these applications, game agents stand out as a direct reflection of AI's ability to learn and adapt. Game agents, equipped to play and excel at various games, demonstrate AI's capacity to understand, strategise, and succeed in complex, dynamic environments that mirror real world scenarios.

The relevance of game agents extends far beyond mere entertainment; their impact extends to numerous real-world applications, from robotics to cybersecurity. In robotics, game-playing AI systems are instrumental in developing autonomous agents capable of navigating dynamic and uncertain environments with efficiency and precision. By simulating real-world scenarios through games, AI researchers can test and refine algorithms that underpin critical decision-making processes in robotics applications.

Similarly, in cybersecurity, game agents play a crucial role in threat detection, intrusion detection, and vulnerability assessment. By modeling adversarial behaviors and strategies within virtual environments, AI-powered game agents can effectively simulate cyberattacks and devise robust defense mechanisms. This proactive approach enables cybersecurity professionals to stay ahead of evolving threats and safeguard digital assets effectively.

B. Connect4 Game

Among other games that have been the target of game agent training, Connect4 emerges as a promising testbed for studying and advancing game-playing AI. Its simple yet strategic gameplay, characterized by a moderate branching factor and clear objectives, offers a suitable ground for exploring different training techniques. By learning or even mastering Connect4, game agents can demonstrate their ability to analyze complex decision trees, anticipate opponents' moves, and devise winning strategies— an achievement that parallels human-level intelligence in strategic reasoning and decision-making.

C. Neuroevolutionary Framework

In this study, we employ a neuroevolutionary approach to train game agents for Connect4. By using genetic algorithm to evolve neural network weights, we aim to optimize decision-making processes by quickly covering a large area of optimal solutions in the search space. The resulting game agent utilizes the minimax algorithm where score calculation is done through a multi-level feed-forward neural network architecture, rather than a heuristic based approach.

Through this research endeavor, we not only seek to advance the state-of-the-art in game agent training but also contribute to broader artificial intelligence research efforts. By pushing the boundaries of what game agents can achieve in game-playing scenarios, we want to test the employability of this framework and generalize it over multiple problems. Ultimately, our exploration of neuroevolutionary techniques in Connect4 serves as a stepping stone towards developing a generalized framework.

II. Literature Review

A. Introduction

The literature review presented herein aims to provide a comprehensive overview of existing research concerning the application of evolutionary algorithms (EA) in training neural networks for game playing, with a specific focus on mastering

Connect4. By delving into previous studies, this review seeks to elucidate the methodologies, findings, and advancements within this domain, thereby establishing a foundation upon which our own research builds.

Connect4, a classic game of strategic alignment, poses unique challenges for artificial intelligence systems due to its branching factor and complexity. Researchers have explored diverse approaches to address these challenges, ranging from traditional minimax algorithms to more sophisticated techniques involving neural networks and evolutionary algorithms.

Through a systematic examination of relevant literature, this review will identify key trends, methodologies, and advancements in evolving neural networks for Connect Four gameplay. By synthesizing insights from previous studies, we aim to contextualize our own research within the broader landscape of game-playing AI, highlighting its novelty, contributions, and potential avenues for future exploration.

B. Comparision & Analysis

In this literature review, we examine two research papers relevant to the field of artificial intelligence and machine learning. Each paper contributes unique insights and methodologies, shedding light on different aspects of optimization and reinforcement learning.

TABLE I
COMPARISON OF REVIEWED PAPERS

Aspect	**Paper 1: "Learning to play Connect-Four using Evolutionary Neural Networks"**	**Paper 2: "Neural-Fitted Temporal Difference Learning to Learn to Play Connect Four"**
Authors	Robur Box	H. de Haan
Focus	Application of evolutionary algorithms to train neural networks for playing Connect4	Application of neural-fitted temporal difference learning to train Connect Four agents
Methodology	Evolutionary approach for optimizing neural networks using genetic algorithms	Empirical evaluation of neural-fitted temporal difference learning in Connect Four gameplay
Findings	Suitability of evolutionary neural networks for learning Connect4 compared to earlier research	Effectiveness of neural-fitted temporal difference learning in training intelligent game-playing agents
Key Contributions	Exploration of evolutionary neural networks for learning Connect4	Demonstration of NFTD's performance in Connect Four gameplay

Both research papers, despite focusing on distinct domains within artificial intelligence and machine learning, share common themes centered around optimization and learning algorithms. While the first paper by Robur Box [1] explores the application of evolutionary algorithms to train neural networks for playing Connect4, the second paper by H. de Haan [2] investigates reinforcement learning techniques applied to the game of Connect4.

