PPO Hex Agent

1st Ivan Birkmaier
Master student
FH Technikum Wien
Vienna, Austria
ai23m049@technikum-wien.at

2nd Leonard Hübner
Master student
FH Technikum Wien
Vienna, Austria
ai23m040@technikum-wien.at

3nd Herbert Grünsteidl
Master student
FH Technikum Wien
Vienna, Austria
ai23m012@technikum-wien.at

***Abstract*—This paper presents an exploration of training Proximal Policy Optimization (PPO) agents to play the 7x7 Hex game. Hex is a strategic board game, with a large state space and a need for long-term planing, therefore posing a relevant challenge in the field of reinforcement learning. Six experiments were conducted to evaluate different training methodologies, focusing on various reward shaping techniques and opponent configurations. The results highlighted the agents' ability to achieve high win rates against random opponents but revealed significant challenges in adaptability and strategic depth, particularly when switching roles or facing more complex strategies. Future work should address these challenges by simplifying reward structures, extending training durations, and incorporating periodic evaluations to enhance learning efficacy and strategic proficiency.**

***Index Terms*—Hex, PPO, Reinforcement Learning**

I. Introduction

The goal of this project was to implement a Proximal Policy Optimization [1] (PPO) agent capable of playing a 7x7 Hex game. Hex, a strategic board game, presents a complex challenge for artificial intelligence due to its large state space and the necessity for long-term strategic planning. This project aimed to explore various training methodologies for PPO agents to achieve competent gameplay performance.

To achieve this, multiple experiments were designed and conducted. Each experiment focused on training agents using different approaches to optimize their learning and gameplay capabilities. The diversity in training strategies aimed to provide insights into the most effective methods for developing a strong PPO agent for Hex. Importantly, across all experiments, the same initial agent configuration was employed to ensure consistency and fairness in the evaluation process.

The Evaluation of the experiments where evaluated either against other trained agents or against random agents. A pipeline to let the agents compete against each other was developed and used for evaluation of the experiments with non random evaluation. Through these matches we aimed to find the best overall agent.

II. Methods

A. Agent Setup

This chapter provides an overview of the core functionalities of the PPO agent, which we have rebuilt from Tabor Phil [2], and details the initialization parameters used in the project. The PPO agent comprises several key components: the memory management system (`PPOMemory`), the actor network (`ActorNetwork`), the critic network (`CriticNetwork`), and the overarching agent class (`Agent`) that integrates these components. Each part plays a crucial role in the agent's ability to learn and perform in the 7x7 Hex game.

1) PPOMemory: The `PPOMemory` class manages the storage and retrieval of experiences during the training process. It maintains lists for states, actions, probabilities, values, rewards, and done flags. Key methods include:

- `__init__(self, batch_size)`: Initializes memory storage lists and sets the batch size.
- `generate_batches(self)`: Generates batches of experiences for training.
- `store_memory(self, state, action, probs, vals, reward, done)`: Stores a single experience in memory.
- `clear_memory(self)`: Clears all stored experiences.

2) ActorNetwork: The `ActorNetwork` class defines the neural network architecture and operations for the policy (actor) network. This network outputs action probabilities given an input state.

- `__init__(self, n_actions, input_dims, alpha, fc1_dims=64, fc2_dims=64, chkpt_dir='tmp/ppo')`: Initializes the network layers, optimizer, and device settings. The activation function was Tanh.
- `forward(self, state)`: Executes a forward pass through the network.
- `save_checkpoint(self)`: Saves the current network state to a file.
- `load_checkpoint(self)`: Loads the network state from a file.

3) CriticNetwork: The `CriticNetwork` class defines the neural network architecture and operations for the value (critic) network. This network outputs a value estimate given an input state.

- `__init__(self, input_dims, alpha, fc1_dims=64, fc2_dims=64, chkpt_dir='tmp/ppo')`: Initializes the network layers, optimizer, and device settings. As activation function Tanh was used again.
- `forward(self, state)`: Executes a forward pass through the network.

- `save_checkpoint(self)`: Saves the current network state to a file.
- `load_checkpoint(self)`: Loads the network state from a file.

4) Agent: The `Agent` class integrates the actor and critic networks and manages the interaction between them during training and inference.

