Fuzzy Systems and Soft Computing
ISSN : 1819-4362

AI AGENT FOR A STRATEGY-BASED GAME

[1]S.Santhi Priya, [2] N. Sri Ranga Sai Saranya,[3] N. Naveen Kumar,[4]A. Ravi, [5]P. Venkat

[1]Assoc. Professor, Department of CSE-AI

[2,3,4,5] UG Scholar, Department of CSE-AI
Chalapathi Institute of Technology, Guntur, Andhra Pradesh, India-522016

ABSTRACT

Strategy-based games, ranging from Chess to Real-Time Strategy (RTS) games, serve as a benchmark for artificial intelligence (AI), requiring strategic thinking, adaptability, long-term planning, and decision-making under uncertainty [2]. The challenge lies in enabling AI to navigate vast decision trees, predict future outcomes, model opponents, and adjust strategies dynamically [1]. Recent advancements in machine learning, particularly in Reinforcement Learning (RL), Monte Carlo Tree Search (MCTS), and neural networks, have significantly improved AI performance in these games through experience and self-play [6]. These techniques allow AI to master complex games, adapt to various strategies, and make competitive decisions against human players. This paper aims to develop an AI agent capable of playing strategy-based games, such as Chess and RTS games, by integrating cutting-edge techniques [3]. The agent will leverage RL for strategic depth and adaptability, MCTS for efficient decision-space navigation, and Minimax with Alpha-Beta Pruning for optimal move selection. Neural networks will enhance pattern recognition, while opponent modelling will enable the AI to adjust its strategy based on the opponent's play style [4]. By developing an AI that excels in strategy-based games while adapting to different environments and opponents, this project contributes to AI research, game development, and strategic problem-solving [7]. The findings have potential applications in AI competitions, game design, educational tools, and decision-making in complex environments [8].

Keywords: Artificial Intelligence, Reinforcement Learning, Monte Carlo Tree Search, Neural Networks, and Minimax Algorithm.

1. INTRODUCTION

Artificial Intelligence (AI) has seen remarkable advancements, particularly in its application to games that require strategic thinking and decision-making [9]. Strategy based games, including Chess, Go, and Real-Time Strategy (RTS)

games, present an ideal platform for AI research due to their complexity, need for long-term planning, and dynamic nature [9]. These games require players to assess multiple possibilities, and adapt to evolving situation challenges that also make them a compelling field for AI development [12]. Historically, AI in gaming was primarily rule-based, relying on predefined heuristics to make decisions. While effective in controlled settings, these approaches lacked adaptability and struggled with unexpected strategies [21].

The introduction of machine learning and search-based techniques has led to AI systems that can learn from experience, improve decision [18]. This shift has been driven by Reinforcement Learning (RL), Monte Carlo Tree Search (MCTS), and neural networks, among other methodologies [16]. This paper explores the development of an AI agent capable of learning and optimizing its game play in strategy-based environments [18]. By leveraging modern AI techniques, the agent aims to exhibit intelligent decision-making, strategic planning, and adaptability against varying opponents. Traditional AI methods in gaming often rely on static rule sets, limiting their effectiveness in unpredictable environments [20]. These approaches struggle against dynamic human play styles and fail to generalize across different game situations [22]. The primary challenge is designing an AI that can analyze vast decision spaces, anticipate future moves, and adjust its strategy dynamically while maintaining computational efficiency [14]. To address this, an AI system must integrate learning-based methods with efficient search algorithms. Reinforcement Learning enables adaptive learning from game play experiences, while MCTS and Minimax with Alpha-Beta Pruning improve decision-making efficiency[17]. Additionally, incorporating neural networks can enhance the AI's ability to recognize patterns and refine its strategy.

The core objective is to develop an AI that a compete effectively, adapt to opponents, and improve over time without requiring explicit programming for every possible scenario [15]. Deployment Considerations: The AI will be integrated into a game environment where it must make decisions efficiently with low computational overhead [19]. By addressing these areas, the project aims to develop an intelligent AI agent capable of competing at a high level while demonstrating adaptability and efficient decision-making [22]. The results of this research could contribute to advancements in AI-driven gaming, decision-making frameworks, and applications beyond entertainment, such as autonomous decision systems and AI-driven simulations.

