Advancing Board Game AI: A Case Study of Reinforcement Learning in Quixo*

*Note: This is a draft for a possible publication

Ali Al Housseini
Department of Control and Computer Engineering (DAUIN)
Politencico di Torino
Turin, Italy
ali.alhousseini@studenti.polito.it

***Abstract*—In the realm of artificial intelligence and game theory, the application of Reinforcement Learning (RL) to board games [1] has shown remarkable success. From Chess to Go, AI has not only mastered these games but also unveiled new strategies. This article introduces a pioneering project in this field: the development of an RL model for Quixo, a less-explored terrain in AI research.**

***Index Terms*—Reinforcement Learning, Board Games, Hyper-parameter tuning, Fast Convergence, Reward Shaping**

I. Introduction

The integration of Artificial Intelligence (AI) in board games represents a pivotal chapter in the evolution of AI research. Historically, board games have served as a benchmark for the capabilities of AI systems. The journey began with early experiments like Arthur Samuel's checkers-playing program in the 1950s, which marked the first successful self-learning program. This milestone paved the way for a series of advancements, leading to more complex games being tackled by AI. The landmark victory of IBM's Deep Blue over chess grandmaster Garry Kasparov in 1997 was a testament to the growing prowess of AI in strategic game play.

The explosion of interest in applying AI to board games can be attributed to several factors. Firstly, board games, with their clear rules and objectives, provide an ideal testing ground for AI algorithms. They offer a controlled environment to simulate strategic decision-making and problem-solving, core aspects of AI research. Additionally, the development of more sophisticated AI techniques, especially in the realm of machine learning and neural networks, has enabled researchers to tackle more complex games. These games, once thought to be beyond the reach of AI, are now new frontiers being conquered.

Quixo, while less known than Chess or Go, is a game rich in strategic depth. Invented in 1995 by Thierry Chapeau, Quixo is a tactile game akin to a dynamic form of Tic-Tac-Toe. Players slide cubes in a 5x5 grid, aiming to align five cubes bearing their symbol. Its simplicity in design belies the strategic complexity it harbors, making it an intriguing candidate for AI research. Unlike the extensively studied Chess or Go, Quixo presents a relatively unexplored challenge for AI, offering fresh ground for innovative research.

The decision to employ Reinforcement Learning (RL) for Quixo stems from its proven success in other board game applications. RL, a type of machine learning where an agent learns to make decisions by performing actions and receiving feedback, is well-suited for the strategic and decision-making elements inherent in board games. In the case of Quixo, RL's ability to learn from the dynamic game environment and adapt its strategy presents a promising approach. The game's unique mechanics of cube sliding and rotation offer a distinct challenge, aligning well with the strengths of RL in exploring and exploiting various strategies through iterative learning.

II. Related Work

The integration of Artificial Intelligence (AI) into board games has been a significant area of research, dating back to the early days of AI development. The application of AI in board games is not a new phenomenon; it has been a fixture since the 1950s when the first AI programs were developed to play games like checkers and chess at the University of Manchester [1]. The field gained substantial momentum with notable milestones, such as IBM's Deep Blue defeating chess grandmaster Garry Kasparov in 1997 and the development of Google DeepMind's AlphaZero.

Interestingly, while classic games like chess have been extensively researched, contemporary strategy board games have received less attention from the AI community, despite their growing popularity [2]. These contemporary games often involve multiple players and elements of chance and uncertainty, presenting unique challenges and opportunities for modern AI algorithms. The exploration of AI in these games could provide invaluable insights into algorithmic complexity and adaptive learning strategies.

AI applications in board games span a wide spectrum, from simple algorithmic solutions to complex learning models. Research has demonstrated that simpler games like Tic-tac-toe can be effectively mastered with basic AI algorithms. However, more intricate games like chess and Go require advanced strategies and optimization techniques [3]. The development of

these AI systems has not only pushed the boundaries of what is algorithmically possible but also enhanced our understanding of strategic decision-making.

The exploration of AI in board games, particularly through reinforcement learning, continues to be a promising area of research. As evidenced by previous studies and implementations, RL's capability to adapt and optimize strategies in complex, dynamic environments makes it well-suited for board games. The application of RL to a less-studied game like Quixo not only adds to this body of work but also opens up new avenues for exploration in AI research.

III. Quixo Game

In the realm of board game strategy and artificial intelligence (AI), Quixo emerges as a fascinating subject, blending the allure of abstract strategy with the challenges of dynamic gameplay. This section delves into the existing research related to Quixo and its unique mechanics, shedding light on the intricacies of applying AI to this intriguing game.