- `__init__(self, n_actions, input_dims, gamma=0.99, alpha=0.0003, gae_lambda=0.95, policy_clip=0.2, batch_size=2, n_epochs=10, chkpt_dir='tmp/ppo')`: Initializes the actor and critic networks, memory, and hyperparameters.
- `remember(self, state, action, probs, vals, reward, done)`: Stores an experience in memory.
- `save_models(self)`: Saves the actor and critic models.
- `load_models(self)`: Loads the actor and critic models.
- `choose_action(self, observation, allow_illegal_moves=False)`: Selects an action based on the current policy. The first challenge for the PPO agent was to learn legal moves in the Hex game, cause each Hex cell can be occupied only once. To address this challenge, we implemented a maskable action space [3] within the action selection process. By assigning a zero probability to already occupied cells, we effectively prevented the agent from selecting these cells. Consequently, the agent was constrained to consider only the available free cells for its moves. This approach improves the agent's learning efficiency and performance, as it no longer needs to explore and learn the invalid moves, which would otherwise waste computational resources and learning time. By focusing on legal moves, the agent can better understand the strategic aspects of the game, leading to more effective policy optimization.
- `learn(self)`: Executes the learning process using stored experiences.

5) Initialization Parameters: The following hyperparameters and settings were used to initialize the PPO agent:

- `board_size = 7`
- `N = 6` % Frequency of learning
- `batch_size = 3` % Batch size
- `n_epochs = 2` % Number of epochs
- `alpha = 0.0005` % Learning rate
- `n_actions = board_size * board_size` % Number of possible actions (for 7x7 board)
- `input_dims = [board_size * board_size]` % Flatten input dimensions
- `gamma = 0.99` % Discount factor
- `gae_lambda = 0.95` % GAE lambda for advantage estimation
- `policy_clip = 0.2` % Clip value for PPO loss

These parameters were chosen to balance learning efficiency and computational feasibility, ensuring the agent could be trained effectively within the project's scope. The actor and critic networks were both designed with multiple fully connected layers to capture the complex relationships in the Hex game state space.

B. Experiments

To train a PPO agent for the Hex game, we conducted six experiments:

1) The agent plays for only one side against a random opponent.
2) The agent alternates between playing as white and black in phases against a random opponent.
3) The agent plays the first half of the training as white and the next half as black against a random opponent and after that against it's own version of the side.
4) The agent plays against a random opponent on varying board sizes, from 3x3 to 7x7.
5) Two agents play against each other and learn in phases.
6) The agent is trained in phases against a random agent and against the best agent of the previous phase. All agents are compared against each other

C. Reward Shaping Method

Before delving into the experiments, it is crucial to explain the reward shaping method used. A structured reward mechanism guides the agents' learning process [4]. The rewards and penalties are designed to promote strategic play:

1) The highest reward was given for winning a game.
2) A reward was given for blocking the opponent, i.e., placing a stone to prevent the opponent from connecting their stones.
3) A reward was given for connecting the agent's own stones.
4) A reward was given for blocking the opponent's winning move, i.e., placing a stone to prevent the opponent from connecting six stones.
5) An additional reward was given if the agent recognized it could win with the next move.
6) The fewer moves the agent took to win, the less was subtracted from the total reward.
7) Similar to reward 2, a reward was given for placing a stone between two opponent's stones to block their connection.

Depending on the experiment, some of these reward shaping methods were applied.

D. Training and Experiment Setup

In the following sections we will discuss each individual experiment setup. In order to be able to track the training, we had logs for agent(s) score, win rate, timesteps, moves per game and learning steps per game.

*1) **Experiment 1**:* In the first experiment, the PPOagent was trained to play the Hex game against an opponent making random moves. The agent was trained exclusively for one side, either white or black, over the course of 2000 games. The reward structure for this experiment was straightforward: the agent received 1 point for each winning game. Additionally, reward shaping method 6 was applied with the formula:

$$\text{reward} -= \text{moves} \times 0.01$$

This adjustment penalized the agent by 0.01 points for each move made during the game, encouraging the agent to develop strategies that led to quicker wins.