A notable similarity between the papers lies in their utilization of advanced computational techniques to enhance the performance of learning systems. Box [1] advocates for the use of evolutionary algorithms as a means to efficiently search the vast space of possible neural network architectures and parameters for mastering Connect4. This approach resonates with de Haan's [2] application of neural-fitted temporal difference learning (NFTD) in training Connect4 playing agents, wherein neural networks are employed to approximate value functions and guide decision-making processes.

Furthermore, both papers highlight the iterative nature of optimization and learning processes. Box [1] emphasizes the importance of iterative improvement through successive generations of neural network configurations evolved by evolutionary algorithms. Similarly, de Haan [2] describes a batch learning approach in NFTD, wherein agents play multiple games and undergo training iterations to refine their strategies based on observed outcomes.

However, while the papers share common themes, they differ significantly in their methodologies and application domains. Box [1] focuses primarily on the optimization of neural network architectures and parameters for mastering Connect4, while de Haan [2] delves into the specifics of game-playing AI, demonstrating the effectiveness of NFTD in training Connect4 agents.

One strength common to both papers is their emphasis on empirical evaluation and performance validation. Box [1] provides a comprehensive review of empirical studies showcasing the effectiveness of evolutionary algorithms in optimizing neural networks for playing Connect4. Similarly, de Haan [2] presents extensive experimental results demonstrating the performance of NFTD agents in playing Connect4 against different opponents.

However, a potential limitation of both approaches lies in their scalability and generalization to more complex tasks and environments. While evolutionary algorithms offer an efficient means of exploring neural network architectures, their scalability to large-scale networks or non-convex optimization problems may be limited. Likewise, while NFTD shows promise in training game-playing agents, its applicability to other domains with dynamic and uncertain environments requires further investigation.

C. Gaps in Literature Review & Relevance to this study

Despite the valuable insights provided by the existing research, there are several gaps and areas where further investigation is warranted, aligning with the purpose of our research. One notable gap lies in the exploration of effective methodologies for training game agents to play Connect Four. While previous studies have explored various techniques, such as neural-fitted temporal difference learning and minimax tree algorithms, there is still room for innovation and improvement.

Our research aims to address this gap by exploring an evolutionary approach for training game agents to play Connect4. By combining evolutionary algorithms with neural networks and minimax tree algorithms, we seek to develop an efficient and adaptive strategy for teaching game-playing AI. Specifically, our study extends the existing literature by proposing a methodology that leverages evolutionary algorithms to optimize neural network architectures for mastering Connect Four gameplay.

Through our research, we aim to contribute to the advancement of artificial intelligence and machine learning techniques in the domain of game-playing agents. By building upon the foundations laid by existing research and exploring novel approaches, we strive to bridge the gap between theoretical knowledge and practical application, ultimately enhancing the capabilities of game-playing AI systems.

III. Methodology

Our proposed solution integrates three core components: an evolutionary algorithm, a neural network, and a minimax tree. These components interact by exchanging inputs and outputs, forming an interconnected system, as illustrated in Figure 1.

The process commences with the random initialization of weights for the neural network sent to the evolutionary algorithm. These weights are evolved iteratively through the evolutionary algorithm, optimizing the performance of the neural network. The updated weights are then passed to the neural network, which utilizes them to generate an output. This output serves as the score in the minimax tree, facilitating decision-making regarding the optimal move.

Each of these components plays a pivotal role in the overall system, contributing to its effectiveness in generating the game agent. In the following sections, we delve into comprehensive discussions of each component, explaining their functionalities, interactions, and contributions to the system's performance.