2. LITERATURE SURVEY

2.1 Introduction: Artificial Intelligence (AI) has played a significant role in strategy-based games, enabling computers to compete at human and even superhuman levels. Over the years, various AI techniques have been developed to tackle the challenges of decision-making, opponent modeling, and game state evaluation. This chapter presents a survey of relevant AI methodologies, focusing on Reinforcement Learning (RL), Monte Carlo Tree Search (MCTS), Neural Networks, and Minimax with Alpha-Beta Pruning.

2.2 AI Techniques in Strategy-Based Games: Reinforcement Learning is a machine learning approach in which an agent learns to make decisions by interacting with an environment and maximizing cumulative rewards. It has been successfully applied to games where decision-making is sequential. Deep Q-Networks (DQN): Introduced by Deep Mind, DQN combines Q-learning with deep neural networks to handle large state spaces in games. Policy Gradient Methods: Algorithms such as Proximal Policy Optimization (PPO) and Actor-Critic Methods allow the AI to learn policies for selecting optimal moves. Self-Play Learning: Used in advanced AI models, self-play allows an AI agent to improve by playing against it, refining strategies over time.

2.3. Monte Carlo Tree Search (MCTS): Monte Carlo Tree Search is a widely used algorithm in AI game-playing, especially in games with large state spaces such as Chess and Go. It consists of: Selection: Traversing the game tree based on a selection policy. Expansion: Adding new possible game states. Simulation: Running simulated games to estimate the value of a move. Back propagation: Updating the tree with the results of simulations. MCTS has proven to be effective in games where an exhaustive search is infeasible. It was a key component in AI models like Alpha Go, demonstrating its power in high-level strategic decision-making.

2.4. Neural Networks for Game AI: Neural networks play a crucial role in modern game AI by recognizing patterns, evaluating game positions, and predicting optimal moves. Minimax Algorithm with Alpha-Beta Pruning The Minimax Algorithm is a classical approach for decision-making in two-player games with perfect information. It evaluates possible game states by considering all possible moves and counter-moves. However, due to computational limitations, Alpha-Beta Pruning is employed to reduce the number of states explored, improving efficiency while maintaining optimal decision-making.

3. EXISTING SYSTEM

3.1 Existing AI Approaches in Strategy Games: The AI approaches used in strategy games have

evolved over the years, with several key techniques emerging as central to game-playing agents [17]. These include Minimax, Monte Carlo Tree Search (MCTS), and Reinforcement Learning (RL), each having unique strengths and weaknesses depending on the type of game [12]. However, each of these methods also comes with inherent limitations that affect their scalability, efficiency, and adaptability in dynamic environments [11].

Minimax Algorithm: The Minimax algorithm is a classic decision-making algorithm used primarily in deterministic, two-player, zero-sum games (such as Chess and Tic-Tac-Toe). It works by exploring all possible moves and selecting the one that maximizes the player's minimum payoff, assuming the opponent will also play optimally [9]. Alpha-Beta Pruning is often applied to improve its efficiency by eliminating branches that will not be selected.

Monte Carlo Tree Search (MCTS): MCTS is a popular algorithm for games that involve more complexity, such as Go. It builds a game tree incrementally by simulating random play through (rollouts) from a given state and using the results to evaluate moves [5]. The more simulations that are performed, the more reliable the decision-making becomes. This approach is especially useful for games with a high branching factor where traditional tree search becomes inefficient [14].

Reinforcement Learning (RL): Reinforcement Learning involves training an agent to make decisions based on rewards received from interacting with the environment [1]. The agent learns from its past actions and continuously adjusts its strategy to maximize cumulative reward. In games, RL allows the AI to learn strategies through trial and error. Deep Q-Networks (DQN), an RL technique, is used in scenarios where large state spaces exist, enabling the agent to approximate the value of actions using deep learning [19].

1. Minimax Algorithm: Exponential Growth in Search Tree: The Minimax algorithm suffers from an exponential growth of the search tree. As the number of moves increases, the tree branches out exponentially, making it computationally expensive to evaluate all possibilities, particularly in games with large search spaces like Chess [21]. Ineffective for Complex Games: Minimax is highly effective in simple games, but as the complexity of the game increases (in terms of the number of pieces, moves, or strategic possibilities), its performance degrades [22]. It struggles to find optimal solutions for games with more than a few moves ahead. Limited to Deterministic Games: Minimax performs poorly in games with randomness or incomplete information (stochastic games), as it assumes complete knowledge of the game state at every step [12].