*2) **Experiment 2**:* In this experiment, the objective was to train a single PPO agent to proficiently play both sides, white and black, in the Hex game. To achieve this, the agent alternated sides every 200 games throughout a total of 2000 games against a randomly playing opponent. The reward structure for the agent remained consistent with previous experiment. By alternating sides, the agent was exposed to both perspectives of the game, allowing it to develop a more holistic understanding of Hex. This switching mechanism ensured that the agent did not become biased towards the strategies effective for only one side, fostering balanced skill development. This setup aimed to create a more versatile and robust agent capable of handling a variety of situations in the Hex game, ultimately leading to improved adaptability.

*3) **Experiment 3**:* In this experiment, we aimed to train an agent that played as the white player during the first half of the training and as the black player during the second half of the training, adjusting its strategy based on its role and evaluating its adaptability under varying conditions. The training was meticulously structured in several distinct phases, each designed to challenge and refine the agent's abilities:

1) *Initial Training Against Random Opponent*:
   - Initially, the agent was trained against a random opponent, a strategy that introduces unpredictability and compels the agent to adapt to a wide range of potential moves without prior bias.
   - During this phase, the agent was rewarded with a score of 1 for winning and 0 for losing. To incentivize efficiency, the agent received additional rewards based on the speed of victory, encouraging it to win in fewer moves.
   - Checkpoints of the agent were saved whenever the agent's average score surpassed the best score recorded thus far, serving as a benchmark for its progression.
2) *Training Against Best Previous Model*:
   - Following the initial random phase, the agent was pitted against its best previous version for a set number of games. This phase aimed to refine strategies by challenging the agent with its own most effective tactics, thus simulating a more intense and strategic environment.

During each game, the agent alternated between playing as the white player (player 1) and the black player (player -1). Specifically, the agent played as the white player during the first 1500 games of the training and switched to playing as the black player for the remaining 1500 games.

*4) **Experiment 4**:* In this experiment, we aimed to train an agent by progressively increasing the complexity of the game board. Initially, the agent was trained on a 3x3 board, and after a certain number of games, the board size was incrementally increased by one dimension until it reached the final size of 7x7.

The agent was initialized with a 7x7 board. The Hex engine's board was then down-sampled for the initial stages of training. Specifically, the sides where player 1 (white) was supposed to play were filled with 1s, and the sides where player 2 (black) was supposed to play were filled with -1s. The agent consistently played as the white side (player 1). The training was conducted in several phases:

1) *Initial Training Against Random Opponent*:
   - The agent was first trained against a random opponent who made moves randomly.
   - The reward system for the agent was simple: 1 point for a win and 0 points for a loss.
   - During this phase, checkpoints of the agent were saved whenever the agent's average score surpassed the best score recorded thus far.
2) *Training Against Best Previous Model*:
   - After completing the training against the random opponent, the agent was further trained for a certain number of games against its best previous version.
   - In this phase, the agent consistently faced increasingly competent versions of its previous selves, based on the training history.

Throughout the training process, the board size was adjusted according to the following criteria:

- The board started at 3x3 and was increased by one dimension after every 250 games.
- For instance, after 250 games on a 3x3 board, the board size was increased to 4x4, and this process continued until the board size reached 7x7.

During each game, the agent made moves based on its current policy, and the training was performed using the PPO algorithm. The agent's performance was continually evaluated, and improvements were made by learning from the collected experiences.

This curriculum learning approach allowed the agent to progressively adapt to more complex game environments, thereby enhancing its overall learning and performance in playing the Hex game on a 7x7 board.

*5) **Experiment 5**:* In this experiment, we utilized two identically initialized agents, referred to as Agent1 and Agent2, which were trained by competing against each other.Self-play in training two agents for the Hex game fosters balanced skill development and adaptability by ensuring that each agent continuously adjusts and refines its strategies against

an equally capable opponent. This method promotes robust and generalizable learning, as evidenced by its successful application with Alpha Go Zero [5].

A total of 12,000 games were simulated to facilitate the training process. The learning dynamics were structured such that every 100th episode, the agents alternated in terms of which one was actively learning. Additionally, every 50th episode, the agents switched sides on the game board.

