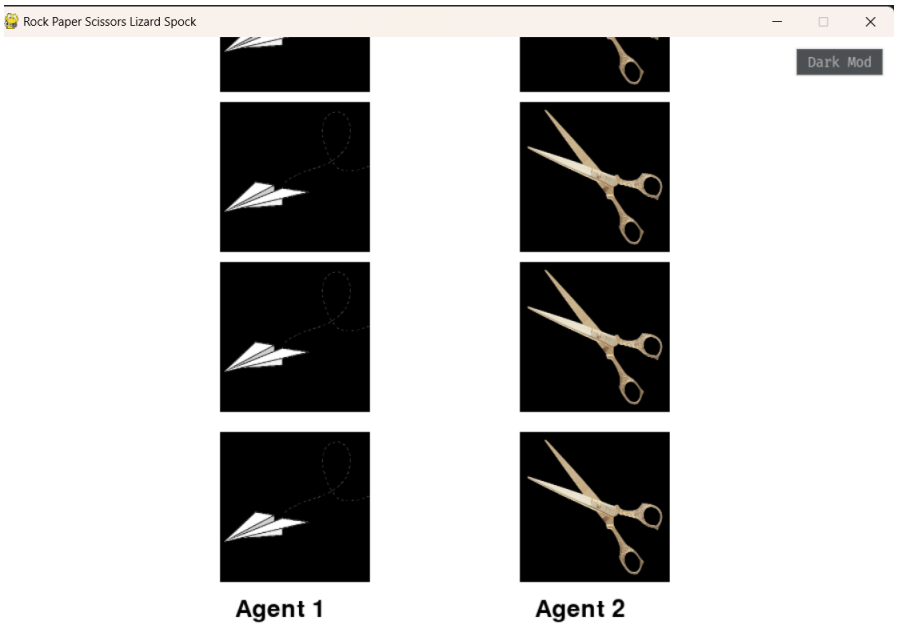
DS251-Artificial intelligence

Multi-Agent Rock-Paper-Scissor-Lizard-Spock Game



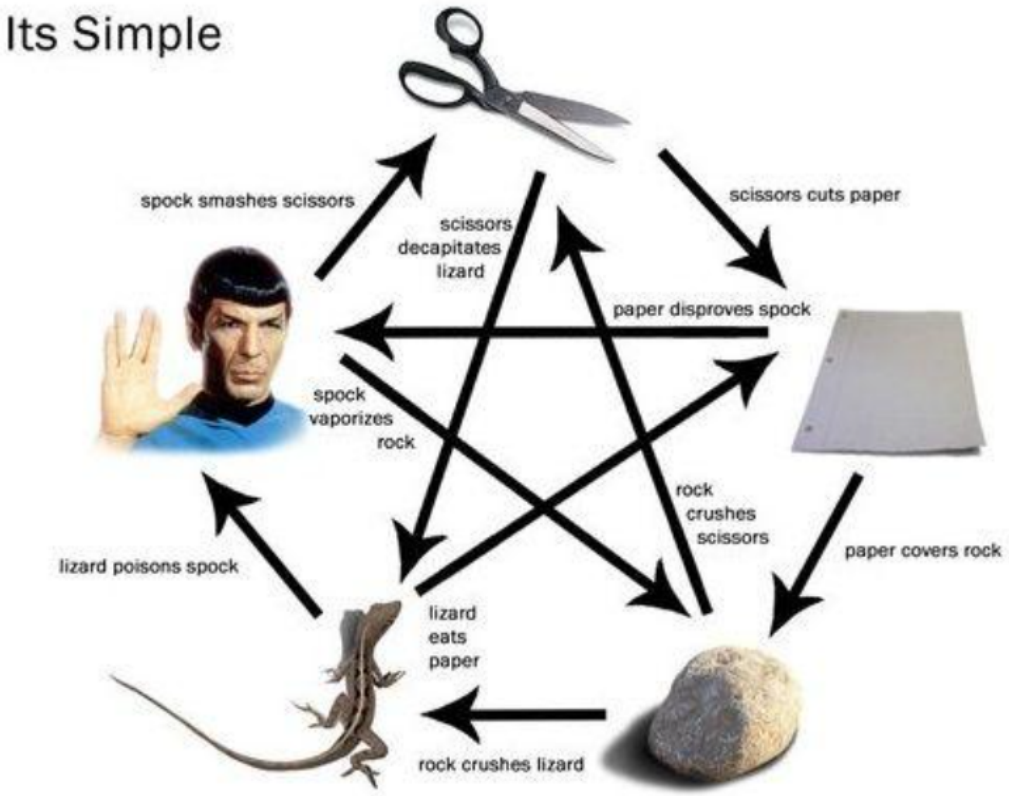
1. Introduction

This project ventures into the dynamic intersection of artificial intelligence and reinforcement learning, introducing a compelling multi-agent environment inspired by the classic Rock-Paper-Scissors-Lizard-Spock game. With the aim of navigating the intricacies of strategic decision-making, this environment employs OpenAI Gym and Pygame to create a visually immersive interface. Agents interact over predefined cycles, engaging in a learning process facilitated by a Q-learning-based reinforcement model. The graphical user interface provides real-time insights into agent actions and historical gameplay, fostering a holistic understanding of the evolving strategies. The ensuing report delves into the technical nuances of game logic, GUI design, Q-learning algorithms, and performance metrics, offering a comprehensive exploration of an intelligent and adaptive system within the captivating context of Rock-Paper-Scissors-Lizard-Spock.

2. Multi-Agent Rock-Paper-Scissors-Lizard-Spock Environment

2.1. Game Overview

The Rock-Paper-Scissors-Lizard-Spock game extends the traditional Rock-Paper-Scissors game with two additional actions, creating a more intricate set of rules. The game involves two agents, each choosing one action from five possible options: Rock, Paper, Scissors, Lizard, and Spock. The winner of each round is determined by specific rules that govern the interactions between these actions.