A. Quixo in Research

In the landscape of AI-driven board game analysis, "Quixo Is Solved" stands out as a seminal piece of research. This groundbreaking study conducted a comprehensive analysis of Quixo, employing a blend of value iteration and backward induction techniques. The researchers faced the daunting task of navigating through the game's complexity within reasonable computational limits. Their findings were revelatory, asserting that a standard 5×5 Quixo game, played optimally, invariably concludes in a draw. This insight not only showcased the game's balanced nature but also highlighted the depth of strategic possibilities inherent in Quixo [5].

Quixo's examination within the AI context gains further dimension when viewed against the backdrop of similar abstract strategy games. Studies in this domain have often focused on classics like Hex, Awari, and Checkers. For instance, Hex, known for its strategic depth, was shown to favor the first player, a conclusion derived from Nash's strategy-stealing argument. Such analyses, while not directly related to Quixo, offer valuable parallels in understanding the game's strategic landscape.

The evolution of computer-assisted techniques has been pivotal in unraveling the mysteries of these games. The intricate dynamics of games like Connect-Four and Awari have been decoded through extensive computational efforts, demonstrating the power of AI in extracting strategic insights from seemingly simple setups.

Further enriching this field are generalized studies in games like Go and Poker. These works have put forth strategies transcending human capabilities, contributing significantly to the broader understanding of AI applications in board games. Although these studies do not offer definitive solutions, they provide a framework for approaching complex, strategy-driven games like Quixo.

B. An Overview of the Game

Unlike many of its contemporaries, Quixo is not bound by a predetermined number of turns, a characteristic that adds an element of unpredictability. The game's design prompts intriguing questions about its theoretical endpoint, especially in scenarios where both players are equally matched and play optimally. This open-ended nature of Quixo makes it a particularly intriguing subject for AI exploration, as it challenges conventional approaches to game termination and strategy formulation.

Quixo is mostly a two-player game that is played on a 5×5 grid, also called board. Each grid cell, also called tile, can be empty, or marked by the symbol of one player: either X or O.

At each turn, the active player first (i) takes a tile $-$ empty or with her symbol $-$ from the border (i.e. excluding the 9 central tiles), and then (ii) inserts it, with her symbol, back into to the grid by pushing existing tiles toward the hole created by the tile removal in step (i). The winning player is the first to create a line of tiles all with her symbol, horizontally, vertically, or diagonally. Note that if a player creates two lines with distinct symbols in a single turn, then the opponent is the winner.

Figures "Fig. 1" and "Fig. 2" show the real game and our corresponding representation. Figure "Fig. 3" depicts the resulting board after a valid turn by player O from the board depicted in "Fig. 2": player O first (i) takes the rightmost (empty) tile of the second row, and then (ii) inserts it at the leftmost position shifting the other tiles of this second row to the right.

Fig. 1. Real Game

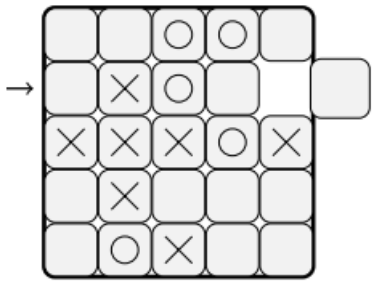

Fig. 2. Simplified Illustration

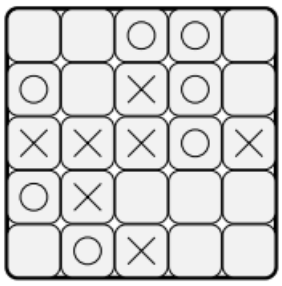

Fig. 3. After a valid move from player O

Quixo bears an immediate resemblance with some classical games such as Tic-Tac-Toe, Connect-Four, or Gomoku. However there are two major differences: (1) the board is "dynamic"; a placed $XorO$ may change its location in subsequent turns, (2) the game is unbounded (in term of turns). The first point is what makes the game interesting to play: dynamicity makes Quixo very difficult for a human player to plan more than a couples of turns. The second point raises immediately a natural scientific question about termination. Trivially, players could "cooperate" to create an infinite game, as official rules do not specify any terminating rules, such as the 50-move or the threefold repetition rules of chess. For example, by always taking the same tiles and having only one X and one O on the board. However, it remains unclear whether an infinite game exists when both players play to win.