2. Monte Carlo Tree Search (MCTS): High Computational Cost: MCTS relies on simulating many random play through to estimate the value of moves [5]. The number of simulations required for good decision-making is very high, leading to significant computational overhead, especially for large games. Memory Overhead: The memory required to store the outcomes of numerous simulations can be substantial, limiting its scalability and making it less practical for games with large

state spaces. Real-Time Decision Making: MCTS,

while effective in turn-based games, struggles to make timely decisions in real-time strategy games [8]. The required simulations make it too slow for real-time applications, which is a critical limitation in fast-paced games.

3. Reinforcement Learning (RL): Slow Training: RL algorithms require a large number of interactions with the environment (games or simulations) to learn an optimal policy [9]. This can make training extremely slow, especially in complex games with large state spaces, as the agent needs to explore many different strategies before it converges. Exploration vs Exploitation Dilemma: RL agents often struggle with the exploration-exploitation trade off. In the early stages of training, the agent might explore suboptimal moves, leading to inefficient learning [10]. Balancing exploration and exploitation is a challenging aspect, especially in games that require long-term planning. Data and Resource Intensive: Training RL agents requires vast amounts of data and computational resources [4]. The agent must play numerous games or simulations to converge on an optimal strategy, making the approach resource-heavy and time-consuming.

4. PROPOSED SYSTEMS

The system design of the AI agent for strategy-based games focuses on creating a structured framework that enables intelligent decision-making, adaptability, and efficient game play [21]. The AI agent is designed to analyze the game environment, evaluate potential moves, predict opponent strategies, and continuously improve its decision-making through learning mechanisms. This section provides an overview of the system's architecture, data flow, AI methodologies, and interactions between different components.

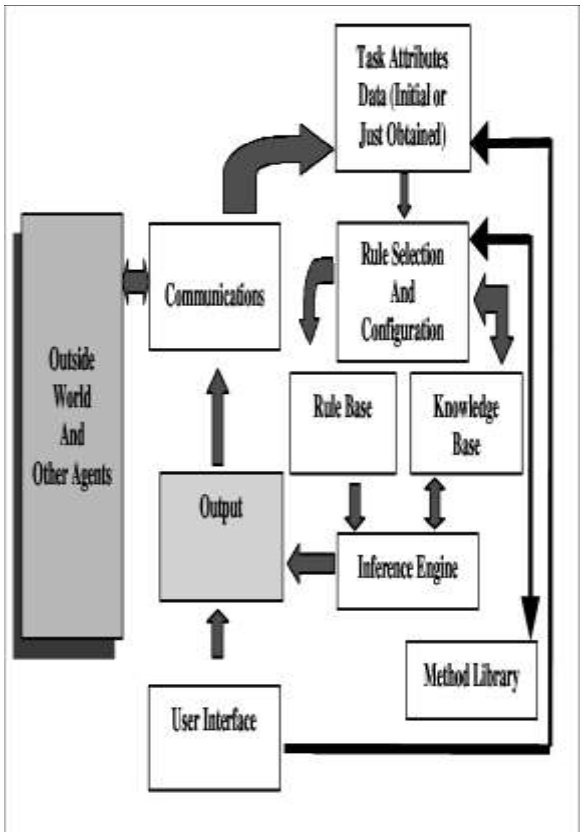

Fig: 1 System Design

4.1 System Architecture: The AI agent follows a modular architecture to ensure flexibility, scalability, and efficient processing. The key modules include: Game Environment Module: Maintains the game state, enforces rules, and determines legal moves. AI Decision-Making Module: Evaluates possible moves, predicts outcomes, and selects the best

action. Learning Module: Incorporates techniques such as reinforcement learning, Monte Carlo Tree Search (MCTS), and Minimax with Alpha-Beta Pruning for continuous improvement. Opponent Modeling Module: Analyzes the playing style of opponents and adapts strategies accordingly. User Interface (UI) Module: Provides a visual representation of the game state, allowing human interaction and move tracking.

4.2. The system is structured in a layered approach for better organization and efficiency: Application Layer: Handles user interactions and manages game inputs/outputs. Processing Layer: Executes AI algorithms, decision-making, and strategy computations. Data Layer: Stores game history, learning models, and relevant datasets for improving AI performance.