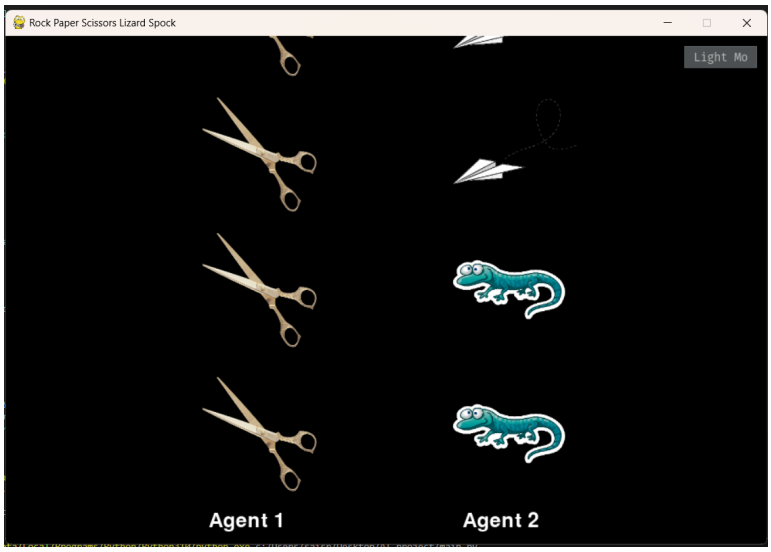
2.2. Environment Architecture

The multi-agent environment is implemented using the OpenAI Gym framework, providing a flexible and standardized interface for reinforcement learning tasks. The action space is discrete, corresponding to the five available actions, and the observation space is also

discrete, representing the current state. Pygame is employed for graphical representation, allowing for an interactive and visually appealing display of the game.

2.3. Graphical User Interface (GUI)

The Pygame interface includes images for each action (Rock, Paper, Scissors, Lizard, and Spock), enhancing the visual experience. The environment supports a toggleable dark mode, catering to user preferences. Agent actions are displayed in real-time, and a history of actions is presented in a graphical format, contributing to a comprehensive understanding of the game dynamics.The **render** function dynamically updates the display to show agent actions, history, and other relevant information.



Dark Mode

2.4. Game Dynamics

The game progresses through a predefined number of cycles, each cycle representing a round of interactions between the agents. The **step** function processes agent actions, determines the winner based on predefined rules, and updates the game state accordingly. Rewards are assigned to agents based on the outcome of each round, with positive rewards for winning and negative rewards for losing. The environment is designed to promote strategic thinking and learning over multiple cycles.

