**CS 370 Project 2 Report**

This project involved creating and training a pirate agent using deep Q-learning to navigate a maze and find a treasure. The maze is an 8x8 grid with free and blocked cells, and the treasure is in the bottom-right corner. The pirate learns the best path to the treasure through its experiences.

**Goals of the Project**

1. Create a deep Q-learning algorithm to help the pirate find the treasure.
2. Train the pirate to navigate the maze from different starting positions.
3. Test how well the model works based on training and gameplay results.

**Maze Setup**

The maze is represented as a grid where:

* **1.0** means the pirate can move through the cell (free).
* **0.0** means the cell is blocked.
* The treasure is always in the bottom-right corner.

The TreasureMaze class manages the maze, tracks where the pirate moves, and gives rewards based on the pirate’s actions.

**Experience Replay**

The GameExperience class saves the pirate’s previous moves (episodes) in memory. These episodes are used during training to help the pirate learn from past experiences. Randomly picking episodes from memory ensures training is steady and efficient.

**Neural Network**

The neural network decides the best move for the pirate in any situation:

* **Input Layer:** Processes the maze state as a flattened array.
* **Hidden Layers:** Two layers with PReLU activation for better decision-making.
* **Output Layer:** Provides Q-values for the four possible moves: LEFT, RIGHT, UP, and DOWN.

The model uses Mean Squared Error (MSE) as the loss function and Adam as the optimizer to adjust weights during training.

**Training Process**

The qtrain function handles the training process:

1. The maze is initialized, and the pirate starts at a random position.
2. Over several rounds (epochs), the pirate:
   * Chooses actions randomly (exploration) or based on learned strategies (exploitation).
   * Updates its position, calculates rewards, and saves the episode in memory.
   * Uses the saved episodes to train the neural network.
3. Training stops when the pirate wins 100% of the time or when the maximum number of epochs is reached.

**Testing the Model**

**Completion Check:**  
This test ensures the pirate can find the treasure starting from any free cell in the maze. Passing this test means the pirate has learned how to navigate the maze effectively.

**Gameplay Simulation:**  
The pirate starts at the top-left corner and tries to reach the treasure in the bottom-right corner. The gameplay shows the pirate’s path and its interactions with the maze.

**Results**

**Training Results:**  
During training, the pirate improved as loss decreased and the win rate increased. Example training log:  
*Epoch: 003/14999 | Loss: 0.0026 | Episodes: 18 | Win count: 1 | Win rate: 0.250 | Time: 47.6 seconds*

**Completion Check Results:**  
The pirate passed the completion check, showing it could navigate to the treasure from any free cell.

**Gameplay Results:**  
The pirate successfully navigated the maze from the top-left corner to the treasure, avoiding obstacles along the way.

**Challenges**

Adjusting the exploration rate and learning rate was difficult, as these settings affected how quickly and effectively the pirate learned. Preventing the pirate from revisiting cells and getting stuck was another key challenge.

**References**

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction (2nd ed.).* MIT Press.

Surma, G. (2018). *Cartpole: Introduction to Reinforcement Learning.* GitHub Repository.

Mnih, V., et al. (2015). *Human-level control through deep reinforcement learning.* *Nature,* 518(7540), 529-533.