4.3 Data Flow and Processing: The AI processes data through a structured pipeline, ensuring real-time decision-making and efficient game play. Game State Representation: The current state of the game is stored in a structured format, allowing quick analysis and computation. Move Generation: The AI identifies all possible legal moves based on the game rules. Move Evaluation: Different algorithms such as MCTS, Minimax, and reinforcement learning analyze each possible move.

4.4 AI Decision-Making Process: The AI decision-making process integrates multiple techniques to achieve optimal game play: Monte Carlo Tree Search (MCTS): Simulates multiple future game scenarios to evaluate the best move. It balances exploration (trying new strategies) and exploitation (using known successful moves). Minimax with Alpha-Beta Pruning: Evaluates possible future game states under the assumption that the opponent will always play optimally. Alpha-Beta pruning helps reduce unnecessary computations, making the process more efficient. Reinforcement Learning (RL): Uses experience-based learning to refine the AI's strategy by rewarding successful decisions and discouraging poor choices. Opponent Modelling: Analyzes past moves of the opponent, identifies patterns, and dynamically adjusts its strategy to counter different play styles.

4.5 Interaction between AI and Game Environment: The AI interacts with the game environment in a continuous feedback loop: The player makes a move, updating the game state. The AI analyzes the updated state and determines the best response. The AI executes the move, leading to a new game state. This process repeats until the game concludes.

5. CONCLUSION

The development of an AI agent for a strategy-based game has been an engaging and challenging project that integrates artificial intelligence with game design principles. The project aimed to create an AI system capable of making intelligent decisions, adapting to different game play scenarios, and providing a dynamic, competitive experience for players. Leveraging advanced techniques like Monte

Carlo Tree Search (MCTS), Deep Q-Networks (DQN), and reinforcement learning, the AI agent was designed to not only perform strategically but also evolve by learning from interactions with the game environment.

The first and foremost challenge in building such an AI was ensuring that it could make optimal decisions based on the game state and the actions of other players. Monte Carlo Tree Search was employed to evaluate different future game states by simulating potential moves and their consequences. This method allowed the AI to consider multiple paths and select the most promising one. Reinforcement learning technique further enhanced the agent's ability to adapt over time by learning from past experiences and gradually improving its strategies. Through these methods, the AI agent was able to predict, adapt, and make decisions that are contextually relevant, demonstrating a significant leap in the sophistication of AI in gaming.

A key element of the project was ensuring that the AI agent could model and respond to the strategies employed by different opponents. Unlike traditional AI, which often follows a fixed set of rules, the agent had to be capable of recognizing patterns in the opponent's behaviour and adapting accordingly. This adaptive behaviour ensured that the game did not become predictable or repetitive, maintaining player interest and challenge.

REFERENCES

[1] Kommineni, K.K., Prasad, A. Enhancing Data Security and Privacy in SDN-Enabled MANETs Through Improved Data Aggregation Protection and Secrecy. Wireless Pers Commun (2024). https://doi.org/10.1007/s11277-024-11635-w

[2] Kumar, K. K., Kumar, S. G. B., Rao, S. G. R., & Sydulu, S. S. J. (2017, November). Safe and high secured ranked keyword searchover an outsourced cloud data. In 2017 International Conference on Inventive Computing and Informatics (ICICI) (pp. 20-25). IEEE.

[3] Dr.K.Sujatha, Dr.Kalyankumar Dasari , S. N. V. J. Devi Kosuru , Nagireddi Surya Kala , Dr. Maithili K , Dr.N.Krishnaveni, " Anomaly Detection In Next-Gen Iot:Giant Trevally Optimized Lightweight Fortified Attentional Convolutional Network," Journal of Theoretical and Applied Information Technology, 15th January 2025. Vol.103. No.1,pages:22-39.

[4] Kalyankumar Dasari*, Dr. K. Venkatesh Sharma, "Analyzing the Role of Mobile Agent in Intrusion Detection System", JASRAE, vol :15, Pages: 566-573,2018.

[5] Kalyan Kumar Dasari&amp, M Prabhakar, "Professionally Resolve the Password Security knowledge in the Contexts of Technology", IJCCIT, Vol:3, Issue:1, 2015.

[6] S Deepajothi, Kalyankumar Dasari, N Krishnaveni, R Juliana, Neeraj Shrivastava, Kireet Muppavaram, "Predicting Software Energy Consumption Using Time Series-Based Recurrent Neural Network with Natural Language Processing on Stack Overflow Data", 2024 Asian Conference on Communication and Networks (ASIANComNet), Pages:1-6, Publisher: IEEE.