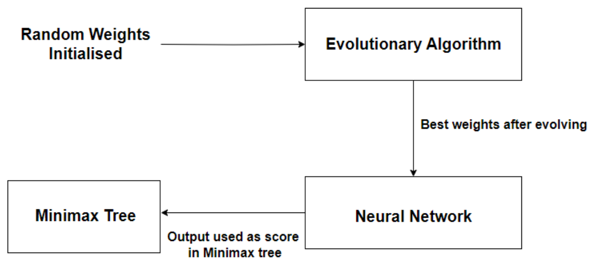

Random Weights Initialised
Evolutionary Algorithm
Best weights after evolving
Neural Network
Output used as score in Minimax tree
Minimax Tree

Fig. 1. Overview of Process

A. Evolutionary Algorithms

The Evolutionary Algorithm (EA) constitutes the foundational component of our system. Its primary function is to evolve the weights of the neural network, enabling it to learn optimal strategies for playing Connect4 and ultimately achieve victory.

The EA begins by initializing a population of individuals, with each individual represented by an array containing the weights of the neural network's input, hidden, and output layers. These weights are initially assigned randomly. Subsequently, each individual neural network within the population engages in a series of Connect4 games against various opponents, repeated a specified number of times (num_games). The outcome of each game—win, draw, or loss—is translated into a numerical score: 1 for a win, 0.5 for a draw, and 0 for a loss. This cumulative score across all games played serves as the fitness metric for the respective neural network.

To iteratively improve the population's overall fitness, the EA employs several evolutionary operators. Initially, parent selection is executed via a binary tournament selection scheme. Then, offspring are generated through crossover and mutation operations. Crossover involves a two-point crossover method applied to the flattened weight arrays, facilitating the exchange of genetic information between parent individuals. Mutation is subsequently applied using an insert mutation approach, introducing stochastic variations in the offspring's genetic makeup. Survivor selection is conducted using a truncation selection scheme, where the fittest individuals are retained to form the next generation. This cyclic process of parent selection, crossover, mutation, and survivor selection iterates over multiple generations to refine the population and improve the overall performance of the neural network.

The EA's optimization process involves the tuning of various hyperparameters, which are enumerated in Table II, allowing for fine-tuning and customization to enhance the evolutionary process.

TABLE II
Hyper-parameters of EA

Parameter	***Value***
POPULATION SIZE	50
GENERATIONS	100
NUM_GAMES	30
MUTATION_RATE	0.5
OFFSPRINGS	10

B. Neural Networks

The Neural Network (NN) serves as a pivotal component in our system, responsible for generating outputs utilized as scores within the minimax tree for game simulation. We explored various configurations of NN architectures to determine the most effective approach.

Initially, we implemented a NN configuration comprising 42 input neurons, each representing a position on the 6x7 Connect4 game board. This architecture featured a hidden layer with 64 neurons and a single output neuron. Subsequently, we experimented with a modified NN configuration, incorporating 138 input neurons. The first 42 neurons remained dedicated to board positions, while the additional neurons accommodated heuristic information about the board derived from the insights provided by de Haan [2].

The heuristic information augmentation aimed to enhance NN performance by integrating strategic insights into the game dynamics. The hidden layers in this configurations was designed to be half the size of the input layer, and the output layer consisted of a single neuron. The activation function employed throughout these NN configurations was the hyperbolic tangent (tanh) function.

Table III presents a concise summary of the various NN variants explored, capturing essential details such as input size, hidden layer configuration, output size, and activation function utilized. These configurations were systematically evaluated to discern their efficacy in facilitating successful Connect4 gameplay.

TABLE III
CONFIGURATIONS OF VARIANTS OF NN

Parameter	*Network 1*	*Network 2*
INPUT LAYER SIZE	42	138
HIDDEN LAYER SIZE	64	69
OUTPUT LAYER SIZE	1	1
ACTIVATION FUNCTION	tanh	tanh

C. Minimax with Alpha-Beta Pruning

The minimax algorithm, augmented with a neural network, serves as a pivotal strategy component in playing Connect Four. It operates through a recursive process that evaluates potential game states, considering both maximizing and minimizing player turns.