IV. Methodology

A. Reinforcement Learning

Reinforcement Learning (RL) [6] is a paradigm of machine learning that focuses on how agents ought to take actions in an environment to maximize some notion of cumulative reward. Distinct from other types of learning, like supervised or unsupervised learning, RL is centered on the decision-making process, learning from interaction, and based on the concept of trial and error.

Key Components of RL:

- Agent: The learner or decision maker.
- Environment: The physical or virtual setting in which the agent operates.
- Action (A): All possible moves the agent can make.
- State (S): A representation of the current condition of the environment.
- Reward (R): An immediate return sent back to the agent from the environment to evaluate the last action.
- Policy (π): A strategy used by the agent, mapping states to actions
- Value Function: A prediction of expected future rewards.
- Model of the Environment: This might be used for planning, predicting the next state and reward.

The process involves the agent interacting with the environment in discrete time steps. At each time step, the agent receives the state of the environment, selects an action based on its policy, receives a reward and the next state, and updates its policy based on the feedback from the environment.

A critical aspect of RL is the balance between exploration (trying new things) and exploitation (using known information). This balance is crucial for the agent to learn effectively.

RL has been successfully applied in various domains, including robotics, autonomous vehicles, finance, healthcare, and games, demonstrating its versatility and effectiveness in complex decision-making scenarios.

Reinforcement Learning represents a powerful framework for autonomous decision-making and has witnessed significant advancements and applications in recent years. Its unique approach of learning through interaction and feedback makes it suitable for a wide range of applications, particularly in environments where explicit programming of all potential scenarios is impractical.

RL algorithms can be broadly classified into three categories:

- Value-Based Algorithms: These algorithms focus on optimizing the value function, which estimates how good it is to be in a given state (or how good it is to perform a certain action in a given state). The goal is to maximize the expected return. A well-known example is Q-learning.
- Policy-Based Algorithms: These directly learn the policy function that maps states to the best action to take. They optimize the policy directly without the need for a value function. An example is the REINFORCE algorithm.
- Actor-Critic Methods: These combine the elements of both value-based and policy-based methods. They utilize two models: the actor, which updates the policy distribution, and the critic, which estimates the value function. A common example is the Advantage Actor-Critic (A2C) algorithm.

Proximal Policy Optimization (PPO) [7] falls under the category of policy-based RL algorithms. Developed by OpenAI, PPO has gained popularity due to its effectiveness and simplicity. It addresses some of the key challenges in policy optimization, particularly in balancing exploration and exploitation and in stabilizing the training process.

Key Features of PPO:

- Simplicity and Efficiency: PPO is simpler to implement and tune compared to its predecessors like Trust Region Policy Optimization (TRPO). Despite this simplicity, it achieves high efficiency and robust performance.
- Clipped Objective Function: One of the novel features of PPO is its objective function, which uses a clipping mechanism to prevent the policy from changing too drastically during each update. This clipping limits the update step size, promoting gradual policy improvement and helping to maintain training stability.
- Multiple Epochs of Stochastic Gradient Descent: PPO allows for multiple epochs of mini-batch updates to the neural network policy, improving sample efficiency.

B. QuixoRL environment

The development of a Reinforcement Learning (RL) model for the Quixo game necessitates a meticulous approach to environment definition and algorithm implementation. This section outlines the methodology employed in creating an environment conducive to RL, focusing on the key components that facilitate the training and functioning of the AI agent.

1) Defining the Game Environment: At the heart of our RL model lies the game environment, a virtual representation of Quixo's playing field. This environment is pivotal for simulating game dynamics and providing a sandbox for the AI agent to learn and adapt.

The initializer sets the stage for the game environment. It creates a game space responsible for processing player moves and reflecting these on the Quixo board. Integral to this setup is the n_steps variable, a counter that tracks the number of step function calls. This counter is not just a mere record of moves but plays a crucial role in refining the AI's decision-making process. Additionally, a $limit_steps$ variable is set as a threshold to ensure the n_steps does not exceed a predefined limit, thus maintaining control over the game's progression.

Reward shaping is an art in itself within the RL paradigm. In this project, the method involves defining the nature of rewards returned by the environment. The objective is to subtly manipulate these rewards to encourage the AI agent to explore a wider range of actions, thus enhancing its learning experience. This method is instrumental in nudging the agent towards more diverse and potentially successful strategies.

2) Utilities: An array of utility functions supports the main RL environment, each serving a specific purpose in the gameplay dynamics.