We employed all reward shaping methods except the second one. The rewards were cumulative and adjusted based on the agents' actions and the outcomes of the games. To discourage ineffective strategies, a penalty of 0.75 points was applied for ignoring blocking opportunities or failing to connect their own stones. Positive reinforcement was structured as follows: 3 points for blocking an opponent's move, 2.5 points for connecting their own stones, and 2.5 points for preventing two opponent pieces from connecting. Furthermore, agents were awarded 3 points for recognizing a winning move and 10 points for achieving a win. Conversely, a penalty of 10 points was imposed for losing a game.

To mitigate the deterministic behavior often associated with the PPO algorithm, an entropy bonus was incorporated. This entropy bonus encourages exploration by promoting a more diverse range of actions, thereby preventing the agent from becoming too predictable and potentially improving its overall strategic capabilities.

*6) **Experiment 6**:* In experiment 6 we trained an agent in phases. In the first phase he faced a random opponent and in every consecutive phase he played against the best agent of the last phase. Every 4th phase we let the agent play against a random opponent to not lose the ability to win against random opponents. Afterwards we let each trained agent compete against every other agent 20 times to determine our best agent. 20 reward points were given for winning, a modulation for winning in fewer moves was also given:

$$\text{Win Reward} = 20 + 5 \cdot \left(\frac{7}{\text{moves}}\right)^2$$

connecting moves received 2 reward points and blocking moves 1. Losing resets the total reward and the agent receives -30 reward points.

Every phase consisted of 1000 games and we ran 12 phases, so we received 12 agents. In theory we expected the last one to be the best agent, because he saw the most training

III. Results

A. General Evaluation

We frequently played against the agents manually. Unfortunately, none of the agents could block a straight line or block in general effectively. Against random opponents, it was evident that they also did not learn to place the winning stone when they had the opportunity. The agents often started the same way, and although the moves sometimes appeared random, they were deliberate. The agents performed worse as black compared to white.

B. Experiment 1

The agent quickly converged against the random opponent (see figure 1), achieving a win rate of approximately 87% (black) and 91% (white). It learned to draw a line to the other side to win against random opponents. Adding the reward for fewer moves led to even faster convergence. However, this approach was limited to learning for only one side. When the agent for white were switched to play black, the agent made incorrect moves, playing as if it were the opposite color. In addition, it could be seen during training that the agent for white often won in 7 moves, while for black it was more like 9.

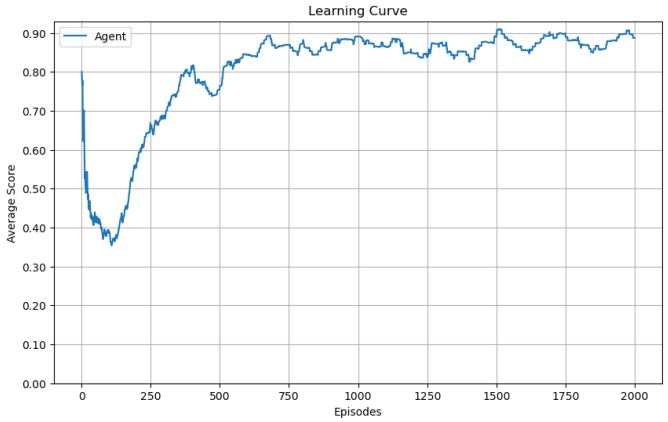


Fig. 1: Learning curve of PPO Agent as white in Experiment 1

C. Experiment 2

In the second experiment, the win rate was no longer as high, around 80%. With the trainnig plot you can now also recognize more spikes (see Figure 2). Nevertheless, you can see a pattern in the agent for both sides, even if the line no longer consists of 7 moves but rather 9 moves. But as in the previous experiment, the agent does not pay attention to the opponent.