At each step, the algorithm checks for base cases, such as reaching the maximum search depth which is set at 5 or encountering a terminal game state, where it leverages the neural network to assign scores accordingly.

Utilizing alpha-beta pruning, the algorithm efficiently prunes branches of the game tree, focusing on promising paths while discarding less favorable ones. Ultimately, it selects the best move by returning the column to drop a piece in, along with its associated score.

This combined approach harnesses the power of neural network evaluation within the context of the minimax algorithm, facilitating strategic decision-making in Connect Four gameplay.

Algorithm 1 gives the pseudocode of the minimax implementation.

IV. EXPERIMENTATION

We conducted a series of experiments to evaluate the performance of our game agent in Connect4 gameplay. The experiments aimed to assess the effectiveness of our neuroevolutionary approach in training the game agent and to identify factors influencing its performance.

The algorithms we implemented are listed below:

```
Algorithms Implemented:
    1. Random Agent
    2. Simple Heuristic Based Agent
    3. Heuristic Based Minimax
    4. NeuroEvolutionary Minimax
    5. Monte Carlo Agent
```

A. Initial Experiments

The experiments were conducted using a Connect4 game environment, where our game agent interacted with various opponents. The game agent was implemented with a neural

Algorithm 1 Minimax Algorithm with Neural Network Evaluation

```
1: Function minimax(node, depth, maximizingPlayer)
2:     valid_locations = GetValidLocations(board)
3:     is_terminal = IsTerminalNode(board)
4: if depth == 0 or is_terminal then
5:     if is_terminal then
6:         if WinningMove(board, PLAYER_2) then
7:             return (None, MAX_SCORE)
8:         else if WinningMove(board, PLAYER_1) then
9:             return (None, MIN_SCORE)
10:        else
11:            return (None, DRAW_SCORE)
12:        end if
13:    else
14:        encoded_state = NN.encode_game_state(board)
15:        return (None, NN.forward_propagation(encoded_state))
16:    end if
17: end if
18: if maximizingPlayer then
19:     column = None
20:     for each col in valid_locations do
21:         row = GetNextOpenRow(board, col)
22:         b_copy = CopyBoard(board)
23:         DropPiece(b_copy, row, col, PLAYER_2)
24:         new_score = minimax_with_NN(b_copy, depth-1, alpha, beta, False, NN)[1]
25:         if new_score $>$ alpha then
26:             alpha = new_score
27:             column = col
28:         end if
29:         if alpha $\geq$ beta then
30:             break
31:         end if
32:     end for
33:     return column, alpha
34: else
35:     column = None
36:     for each col in valid_locations do
37:         row = GetNextOpenRow(board, col)
38:         b_copy = CopyBoard(board)
39:         DropPiece(b_copy, row, col, PLAYER_1)
40:         new_score = minimax_with_NN(b_copy, depth-1, alpha, beta, True, NN)[1]
41:         if new_score $<$ beta then
42:             beta = new_score
43:             column = col
44:         end if
45:         if beta $\leq$ alpha then
46:             break
47:         end if
48:     end for
49:     return column, beta
50: end if
```

network architecture, initially comprising 42 input neurons representing the game board's state (6x7 grid).

In the initial experiments, our game agent played against a random agent, serving as a baseline opponent. Despite the promising neural network architecture, the results were disappointing, with our game agent winning only 8 out of 30 games against the random opponent.

B. Integration of Heuristic Information

To enhance the game agent's decision-making capabilities, we encoded additional heuristic information from the game board into the neural network inputs. This included features such as the number of nearly completed rows, columns, or diagonals for both our agent and the opponent. The input layer was expanded to 138 neurons, while the hidden layer size was halved.