1. Action Decoder:

Central to the Quixo game mechanics is the action decoder. This utility interprets the agent's actions, which involve selecting a tile from the game board's periphery and sliding it in an allowed direction. The decoder ensures that each action adheres to the game's rules – for example, preventing a tile from being moved back to its original position. The action space, therefore, is a series of integers ranging from 0 to 43, each representing a unique move on the game board. The calculation of these moves considers the limited options for corner tiles and the broader possibilities for other tiles, resulting in a structured yet diverse set of potential actions.

2. Action Encoder:

The action encoder functions as the converse of the action decoder. It translates specific coordinates and directions into an integer format, effectively encoding the agent's strategic decisions into a language the game environment can interpret and respond to.

3) The step function: The step function is where the game's core interaction takes place. It instructs the agent on how to execute a move and imposes a negative reward for incorrect or invalid selections. This function is pivotal in guiding the agent's learning process, ensuring that each action is a calculated step towards mastering Quixo.

4) The reset function: Essential to any RL environment is the ability to revert to a baseline state. The reset function in our Quixo environment serves this purpose. It returns the game to its initial state, providing a clean slate for each new iteration of the game. This function is crucial for maintaining the integrity of the learning process, ensuring that each new game is an independent trial for the AI agent.

V. Training

The training of an AI agent for Quixo is a multifaceted process, involving the creation of specialized environments, hyperparameter tuning, and iterative training procedures. Each aspect plays a crucial role in developing a robust AI capable of mastering the complexities of Quixo.

A. Trained Environments

1. QuixoRandom: Balancing Exploration and Exploitation

The first environment, named QuixoRandom, serves as the initial training ground for our agent. In this environment, the agent competes against a player executing random but valid moves, without a coherent strategy for winning. This setup provides an excellent platform for the agent to learn the fundamentals of the game without being overwhelmed by advanced strategies. The rewards are described in "Table I".

TABLE I
REWARD SHAPING IN QUIXORANDOM

Feature	Description
Invalid Move Penalty	Selecting a non-valid action, such as a move belonging to the opponent, incurs a reward of -5. This teaches the agent basic game rules.
Winning and Losing	Winning the game grants a reward of +30, while losing results in a -10 penalty. These rewards emphasize the ultimate goal of winning.
Survival Reward	A dynamic reward encourages the agent to extend the game duration, facilitating deeper exploration of possible strategies. Beyond the 100th step, a constant reward maintains the incentive for prolonged play.
Consecutive Number Alignment	Aligning 3 or 4 numbers in a row, column, or diagonal is rewarded, subtly guiding the agent towards effective strategies.
Limit Step Punishment	To ensure games conclude within a reasonable timeframe, a decreasing limit on the number of steps is enforced. This promotes efficiency in strategy development.

2. QuixoSelf: Learning Advanced Strategies

The second environment, QuixoSelf, is designed to elevate the agent's gameplay by competing against a version trained in the QuixoRandom environment. This approach simulates a high-level competition, pushing the agent to develop sophisticated strategies. Rewards can be found in table "Table II"

B. Hyperparameter Tuning

The success of an RL model heavily relies on finding the right balance in its hyperparameters. Optuna [8], an automated hyperparameter tuning framework, was utilized to optimize key parameters for the PPO algorithm.

TABLE II
REWARD SHAPING IN QUIXORANDOM

Event	Reward and Description
Invalid Move Penalty	A harsher penalty of -20 for invalid moves enforces strict adherence to game rules.
Win-Lose Rewards	The agent receives a high reward of +30 for winning and a substantial penalty of -30 for losing, sharply distinguishing the outcomes.
Step Limit for Quick Resolution	A -20 penalty for games exceeding 50 steps encourages the agent to find efficient winning strategies quickly.

1) Learning Rate: A crucial parameter determining the optimization step size. An optimal balance is sought to avoid both slow convergence and overshooting.

The learning rate is used in the gradient descent update rule. For a parameter vector θ, the update rule in the context of policy gradient methods like PPO is typically:

$$\theta_{new} = \theta_{old} + \alpha \cdot \nabla_\theta J(\theta) \tag{1}$$

This equation represents the parameter update rule in gradient descent, where $\nabla_\theta J(\theta)$ is the gradient of the objective function concerning the parameters.