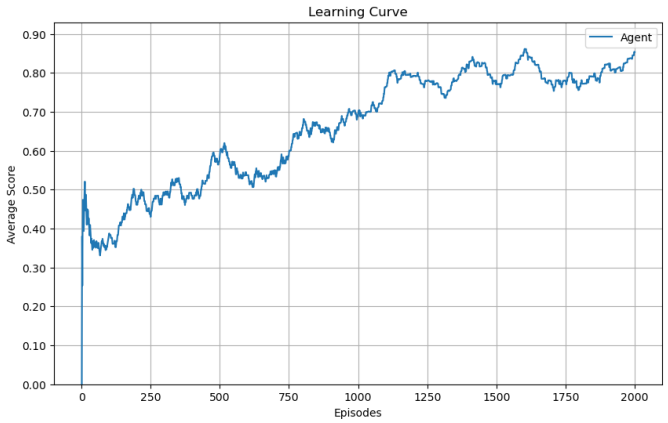


Fig. 2: Learning curve of the PPO Agent in Experiment 2

D. Experiment 3

The agent was engaged in a total of 3000 games against a random opponent, which was a crucial part of its learning curve. Initially, as the white player, the agent showed significant progress and adaptation; however, upon switching to the black side, a notable performance drop was observed. This phase was critical as it highlighted the agent's initial difficulty in adapting to the change in perspective and strategy, necessitating a period of re-convergence.

After this initial training against a random opponent, the agent underwent rounds of competitions against ten of its previous models. This part of the training process was particularly tumultuous, marked by significant fluctuations in performance. The agent's ability seemed to deteriorate when faced with the refined strategies of its predecessors, challenging its adaptability and learning mechanisms.

Continuous evaluation of the agent's performance was crucial. It was not merely about tracking wins and losses but also about understanding how quickly and efficiently the agent could achieve victories, which directly influenced the strategic depth and efficiency of its gameplay.

This color-neutral training approach was meticulously designed to develop an agent capable of playing proficiently as both the white and black player. The aim was not only to enhance its overall adaptability but also to deepen its strategic understanding of the Hex game. Through this rigorous training and evaluation process, we hoped to equip the agent with the versatility needed to handle and excel in varied gameplay situations, thereby maximizing its competitive edge in real-world scenarios.

E. Experiment 4

Initially, the agent demonstrated significant improvement:

- Against random opponents, over the course of 3000 games, the trained agent's performance converged to a win rate exceeding 90

However, when pitted against a total of 10 of its prior models:

- The performance of the agent deteriorated, suggesting challenges in adapting strategies against more strategically advanced versions of itself, leading to a decline in win rates.

This highlights the complexities and potential pitfalls of training against a progressively strengthening sequence of previous models, where the agent might not necessarily continue to improve without tailored adjustments in training strategies.

F. Experiment 5

Unfortunately neither of the two agents converged (see Figure 3). Both agents exhibit similar performance levels, with their scores oscillating between 2.5 and 7.5 points. During the training, the win rate of the two agents ran towards 50%, which is clear as they both had the same parameters and strategies. Also it was seen that they no longer win in just a few moves, but needed at least 15. The total timesteps recorded were 494,191. After training, the agents were evaluated by playing 20 games against random opponents, with win rates often between 50% and 70%.

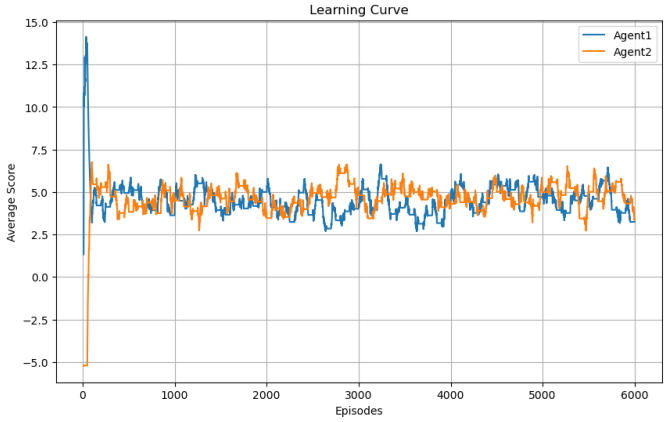
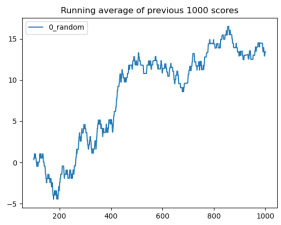


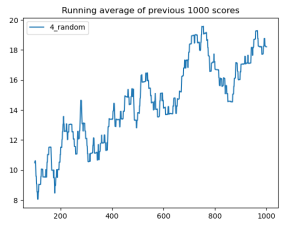
Fig. 3: Learning curve of the two PPO Agents in Experiment 5

G. Experiment 6

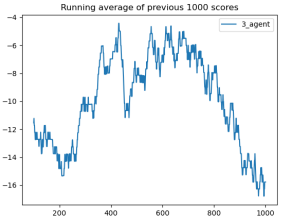
The different phase showed very different results in score history, as shown in fig. 4. In fig. 4a and fig. 4b we see training against random opponents and in fig. 4c and fig. 4d the learning curve against two of our trained agents is shown. While the scores in the cases against random opponents look very similar and also show convergence the training against the other agents seems quite random and it does not show any convergence.