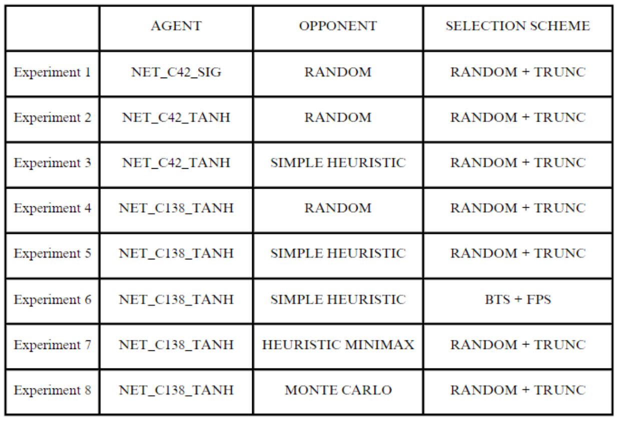
Despite the integration of heuristic information, the game agent's performance against the random opponent did not improve significantly. We observed similar results to the initial experiments, indicating that playing against a random opponent posed challenges due to its unpredictable behavior.

C. Introduction of Simple Heuristic-Based Agent

Recognizing the limitations of playing against a random opponent, we implemented a simple heuristic-based agent as an alternative opponent. This agent prioritized making moves in columns with existing pieces, mimicking a basic strategy. Surprisingly, our game agent's performance deteriorated further, with an average fitness of 3 out of 30 games.

Throughout the experimentation process, we conducted numerous trials against different opponents and modified the neural network inputs to enhance performance. However, despite our efforts, the game agent failed to evolve successfully. Its performance remained below expectations, failing to outperform even a random agent.

Our initial experiments are detailed in the figure 2 below.



	AGENT	OPPONENT	SELECTION SCHEME
Experiment 1	NET_C42_SIG	RANDOM	RANDOM + TRUNC
Experiment 2	NET_C42_TANH	RANDOM	RANDOM + TRUNC
Experiment 3	NET_C42_TANH	SIMPLE HEURISTIC	RANDOM + TRUNC
Experiment 4	NET_C138_TANH	RANDOM	RANDOM + TRUNC
Experiment 5	NET_C138_TANH	SIMPLE HEURISTIC	RANDOM + TRUNC
Experiment 6	NET_C138_TANH	SIMPLE HEURISTIC	BTS + FPS
Experiment 7	NET_C138_TANH	HEURISTIC MINIMAX	RANDOM + TRUNC
Experiment 8	NET_C138_TANH	MONTE CARLO	RANDOM + TRUNC

Fig. 2. Different Experiments Tested

D. Refinement of Approach

Recognizing the limitations of integrating the minimax algorithm into our game agent's decision-making process, we opted to refine our approach. We decided to remove the minimax algorithm altogether and instead focus on training the neural network directly to make moves in Connect4 gameplay.

In this refined approach, we mapped the output of the neural network directly to the columns of the Connect4 game board (i.e., columns 0-6). This enabled the neural network to generate moves autonomously without the need for intermediate scoring or involvement of the minimax algorithm.

With the removal of the minimax algorithm, the evolutionary algorithm was tasked solely with training the weights of the neural network. The training process involved iteratively evolving the neural network weights based on the game outcomes, with a focus on maximizing the number of wins achieved by the game agent.

The revised approach streamlined the game agent's decision-making process, eliminating the complexity associated with integrating the minimax algorithm. By directly training the neural network to make moves, we aimed to simplify the training process and potentially improve the game agent's performance. This yielded promising results. The game agent's performance saw a substantial improvement, with an average fitness of 27 out of 30 games.

The success of the refined approach validated our decision to simplify the game agent's decision-making process and focus on training the neural network exclusively. By leveraging the evolutionary algorithm to optimize the neural network weights based on game outcomes, we achieved notable progress in enhancing the game agent's performance in Connect4 gameplay.

Though we found notable success against the random agent, winning 27 out of 30 games initially, this early dominance made it difficult to observe clear signs of improvement through evolution. To gain a more nuanced understanding of its performance, we extended our testing to include the simple heuristic agent.

