**Author:** Ryan Hatch  
**Course:** CS-370**:**  Treasure Hunt Game Using Deep Q-Learning  
**Date:** 12/12/24  
**Assignment:** Development of the Design Defense

**Design Defense: Treasure Hunt Game Using Deep Q-Learning:**

**Introduction:**

The goal of this project is to create an intelligent agent (the “pirate”) that can navigate a maze and reach the treasure using a Deep Q-Learning (DQL) algorithm. By training the agent to maximize rewards and minimize penalties, we enable it to discover an optimal path to the treasure without human intervention. This design defense will explain the approach taken, compare human and machine strategies, and evaluate the effectiveness of deep Q-learning in solving this pathfinding problem.

**Human vs. Machine Problem-Solving:**

* **Human Approach to Solving Mazes:**
* **Observation and Pattern Recognition:** A human would first look at the maze, identify walls, free spaces, and the goal location.
* **Planning a Route:** Humans typically plan or visualize multiple routes mentally, using intuition and experience to prune clearly bad paths (e.g., dead ends).
* **Trial and Error (Heuristics):** A human might attempt a route, backtrack if it fails, and refine their approach.
* **Adaptation:** Based on successes and failures, humans adjust their routes until they reach the goal or exhaust possibilities.
* **Machine (Agent) Approach to Solving Mazes**
* **Representation of State:** The agent observes the environment as a matrix of free, blocked, or goal cells.
* **Action Selection (Exploration vs. Exploitation):** At each step, the agent either selects an action randomly (exploration) or picks the highest-reward action learned so far (exploitation).
* **Reward-Based Learning:** After each move, the agent receives a reward (positive or negative). Over many episodes, it updates its Q-values to favor high-reward sequences.
* **Convergence to Optimal Policy:** As training progresses, the agent’s policy converges to an optimal strategy that efficiently navigates the maze.
* **Similarities and Differences:**
  + **Similarities:**
    - In terms of Similarities, both humans and the agent aim to minimize “costly” or inefficient moves and maximize the chance of successfully reaching the goal.
  + **Differences:**
    - **Search Method:** Humans rely on intuition and domain-specific heuristics; the agent relies on systematic exploration and learned Q-values.
    - **Memory and Precision:** Humans can forget or overlook paths, whereas the agent consistently accumulates experience in replay memory.
    - **Scale and Consistency:** Machine approaches can handle large or complex mazes for many epochs without fatigue, while humans become error-prone over time.

**Purpose of the Intelligent Agent in Pathfinding**

The primary purpose of the agent is to autonomously discover and use an **optimal pathfinding strategy**:

* **Autonomy:** The agent should require minimal human guidance once the environment is defined.
* **Adaptability:** By balancing exploration, or discovering new paths, along with exploitation, or using known high-value paths, the agent can adapt to various maze layouts.
* **Efficiency:** The final policy should minimize penalties and quickly find the treasure.

**Exploitation vs. Exploration:**

* **Exploitation:** The agent chooses the move it currently believes is best, based on learned Q-values.
* **Exploration:** The agent tries less-familiar moves to discover potentially better rewards.

For an ideal proportion of both exploitation and exploration**,** it’s also good to note how early in training using a higher exploration, for example 0.1 or 0.2, helps gather broad experience. As training progresses, the exploration can be reduced, for example an optimal number would be ~0.05 to exploit knowledge at an optimal rate. This makes sure that the agent does not get stuck in a inferior or deficient policy.

**Reinforcement Learning for the Pirate:**

Reinforcement learning implementation directly tie the agent’s decisions to its outcomes:

* **Immediate Feedback:** Each move yields a small penalty or reward which helps in shaping the agent’s behavior.
* **Delayed Gratification:** Larger positive rewards only come at the goal, encouraging the agent to reach the treasure quickly and avoid random wandering.
* **Cumulative Reward Maximization:** Over many episodes, the agent refines its strategy to maximize total rewards, consistently leading to the most efficient path.

**Evaluating the Use of Algorithms to Solve Complex Problems by Implementing Deep Q-Learning:**

* **Neural Network Architecture:**
  + **Input Layer:** A flattened representation of the maze, or the environment state.
  + **Hidden Layers:** Dense layers, for example with PReLU or ReLU activations, in order to capture complex relationships between positions and actions.
  + **Output Layer:** Q-values for each possible action, via up, down, left, and right.
* **Key Algorithms and Techniques**
  + **Experience Replay:** Stores transitions into a tuple (s,a,r,s′,done). Random sampling from this memory breaks correlations between consecutive states and stabilizes learning.
  + **Target Network:** In some advanced variants, a secondary network is used to provide stable targets and reduce training cycles.
  + **Bellman Equation (Update Rule):**  
    Q(s,a)←reward+γ maxa′ Q(s′,a′) - This equation calculates the “true” expected return for the chosen action.
* **Hyperparameters:**
  + **Learning Rate:** Manages how quickly the network adapts to new data.
  + **Discount Factor (γ):** Determines the importance of future rewards.
  + **Batch Size / Data Size:** The number of experiences sampled from the replay memory for each training update.
  + **Max Memory:** Limits the size of replay memory to avoid overfitting and reduce memory usage.
* **Why Deep Q-Learning is Effective**
* **Scalability:** Neural networks handle high-dimensional state spaces, for example larger mazes are better than tabular Q-learning.
* **Generalization:** A trained model can adapt to slightly modified mazes without starting from scratch.
* **Proven Success:** DQN variants are widely used in many reinforcement learning applications, from games like Atari to robotics.
* **Potential Limitations**
* **Long Training Times:** Large neural networks and many episodes can lead to extended training durations.
* **Parameter Sensitivity:** The agent’s performance depends heavily on choosing appropriate hyperparameters.
* **Exploration Challenges:** If ε is too small too soon, the agent may converge to a suboptimal path. If ε is too large for too long, training can become inefficient.

**Conclusion:**

This project utilizes deep Q-learning to train a pirate agent to navigate a maze and locate treasure. By balancing exploration and exploitation, the agent refines its strategy through repeated trials, eventually converging on the most efficient path. Unlike a human approach, the reinforcement learning agent systematically accumulates experience and quantifies the value of each action. While training can be time-intensive, deep Q-learning proves to be a solid solution for pathfinding tasks in complex environments showing the effective combination of neural networks and reinforcement learning.

**References**

* Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., et al. (2015). *Human-level control through deep reinforcement learning.* Nature, 518(7540), 529–533.
* Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
* TreasureHuntGame Project Resources (Provided code and environment specifications).