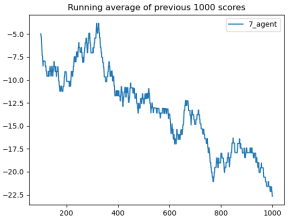
(a) Learning curve of the PPO Agent against Random Opponent Phase 0



(b) Learning curve of the PPO Agent against Random Opponent Phase 4



(c) Learning curve of the PPO Agent in Phase 3



(d) Learning curve of the PPO Agent in Phase 7

Fig. 4: Learning curves of the PPO Agents in different Phases

The final test result of the 20 games as biginning player and 20 games as 2nd player between all agents are shown in fig. 5, the color corresponds to the win rate of the agent in the row starting against the agent in the column. We can see that in general the agents seem to be not very different to each other only one significant difference between agent 10 and 6 can be observed. The best overall agent was agent

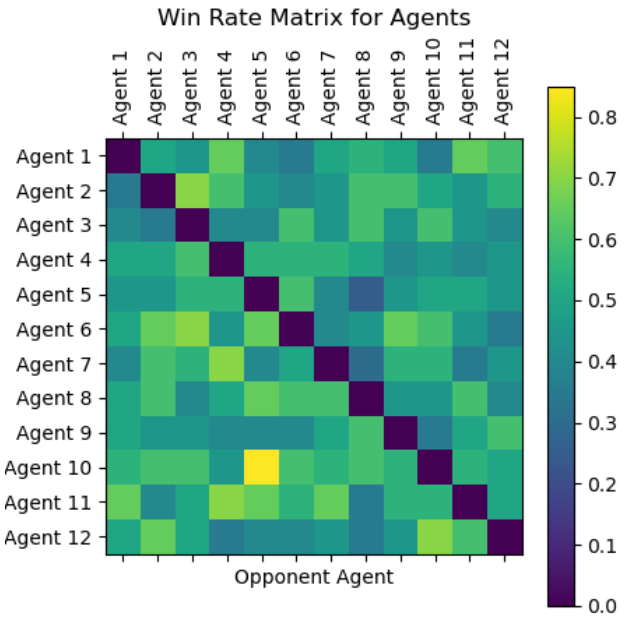


Fig. 5: Win Rates of Agents (row) against Agents (column)

The overall win rates can be found in table I. Agent 10 performed the best with a 53 percent win rate, while agent 5 performed the worst with only 46.5 percent win rate. In total every agent played 40 games, this

Statistical Significance: To determine if we found statistical significant result we determine the likelihood that the observed win rates are due to randomness, we conduct a chi-square goodness-of-fit test. This test helps us assess whether the distribution of win rates differs significantly from what we would expect under the null hypothesis of randomness.

Given that each agent plays 40 games and the observed win rates, we assume that the expected win rate under randomness is 0.5 for both roles. The chi-square statistic is calculated using the formula:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where O_i is the observed win rate and E_i is the expected win rate (0.5).

Calculations: - **Expected win rate** under the null hypothesis (randomness): 0.5 - **Number of agents**: 12 - **Degrees of freedom**: df $= 12 - 1 = 11$

The calculated chi-square statistic is approximately 0.0096. The p-value associated with this chi-square statistic and 11 degrees of freedom is extremely close to 1 (approximately 1).

Interpretation: - **Chi-square statistic**: This value is very low, indicating that the observed win rates are very close to the expected win rate of 0.5. - **P-value**: The extremely high p-value suggests that there is a very high probability that the observed results are due to randomness.

Conclusion: Given the high p-value, we fail to reject the null hypothesis. This means that there is no significant evidence to suggest that the observed win rates are different from what we would expect if the results were purely random. Therefore, it is very likely that these results are due to randomness.