The refined approach struggled more against the heuristic agent, achieving only about 9 wins out of 30 initially. Yet, this provided an opportunity to witness the evolutionary process in action. Over successive generations, we observed a significant uptick in performance, culminating in approximately 20 wins out of 30 by the end of the evaluation period. Later when run for 50 games and more generations, it started initially at score 19 and evolved to best score of 41 out of 50 games.

This broader testing validated the effectiveness of our evolutionary algorithm in refining the weights which was not evident in the initial approach.

V. Results

A. Evolutionary Neural Network with Minimax

Figures 3 to 6 showcase the performance of NETC42 and NETC138 against random, simple heuristic. As shown, the fitness started off really low and did not evolve throughout the process. Even against random agents, our agent was not performing well. Even after adding heuristic information about the board state in the input layer, we were not able to get results.

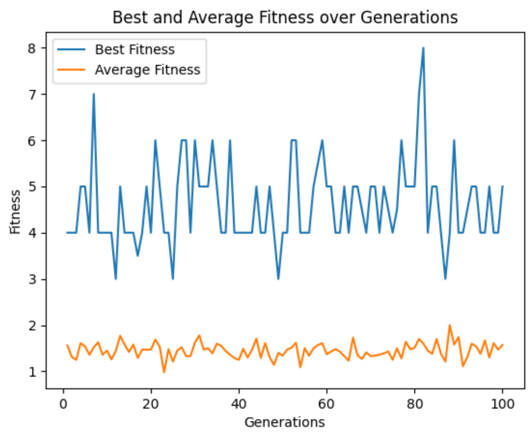


Fig. 3. cs42 against random agent

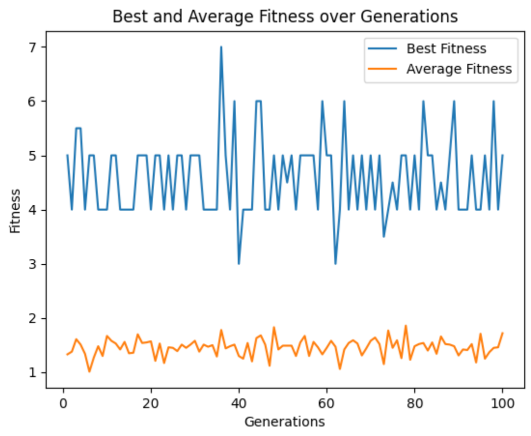


Fig. 4. cs42 against simple heuristic agent

B. Evolutionary Neural Network

Figure 7 shows the results of the NETC42 agent refined approach without minimax. The decision to refine the approach by removing the minimax algorithm and focusing solely on training the neural network directly yielded promising results. The simplification of the decision-making process and the exclusive reliance on evolutionary algorithms for training demonstrate a strategic pivot based on our findings. Similar trend can be seen in figures 8 and 9 respectively where we evolve it against the simple heuristic based agent.