3. Reinforcement Learning Model

3.1. Q-Learning Framework

The reinforcement learning model is implemented using Q-learning, a model-free and off-policy algorithm. Q-values are maintained for each combination of agent and action, representing the expected cumulative reward for taking a specific action in a particular state. The exploration-exploitation trade-off is addressed through an epsilon-greedy strategy, allowing agents to explore new actions with a certain probability (**epsilon**).

3.2. Training Process

The model undergoes training over a specified number of episodes, with each episode comprising multiple cycles of interactions. During training, agents select actions based on their learned Q-values, and these values are updated using the observed rewards, the discount factor, and the maximum Q-value for the next state.The Q-values are updated using the Bellman equation, guiding the model toward optimal strategies. The epsilon-greedy strategy introduces a controlled level of exploration to ensure a balance between exploration and exploitation. The training process is iterative, allowing agents to adapt and improve their strategies over time.

4. Results and Analysis

4.1. Performance Metrics

The model's performance is evaluated by tracking the average reward over episodes. This metric provides insights into the learning progress of the agents and the effectiveness of the Q-learning algorithm.This metric is calculated with both positive and negative rewards(1 or -1) and only positive rewards(1 or 0).

```
PS C:\Users\saisr> & C:/Users/saisr/AppData/Local/Programs/Python/Python310/python.exe c:/Users/saisr/Desktop/AI_project/main.py
pygame-ce 2.3.2 (SDL 2.26.5, Python 3.10.4)
Episodes 1-50: Average Reward -2.28
Episodes 51-100: Average Reward -1.2
Episodes 101-150: Average Reward -1.0
Episodes 151-200: Average Reward -0.76
Episodes 201-250: Average Reward -1.48
Episodes 251-300: Average Reward -1.12
Episodes 301-350: Average Reward -0.8
Episodes 351-400: Average Reward -1.4
Episodes 401-450: Average Reward -1.24
Episodes 451-500: Average Reward -1.12
Episodes 501-550: Average Reward -1.44
Episodes 551-600: Average Reward -1.12
Episodes 601-650: Average Reward -1.52
Episodes 651-700: Average Reward -1.6
Episodes 701-750: Average Reward -1.32
Episodes 751-800: Average Reward -0.88
Episodes 801-850: Average Reward -1.12
Episodes 851-900: Average Reward -1.48
Episodes 901-950: Average Reward -0.76
Episodes 951-1000: Average Reward -1.28
```

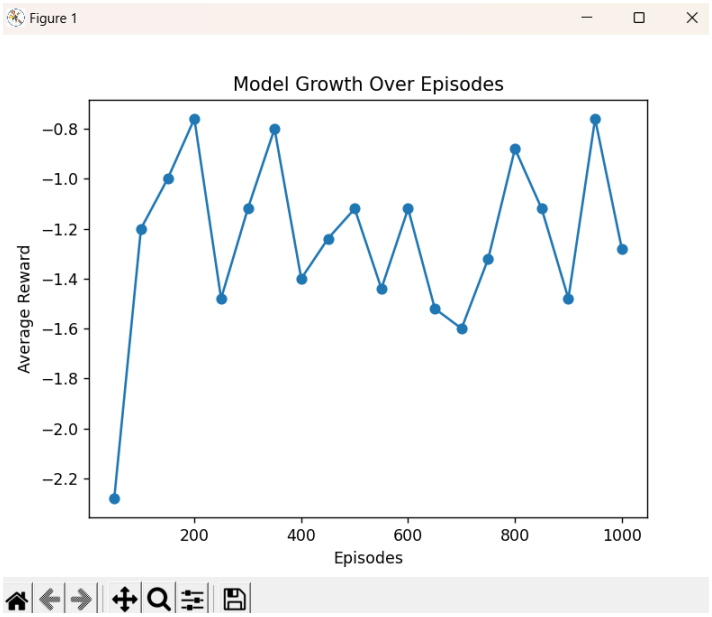
Positive and Negative rewards(1 or -1)

```
PS C:\Users\saisr> & C:/Users/saisr/AppData/Local/Programs/Python/Python310/python.exe c:/Users/saisr/Desktop/AI_project/main.py
pygame-ce 2.3.2 (SDL 2.26.5, Python 3.10.4)
Episodes 1-50: Average Reward 9.18
Episodes 51-100: Average Reward 9.5
Episodes 101-150: Average Reward 9.62
Episodes 151-200: Average Reward 9.26
Episodes 201-250: Average Reward 9.44
Episodes 251-300: Average Reward 9.44
Episodes 301-350: Average Reward 9.44
Episodes 351-400: Average Reward 9.42
Episodes 401-450: Average Reward 9.58
Episodes 451-500: Average Reward 9.34
Episodes 501-550: Average Reward 9.4
Episodes 551-600: Average Reward 9.34
Episodes 601-650: Average Reward 9.48
Episodes 651-700: Average Reward 9.6
Episodes 701-750: Average Reward 9.3
Episodes 751-800: Average Reward 9.36
Episodes 801-850: Average Reward 9.52
Episodes 851-900: Average Reward 9.68
Episodes 901-950: Average Reward 9.34
Episodes 951-1000: Average Reward 9.4
```

