**“Designing an Intelligent Agent for Ludo Game using Reinforcement Learning**”

1. **Introduction**

The Ludo game is a strategic multiplayer board game that challenges players to move all of their pieces to the final goal. This project aims to develop a smart game-playing agent using Reinforcement Learning, particularly the Q-learning and Deep Q-learning algorithms, to learn and play Ludo intelligently.

1. **Problem Definition**

Traditional game-playing agents often use fixed strategies, which are not adaptable to different scenarios. This project proposes using model-free RL approaches (Q-learning and Deep Q-learning) to allow agents to learn from interactions with the environment and improve over time.

**3. Methodology**

* **Environment Design:**  
  A simplified Ludo environment was designed where each player has 4 pieces and the goal is to reach position 10. The environment includes basic rules such as dice rolls, piece movement, safe spots, and opponent attacks.
* **Agent Design:**  
  Two intelligent agents were implemented:
  + A **Q-learning agent** using a Q-table to store state-action values.
  + A **Deep Q-learning agent** using a neural network to approximate Q-values.
* **Training:**  
  Both agents were trained over thousands of episodes. The Q-learning agent updated its Q-table, while the DQN agent used experience replay and a neural network to learn optimal strategies.
* **Evaluation:**  
  The agents were evaluated based on cumulative reward, convergence speed, and generalization ability under varying gameplay conditions.

**4. Key Components**

* **State:**  
  A tuple representing the position of each player's 4 pieces and the current player.
* **Actions:**  
  The set of valid moves for the current player — i.e., which piece to move.
* **Reward System:**  
  Rewards and penalties are given based on the outcome of each move. The reward system is defined as follows:

| **Event** | **Reward** | **Penalty to Opponent** | **Description** |
| --- | --- | --- | --- |
| Moving a piece (normal move) | 0 | 0 | No special outcome |
| Landing on a safe spot (e.g., 5, 15...) | +5 | 0 | Safe zones offer a small reward |
| Reaching the goal (position 10) | +20 | 0 | Reaching the target gives a high reward |
| Hitting an opponent's piece | +20 | -5 | Reward for knocking an opponent’s piece |
| Winning the game (all pieces reach goal) | +50 | 0 | Bonus reward for completing the game |

* **Q-table / DQN:**
  + Q-learning agent uses a lookup table.
  + DQN agent uses deep neural networks to estimate Q-values.

**5. Results**

* Both agents improved their performance progressively with training.
* The Q-learning agent showed faster convergence for smaller state spaces.
* The Deep Q-learning agent generalized better for larger or more complex scenarios.
* Both agents demonstrated consistent performance across different gameplay conditions.A screen shot of a graph

  AI-generated content may be incorrect.A graph of a graph

  AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

**6. Conclusion**

Reinforcement Learning, particularly Q-learning and Deep Q-learning, proved effective in training intelligent agents for the Ludo game. While Q-learning performs well in small environments, Deep Q-learning offers scalability and adaptability. With future enhancements like multi-agent learning or curriculum learning, these agents can achieve even more sophisticated gameplay.