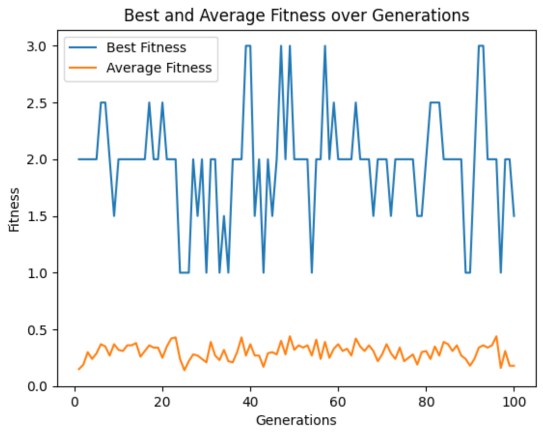


Fig. 5. cs138 against random agent

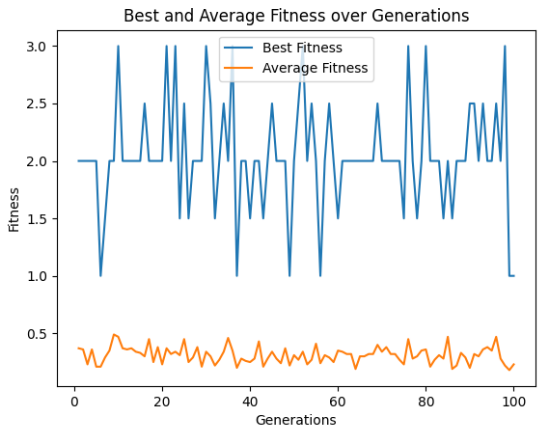


Fig. 6. cs138 against simple heuristic agent

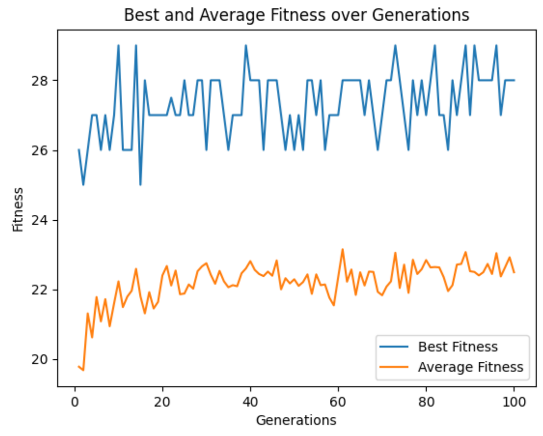


Fig. 7. cs42 against random agent using refined approoch

VI. Conclusion

A. Results

The initial research did not achieve the intended goal of evolving a competent Connect4 player through neuroevolutionary minimax algorithm, highlighting challenges in search space exploration and strategy capture. Despite extensive efforts and experimentation, the evolved agents did not demonstrate sufficient proficiency in playing Connect4 at a compet-

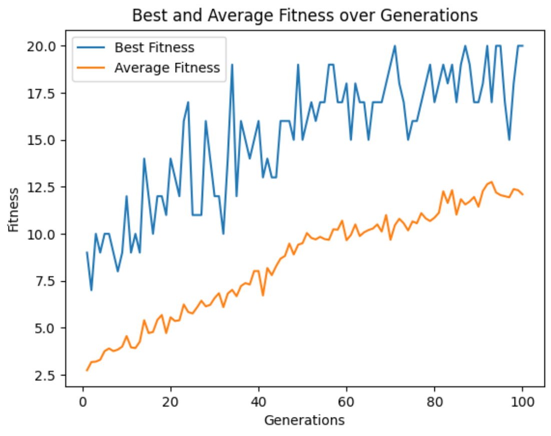


Fig. 8. cs42 against simple heuristic agent using refined approach

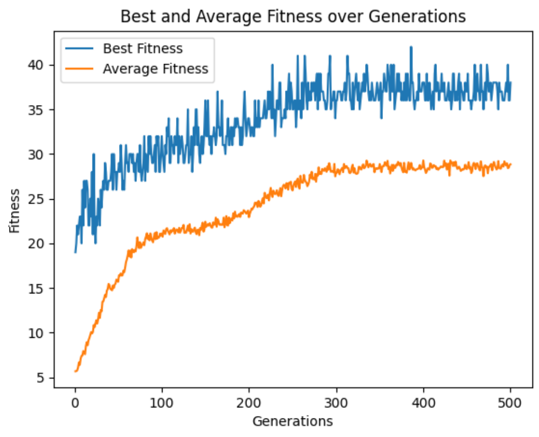


Fig. 9. cs42 against simple heuristic agent using refined approach, 500 generations, 50 games

itive level. This outcome underscores the complexity of the task and the limitations of current evolutionary algorithms in effectively exploring the vast search space of potential gameplay strategies.

The secondary efforts towards neural network training without the minimax algorithm yielded more promising results. By focusing solely on training the neural network to make moves directly in Connect4 gameplay, we observed notable improvements in the game agent's performance. However, due to time constraints, we were unable to conduct comprehensive testing and comparison with other reinforcement learning techniques such as Temporal Difference (TD) learning and Monte Carlo methods.

B. Challenges

The research encountered several formidable challenges throughout the exploration of evolutionary algorithms for training Connect4-playing agents. These challenges underscored the complexity inherent in developing competent game-playing AI and highlighted areas requiring further investigation and innovation.