2) Gamma: The discount factor for future rewards, crucial for balancing immediate and future gains.
Gamma is used in the calculation of the discounted sum of future rewards, known as the return G_t from a time step t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2}$$

Where R_{t+k} is the reward received at time $t+k$. The discount factor γ determines the present value of future rewards; a smaller γ emphasizes immediate rewards more.

3) GAE Lambda: This parameter in Generalized Advantage Estimation helps balance bias and variance, influencing how the agent values long-term rewards.
Generalized Advantage Estimation (GAE) is used to estimate the advantage function A_t, which is a measure of how much better an action is compared to the average. The GAE, incorporating λ, is given by:

$$A_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1} \tag{3}$$

Where $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is the temporal difference error, and $V(S_t)$ is the value function at state S_t. The parameter λ balances the bias-variance trade-off in this estimation.

4) Batch Size: Determines the number of training samples per optimization iteration, impacting learning stability and computational demands.

5) N steps: Defines the number of steps collected before updating the policy, a critical factor in sample and computational efficiency. In PPO, n steps refers to the number of steps used to compute the returns and advantage estimates. The returns G_t over n steps can be calculated as:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{n+1} + \gamma^n V(S_{t+n}) \tag{4}$$

This is used when the returns are truncated after n steps and bootstrapped with the value function estimate $V(S_{t+n})$

6) Entropy Coefficient: Encourages policy exploration by penalizing deterministic behavior, essential for preventing premature convergence to suboptimal strategies.

The entropy bonus is added to the objective function to encourage exploration. The modified objective function with the entropy term is:

$$J(\theta)_{new} = J(\theta)_{old} + ent_{coef} H(\pi(\theta)) \tag{5}$$

Here, $H(\pi(\theta))$ is the entropy of the policy π parameterized by θ, and ent_{coef} is the entropy coefficient. This term encourages the policy to maintain a level of randomness (exploration).

For the first model trained on the first environment, the following parameters yielded optimal results:

TABLE III
HYPERPARAMETERS OF QUIXORANDOM

Parameter	Values
Learning Rate	0.00047987507467331137
Gamma	0.9037646415026495
GAE Lambda	0.8496204061939574
Batch Size	256
N steps	930
Entropy	0.04630735973520918

While for the second model trained on the second environment, the results are shown below:

TABLE IV
HYPERPARAMETERS OF QUIXOSELF

Parameter	Values
Learning Rate	3.8516223541221167e-05
Gamma	0.935919311668545
GAE Lambda	0.869609567097365
Batch Size	256
N steps	997
Entropy	0.08256606818578316

C. Training

Building upon the foundational training setup outlined in the previous section, this segment delves into the advanced stages of training the AI agent for Quixo. This phase is characterized by iterative refinement and exposure to increasingly complex scenarios, ensuring the development of a highly competent AI capable of sophisticated gameplay.

After the initial training against a random player, the model underwent a critical transition. The AI, now attuned to the game's basic dynamics, was introduced to the QuixoSelf environment. This environment represented a significant leap in complexity, as the AI was pitted against a version of itself that had been optimized in the previous training phase. This self-play methodology is a powerful tool in AI training, often leading to rapid improvements in strategic depth and decision-making.

Post-transition, the AI underwent another round of training and tuning, this time against a more sophisticated version of itself. This stage was crucial for fine-tuning the AI's strategy, ensuring it not only understood the game's mechanics but also developed advanced tactics. The tuning process was guided by the previously established optimal hyperparameters, providing a well-calibrated framework for the AI's learning.

The culmination of these efforts is reflected in the data presented in Table 4. This table encapsulates the journey of the AI from its initial steps to becoming a formidable Quixo player, charting its progress and the nuanced adjustments made along the way to refine its capabilities.

The final training phase involved an extensive 6 million timesteps session, a duration surpassing the combined length of the initial training stages. During this phase, the AI was exposed to a diverse array of game scenarios through a set of parallelized environments, randomly alternating between QuixoRandom and QuixoSelf.

This diversity in training scenarios is crucial. By facing both a random player and its previous, more advanced self, the AI was continuously challenged, preventing it from settling into predictable patterns. This approach ensures a well-rounded development, equipping the AI with the ability to adapt to various game situations and strategies.

The key to the AI's success in mastering Quixo lies in this final, extended training phase. The combination of duration and diversity in training environments allowed the AI to thoroughly explore the game's strategic landscape. It refined its tactics against both unpredictable and highly strategic opponents, an essential factor in developing a robust AI player.