[7] S Neelima, Kalyankumar Dasari, A

Lakshmanarao, Peluru Janardhana Rao, Madhan Kumar Jetty, "An Efficient Deep Learning framework with CNN and RBM for Native Speech to Text Translation", 2024 3rd International Conference for Advancement in Technology (ICONAT), Pages: 1-6,Publisher :IEEE.

[8] A Lakshmanarao, P Bhagya Madhuri, Kalyankumar Dasari, Kakumanu Ashok Babu, Shaik Ruhi Sulthana, "An Efficient Android Malware Detection Model using Convnets and Resnet Models", 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Pages :1-6, Publisher :IEEE

[9] Kommineni, K. K., Pilli, R. B., Tejaswi, K., & Siva, P. V. (2023). Attention-based Bayesian inferential imagery captioning maker. Materials Today: Proceedings.

[10] "Customer Churn Prediction in Subscription- Based Businesses Using Machine Learning", https://www.sciencedirect.com/science/article/pii/S1877050919315523.

[11] "Predicting Customer Churn with Machine Learning – A Systematic Review": https://arxiv.org/abs/2001.01537,

[12] Dr.D.Kalyankumar, Kota Nanisai Krishna, Gorantla Nagarjuna, PuvvadaVenkata Naga Sai Jagadesh Kumar, Modepalli Yeswanth Chowdary, "Email Phishing Simulations Serve as a Valuable Tool in Fostering a Culture of Cyber security Awareness", IJMTST, Vol: 10, Issue: 02, Pages:151-157, 2024.

[13] "Deep Learning for Customer Retention in Subscription-Based Services": https://www.sciencedirect.com/science/article/pii/S095741742100587X

[14] Dr.D.Kalyankumar, Muhammad Shaguftha, Putti Venkata Sujinth, Mudraboyina Naga Praveen Kumar, Namburi Karthikeya, "Implementing a Chatbot with End-To-End Encryption for Secure and Private Conversations", IJMTST, Vol: 10, Issue: 02, Pages:130-136, 2024.

[15] Kommineni, K. K. ., & Prasad, A. . (2023). A Review on Privacy and Security Improvement Mechanisms in MANETs. International Journal of Intelligent Systems and Applications in Engineering, 12(2), 90–99. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/4224

[16] Kalyan Kumar Dasari, K Dr , "Mobile Agent Applications in Intrusion Detection System (IDS)'-JASC, Vol: 4

Issue : 5, Pages: 97-103, 2017.

[17] V.Monica, D. Kalyan Kumar, "BACKGROUND SUBTRACTION BY USING DECOLOR ALGORITHM", IJATCSE, Vol. 3 , No.1, Pages : 273 – 277 (2014).

[18] Netflix Prize Dataset (User behavior and ratings): https://www.netflixprize.com

[19]Kaggle: Customer Churn Prediction Datasets: https://www.kaggle.com/datasets

[20] Kalyan Kumar Dasari & Dr, K Venkatesh Sharma, "A Study on Network Security through a Mobile Agent Based Intrusion Detection Framework", JASRAE, vol : 11, Pages: 209-214, 2016.

[21] Dr.D.Kalyankumar, Saranam Kavyasri, Mandadi Mohan Manikanta, Pandrangi Veera Sekhara Rao, GanugapantaVenkata Pavan Reddy, "Build a Tool for Digital Forensics to Analyze and Recover Information from Compromised Systems", IJMTST, Vol: 10, Issue: 02, Pages:173-180, 2024.

[22] Kalyankumar Dasari, Mohmad Ahmed Ali, NB Shankara, K Deepthi Reddy, M Bhavsingh, K Samunnisa, "A Novel IoT-Driven Model for Real-Time Urban Wildlife Health and Safety Monitoring in Smart Cities" 2024 8th International Conference on I-SMAC, Pages 122-129.

[23] Dr.D.Kalyankumar, Panyam Bhanu Latha, Y. Manikanta Kalyan, Kancheti

Deepu Prabhunadh, Siddi Pavan Kumar, “A Proactive Defense Mechanism against Cyber Threats Using Next-Generation Intrusion Detection System”, IJMTST, Vol: 10, Issue: 02, Pages:110-116, 2024.