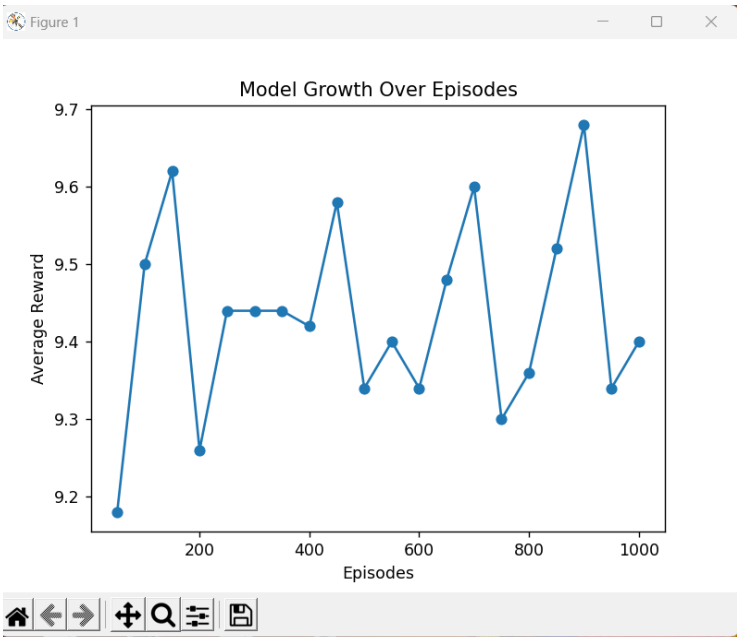
Only Positive rewards(1 or 0)

4.2. Visualization

A graphical representation of the average reward over episodes is presented through a plot.The x-axis represents the episode number, and the y-axis represents the average reward. This visualization allows for a clear understanding of the model's growth and learning trends. Patterns and fluctuations in the average reward provide valuable insights into the training dynamics. The graphs are also taken for both positive and negative rewards(1 or -1) and only positive rewards(1 or 0).



Positive and Negative rewards(1 or -1)



Only Positive rewards(1 or 0)

4.3. Observations

An analysis of the training results involves examining how well the agents adapt to the game dynamics. It includes an exploration of whether agents successfully exploit optimal strategies and if the Q-learning algorithm converges to a stable solution. Patterns and fluctuations in the average reward are scrutinized to derive insights into the training dynamics

5. Conclusion

The multi-agent Rock-Paper-Scissors-Lizard-Spock environment and its associated reinforcement learning model provide a robust platform for exploring complex decision-making in interactive scenarios. The combination of graphical representation, game dynamics, and Q-learning yields a versatile framework for further research and experimentation in the domain of multi-agent reinforcement learning. The toggleable dark mode and interactive Pygame interface contribute to a user-friendly and engaging experience. Overall, this project lays the groundwork for future investigations into advanced reinforcement learning techniques and multi-agent systems.

Group24:

-Raghavapurapu Chiranjeevi Srinivas-12141290

-M Jashraj-12141110

-D Sai Srihan-12140620

-B Vamshi Krishna-12140410