VI. Results

The culmination of rigorous training and methodical refinement of the QuixoRL agent is reflected in its performance during testing. The results underscore the agent's exceptional capability, marking a significant achievement in the application of Reinforcement Learning (RL) to board games.

In a series of 1000 games against a random player, the QuixoRL agent demonstrated remarkable dominance, securing victories in 985 games. This high win rate of 98.5% is indicative of the agent's ability to navigate the game's dynamics adeptly, making strategic moves that capitalize on the randomness of its opponent. The agent's performance in this scenario underscores its proficiency in handling games where the opponent's moves are unpredictable but lack strategic depth.

A more challenging test of the QuixoRL agent's capabilities came in the form of self-play, where it competed against a copy of itself. In this rigorous testing environment, the agent won 886 out of 1000 games, translating to an 88.6% win rate. These results are particularly impressive as they demonstrate the agent's ability to engage in strategic gameplay, countering moves from an opponent with an identical level of skill and strategy. The high win rate in self-play highlights the agent's advanced strategic understanding and adaptability.

The performance of the QuixoRL agent in these testing scenarios suggests that it has reached a level of proficiency that surpasses human players. This accomplishment is notable for several reasons:

- Complexity and Unpredictability: Quixo, with its dynamic board and strategic depth, presents a complex challenge. The agent's ability to consistently win in this environment speaks volumes about its strategic planning and execution capabilities.
- Adaptability and Learning: The results from self-play illustrate the agent's capacity for learning and adaptation. Competing against a version of itself, the agent demonstrated the ability to continually refine and adjust its strategies.
- A Benchmark in Board Game AI: These results set a new benchmark in the field of AI for board games. They showcase the potential of RL in mastering games that require both strategic depth and adaptability, extending beyond the realms of traditional board games like Chess or Go.

VII. Acknowledgment

As this chapter of my academic journey draws to a close, I find myself reflecting on the invaluable support and guidance I have received. I would like to extend my heartfelt gratitude to Prof. Giovanni Squillero and Phd. Andrea Calabrese whose contributions have been instrumental in my learning and growth throughout this semester.

Both of you have played pivotal roles in shaping my academic path this semester. The lessons learned and the skills acquired under your tutelage extend far beyond the confines of the classroom. Your guidance has been a fundamental part of my journey, and I am deeply appreciative of the time, effort, and commitment you have devoted to my education and development.

Thank you for a truly enriching and enlightening semester. Your support and encouragement have been invaluable, and I am grateful for the opportunity to have learned from such dedicated and knowledgeable educators.

VIII. Conclusion and Future Work

The development and subsequent success of the QuixoRL agent in mastering the strategic board game Quixo stands as a notable achievement in the field of artificial intelligence. This endeavor has not only demonstrated the efficacy of sophisticated reinforcement learning techniques in board games but also set a new benchmark for AI performance in strategic reasoning and adaptability.

Looking ahead, there are several promising avenues for future research and application, building upon the foundations laid by the QuixoRL project:

- Application to Other Board Games: Leveraging the methodologies and insights gained from this project, future work could involve adapting the QuixoRL model to other complex board games.
- Improving Human-AI Interaction: Exploring the integration of the QuixoRL agent in educational or recreational settings could provide a unique platform for human-AI interaction.
- Enhancing AI Learning Algorithms: Continuing to refine and improve the underlying reinforcement learning algorithms will be crucial.

References

[1] de Mesentier Silva, F., Lee, S., Togelius, J., & Nealen, A. (2017). AI-based playtesting of contemporary board games. In Proceedings of the 12th International Conference on the Foundations of Digital Games (pp. 1-10).

[2] Eger, M., & Martens, C. (2019, October). A Study of AI Agent Commitment in One Night Ultimate Werewolf with Human Players. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 15, No. 1, pp. 139-145).

[3] Chengyi Jiang, 'The application of artificial intelligence in board games'.

[4] Wolfgang Konen, ' General Board Game Playing for Education and Research in Generic AI Game Learning', Computer Science Institute - TH Köln – Cologne University of Applied Sciences.

[5] Satoshi Tanaka, et al., "Quixo is Solved".

[6] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, Anil Anthony Bharath, 'A Brief Survey of Deep Reinforcement Learning', https://doi.org/10.48550/arXiv.1708.05866

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, 'Proximal Policy Optimization Algorithms', https://doi.org/10.48550/arXiv.1707.06347

[8] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, Masanori Koyama, 'Optuna: A Next-generation Hyperparameter Optimization Framework', arXiv:1907.10902v1