TABLE I: Mean Win Rates for Each Agent

Agent	Mean Win Rate
1	0.510
2	0.495
3	0.465
4	0.485
5	0.475
6	0.515
7	0.495
8	0.525
9	0.480
10	0.530
11	0.520
12	0.505

IV. Discussion

In Experiment 1, the agent achieved high win rates against random opponents (87% as black, 91% as white) but struggled when switching sides, revealing limited adaptability. Reward shaping for fewer moves accelerated convergence but hindered effective opponent blocking.

In Experiment 2, alternating roles every 200 games resulted in a slightly lower win rate (around 80%). The agent developed strategies for both sides but demonstrated limited ability to counter the opponent's moves, indicating a lack of strategic depth.

In Experiment 3, the agent trained for 3000 games, switching roles midway. Initial progress as white was notable, but a significant drop was observed upon switching to black. Training against previous models showed fluctuating performance, highlighting difficulties in adapting strategies and maintaining proficiency against stronger opponents.

In Experiment 4, the agent underwent progressive training on increasing board sizes. Initially, the agent showed significant improvement against random opponents, with a win rate exceeding 90%. However, its performance declined when facing stronger previous models, indicating challenges in adapting to more complex strategies without specific training adjustments.

In Experiment 5, neither agent converged, with performance oscillating between 2.5 and 7.5 points. Post-training evaluation against random opponents showed win rates between 50% and 70%. This means that developing distinct strategies failed. Which suggests that identical initialization and parameters led to mirrored learning, limiting strategic diversity.

In Experiment 6, agents were trained in phases against various opponents. The results showed inconsistent scores, with

convergence only when training against random opponents. Training against other agents did not show clear convergence, indicating challenges in forming effective strategies. The win rate analysis revealed only minor differences among agents, which were not statistically significant. However, it is noteworthy that Agent 1, which trained exclusively against a random opponent and achieved high win rates against it, performed similarly to other agents. This indicates that the agents play is not random, but the strategies they learned are not particularly effective.

V. Conclusion

This paper explored training Proximal Policy Optimization (PPO) agents for the Hex game through six experiments. The results revealed that while agents quickly achieved high win rates against random opponents, they struggled with complex strategies and role switching, highlighting limitations in adaptability and strategic depth.

Self-play in Experiment 5 showed limited success due to mirrored learning. Experiment 6's phased training showed potential but inconsistent results, with convergence primarily against random opponents. Statistical analysis indicated no significant differences among agents, suggesting the results were likely due to randomness.

Future work should focus on simplified reward structures, extended training durations beyond 12,000 games, periodic evaluations against various opponents, and advanced techniques like curriculum learning and opponent modeling to enhance strategic depth and adaptability. By refining these methodologies, we can develop more competent and versatile PPO agents for strategic games like Hex.

References

[1] Schulman, John, et al. "Proximal Policy Optimization Algorithms." arXiv preprint arXiv:1707.06347, 2017. Available: https://arxiv.org/abs/1707.06347.

[2] Tabor, Phil. "Proximal Policy Optimization (PPO) Implementation in PyTorch." GitHub, 2020. Available: https://github.com/philtabor/Youtube-Code-Repository/blob/master/ReinforcementLearning/PolicyGradient /PPO/torch/ppo$_torch.py.Accessed$: $2024-06-03$.

[3] Jiang, Zhouxi, Yang, Jianfeng, and Gao, Xun. "Minimizing Task Age upon Decision for Low-Latency MEC Networks Task Offloading with Action-Masked Deep Reinforcement Learning." Sensors 2024, 24(9), 2812. Available: https://doi.org/10.3390/s24092812 . Submission received: 14 March 2024 / Revised: 18 April 2024 / Accepted: 26 April 2024 / Published: 28 April 2024.

[4] AI Blog Tech. "Top 8 Reward Shaping Techniques in Reinforcement Learning." Medium, 2020. Available: https://medium.com/@aiblogtech/top-8-reward-shaping-techniques-in-a-reinforcement-learning-a0721d59deda. Accessed: 2024-06-15.

[5] Silver, David, et al. "Mastering the game of Go without human knowledge." *Nature* 550.7676 (2017): 354-359.