**Design Defense: Deep Q-Learning for Treasure Maze**

Marissa Lanza

CS 370 Module 7 — Project Two (7-3 Project Two)

August 13, 2025

**Design Defense: Deep Q-Learning for Treasure Maze**

**Introduction**

This project implements a deep Q-learning (DQN) (Mnih et al., 2015) agent to control a pirate NPC navigating an 8×8 Treasure Maze. The environment rewards efficient navigation (+1 for reaching the treasure) and penalizes poor moves such as hitting walls or attempting to move out of bounds, with a small step cost to discourage wandering. I completed the Q-training loop in the provided notebook (without modifying the .py files), trained a neural-network–based Q-function with experience replay and an ε-greedy policy, and verified success using the supplied play/visualize routine.

**Human vs. Machine Problem Solving**

**Human approach to the maze.** A human typically scans the grid, identifies obstacles and corridors, and forms a quick plan toward the goal. On encountering a block, they backtrack, remember dead ends, and update a mental map. With a few trials, humans use spatial reasoning to select short, low-risk paths.

**Agent approach to the maze.** The agent encodes each state as a 64‑element vector and chooses among four actions (up, down, left, right) via an ε‑greedy policy. Each transition (state, action, reward, next\_state, status) is stored in replay memory. A Keras Sequential network with PReLU activations estimates Q‑values for each action. The agent samples random mini‑batches from memory and updates the network using mean‑squared error loss and the Adam optimizer. Exploration (ε) decays from an initial value (≈0.20) toward a minimum (≈0.05) to shift gradually from discovery to exploitation. Training ends early once the policy achieves a perfect win rate over a rolling window of recent games.

**Similarities and differences.** Both humans and the agent learn from trial and error, avoid repeated mistakes, and converge on efficient paths. However, humans rely on abstraction and spatial reasoning and can generalize from a few episodes, while the agent depends strictly on reward signals and typically needs many experiences before its Q‑function approximates optimal behavior.

**Purpose of the intelligent agent in pathfinding**

The agent automates sequential decision‑making in environments where enumerating rules for every situation is infeasible. In pathfinding, an RL agent adapts to layout variations and transfers to related navigation tasks (e.g., games, robotics).

**Exploration vs. exploitation and ideal proportion**

Exploration tries uncertain actions to discover better routes; exploitation selects the current best‑estimated action. For this maze, a good schedule begins with ε≈0.20–0.30 to ensure coverage, decaying to ε≈0.05 once the agent repeatedly reaches the treasure. This schedule prevents premature convergence while enabling a stable final policy.

**How reinforcement learning determines the path to the goal**

Temporal‑difference learning propagates the value of reaching the treasure back through preceding states. Shaped rewards (goal bonus, penalties for invalid moves, and a small step cost) align the objective with efficient paths, so repeated interaction yields a policy that maximizes expected cumulative reward.

**Deep Q‑learning implementation details**

Model: a Sequential network with two hidden layers using PReLU activations and a linear 4‑unit output head (one Q‑value per action). Training: experience replay buffer, mini‑batches (e.g., 32–512), discount factor γ≈0.95, mean‑squared error loss, Adam optimizer, and ε‑greedy action selection limited to valid moves. Early stopping triggers when the rolling win rate reaches 100% over a set window, indicating a stable policy.

**Conclusion**

The design combines reinforcement learning and neural networks to solve a discrete pathfinding problem effectively. By balancing exploration and exploitation, leveraging replay memory, and using a compact network, the agent consistently reaches the treasure while adhering to the project’s constraints and best practices. The trained model achieved a 100% win rate over the rolling window and early-stopped upon convergence, confirming a stable policy.

**References**

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. Machine Learning, 8(3–4), 293–321. https://doi.org/10.1007/BF00992699

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236

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