1) Vast Search Space: Connect4 presents a vast and intricate search space, characterized by the multitude of possible game states and potential moves at each turn. Evolutionary algorithms rely on exhaustive search or sampling techniques to explore this space, but the sheer size of the search space poses a significant challenge. Efficiently navigating this expansive space to discover optimal or near-optimal gameplay strategies is a non-trivial task, requiring sophisticated search algorithms and exploration strategies.

2) Capturing Gameplay Strategy: One of the central challenges faced was capturing the nuanced gameplay strategy within the neural network architectures. Connect4 involves complex decision-making processes influenced by various factors such as board configuration, opponent moves, and strategic long-term planning. Traditional approaches to encoding gameplay strategy within neural networks often struggled to capture these intricacies adequately. Developing neural network architectures capable of effectively learning and representing the diverse gameplay strategies in Connect4 remains an ongoing challenge.

3) Understanding ANN Inner Workings: Artificial neural networks serve as the foundation for learning and decision-making in AI agents. However, understanding the inner workings of ANNs and effectively optimizing them for gameplay proved to be a significant challenge. ANNs comprise numerous interconnected nodes and layers, making it challenging to interpret how they encode and process information related to Connect4 gameplay. Developing insights into the behavior of ANNs and devising strategies for optimizing their performance in game-playing scenarios requires further research and experimentation.

Addressing these challenges will be crucial for advancing the field of game-playing AI and developing more capable and adaptive agents for complex games like Connect4. Future research efforts will need to focus on innovative approaches to search space exploration, neural network architecture design, and optimization techniques to overcome these obstacles and unlock the full potential of evolutionary algorithms in training game-playing agents.

C. Future Work

In future work, we aim to address the identified challenges and enhance the effectiveness of evolutionary algorithms in training game-playing agents for Connect4. One avenue for exploration involves augmenting heuristic information within the evolutionary process to provide additional guidance to the search algorithm, thereby enhancing its ability to discover effective gameplay strategies. Additionally, we plan to progressively upgrade opponent difficulty levels and conduct extensive experiments with varied parameters to refine the training process and improve agent performance. Leveraging computational advancements such as GPU acceleration can significantly speed up the training process, enabling more extensive exploration of the search space and potentially leading to improved results.

Furthermore, we intend to further explore the efficacy of the neural network without the minimax approach and directly compare it with TD learning and Monte Carlo methods. By conducting rigorous experiments with varied parameters and opponent difficulty levels, we aim to provide a comprehensive evaluation of each approach's strengths and weaknesses. Additionally, leveraging computational advancements such as GPU acceleration will expedite the experimentation process, enabling more extensive exploration and evaluation of different training techniques.

By systematically comparing and evaluating various reinforcement learning methods, we can gain valuable insights into their respective capabilities and applicability in training game-playing agents. This comparative analysis will inform future research directions and guide the development of more effective and adaptive AI agents for complex games like Connect4.

Source Code

The source code for the project can be found at: https://github.com/ZainabbHaider/CI-Project.git.

Acknowledgment

We would like to express our sincere gratitude to Dr. Saleha for her invaluable supervision and mentorship throughout the research process. Her guidance, expertise, and support have been instrumental in shaping this study and steering us towards meaningful insights and discoveries. We deeply appreciate her dedication and commitment to our academic and intellectual growth.

References

[1] B. Robur, "Learning to play Connect-Four using Evolutionary Neural Networks," 2015. Available: https://fse.studenttheses.ub.rug.nl/12645/1/AI-BA-2015-RoburBox.pdf

[2] H. de Haan, "Neural-Fitted Temporal Difference Learning to Learn to Play Connect Four," 2013. Available: https://fse.studenttheses.ub.rug.nl/11267/1/AI_BA_2013_HEINDEHAAN.pdf

[3] J. Schaeffer, Y. Bjornsson, N. Burch, A. Kishimoto, M. M ¨ uller, R. Lake, ¨ P. Lu, and S. Sutphen, "Solving checkers," in Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2005, pp. 292- 297