**Project Two**

**Design Defense; Pirate Intelligent Agent**

Laura Malone

Southern New Hampshire University

CS370: Current/Emerging Trends in Computer Science

Mohammad Habibi

October 19th, 2025

**Project Two**

**Design Defense; Pirate Intelligent Agent**

**Introduction**

This project develops and defends an intelligent pirate agent that learns to navigate a maze to reach hidden treasure before the human player. Using deep Q-learning, the agent trains through reinforcement, balancing exploration of new routes with exploitation of successful paths to solve a classic pathfinding problem. The model operates within a grid-based environment where obstacles and limited vision require adaptive, experience-based decision making.

My own development process often mirrored the pirate’s wandering through the maze. Early in the project, I followed several “rabbit holes” of experimentation, testing small adjustments to training cadence, step limits, and reward feedback to guide the model toward convergence. Over time, these refinements, such as adding a maximum step limit to reduce unproductive wandering and diagnostic counters like ‘bump\_left’ and ‘saw\_left’, transformed random exploration into purposeful navigation. After just over 20 hours of training across fifteen hundred epochs, the agent achieved consistent treasure discovery and demonstrated a clear understanding of optimal routes.

This design defense explains the reasoning behind those choices, contrasts how humans and machines solve similar problems, and evaluates the use of reinforcement learning for adaptive pathfinding in games.

**Human vs. Machine Problem-Solving**

When faced with a maze or pathfinding challenge, a human naturally combines reasoning, pattern recognition, and memory to plan an efficient route. A player observing the maze from above would likely start by identifying the goal’s location, tracing mental routes, and ruling out dead ends before moving. Even without full visibility, humans tend to rely on heuristics such as “keep the wall on your left” or “move toward open paths” to simplify choices. Once an obstacle is encountered, a human adapts quickly remembering previous dead ends, adjusting strategy, and mentally mapping progress. In short, human problem-solving is guided by foresight, intuition, and the ability to generalize past experiences to new contexts.

By contrast, the pirate agent has no innate understanding of goals, obstacles, or strategy. Instead, it begins with a blank policy and learns purely from interaction rewarded for reaching the treasure, penalized for hitting walls, and ignored when wandering aimlessly. Its learning is incremental: over many episodes, it evaluates the value of each state–action pair through trial and error, gradually building an internal map of which actions yield the highest expected rewards. Whereas a human uses planning and visualization, the machine relies on statistical approximation through deep Q-learning, adjusting neural network weights to estimate long-term value for every possible move.

The key similarity between the two lies in adaptation. Both start uncertain and improve through feedback humans through reflection and memory, the agent through rewards and loss minimization. The primary difference is in cognition versus computation: humans can infer structure and intention after only a few attempts, while the agent must experience thousands of episodes before forming the equivalent understanding numerically. Despite this, reinforcement learning allows the agent to achieve similar mastery over time, discovering paths that a human might find through reasoning but in a purely data-driven way.

As the pirate agent’s training progressed, its approach began to resemble purposeful reasoning rather than random trial. This shift from blind exploration to focused navigation highlights the central purpose of the intelligent agent in reinforcement learning: to discover optimal actions through experience rather than explicit instruction. Understanding how the agent balances exploration of new possibilities with exploitation of known strategies is critical to shaping its learning efficiency and ultimate success in pathfinding.

**Purpose of the Intelligent Agent and the Exploration–Exploitation Balance**

The purpose of the intelligent agent in this project is to simulate adaptive decision-making within a dynamic environment. The pirate must learn not only to move but to plan indirectly navigating toward a goal it cannot initially “see” and avoiding obstacles that impede progress. Unlike traditional path-finding algorithms such as A\*, which use predefined heuristics to compute an optimal path, the deep Q-learning pirate agent develops its understanding through reinforcement. Its behavior emerges from experience rather than instruction, enabling it to adapt to environments that may change or contain uncertainty.

During early training, the agent’s actions were highly exploratory it would move aimlessly, collide with walls, or loop through inefficient routes. Over time, as rewards accumulated and penalties reinforced avoidance of obstacles, its policy stabilized. This learning process required a balance between exploration (trying new paths) and exploitation (choosing the best-known route). To encourage both behaviors, the model used an epsilon-greedy policy, where the value of epsilon determined the probability of taking a random action. High epsilon values at the start allowed the agent to discover varied states of the maze, while gradual decay promoted exploitation as it began to identify reliable patterns.

Finding the ideal proportion between exploration and exploitation required iterative tuning. Too much exploration caused inefficient wandering and wasted training time, while too much exploitation risked the agent getting “stuck” repeating suboptimal paths. I found that maintaining a small but non-zero epsilon value late in training (around 0.28–0.30) kept the model adaptive even as it neared convergence. This balance enabled the pirate to avoid stagnation, recover from local traps, and consistently locate the treasure. Reinforcement learning thus allowed the agent to transform from a random wanderer into a capable problem solver, mirroring human adaptability through numerical optimization rather than intuition.

**Results and Validation**

To evaluate the pirate agent, I relied on quantitative training signals and targeted rollouts rather than the default notebook screenshot, which intermittently fails to render a path even when the agent has learned a viable policy. Across training, the agent’s wins increased steadily while loss decreased:

* Early training (≈ epoch 260): win rate stabilized around 0.19–0.22 with occasional short, high-reward trajectories.
* Mid training (≈ epochs 900–1100): win rate rose to ≈0.28–0.31, with multiple one- to four-step wins indicating confident greedy choices from favorable states.
* Late training (≈ epochs 1160–1190): instantaneous win rate peaked near ≈0.40–0.44 for several evaluation windows, reflecting a policy that often finds the treasure before timing out or exhausting reward.

These outcomes align with the expected learning curve for deep Q-learning on a sparse-reward navigation task: early exploration yields rare successes, followed by a phase where the network exploits emerging action values and produces shorter, repeated wins. During training I also observed repeated near-greedy rollouts (ε small) where the agent reached the goal in well under the maximum step budget, further confirming that the learned Q-values encode a usable path through the maze.

To ensure the assessment measured the policy rather than exploration noise, I performed evaluation passes with ε = 0 (pure exploitation). In those runs, the agent consistently selected the highest-valued move at each step and achieved multiple wins from diverse starting rows. The pattern of distinct, non-uniform Q-values at decision time (i.e., one action clearly preferred over the others) is a strong indicator that the network captured directional preferences around obstacles rather than acting randomly.

**Known Limitations and How I Mitigated Them**

The built-in “completion check” visualization sometimes displays only the start and goal cells without a drawn route. This behavior is tied to the evaluation helper’s state-update routine; it does not affect training and does not imply that the agent failed to move. Because course guidelines prohibit editing the provided .py files, I validated learning with **(a)** the training telemetry (wins, loss, step counts), **(b)** greedy rollouts with ε = 0, and **(c)** inspection of action preferences (Q-value asymmetry). Together these provide robust evidence that the agent learned a policy that reaches the treasure across many episodes, even when the static plot is not rendered.

*Practical note for reproducibility:* after training I save the model and reload it before evaluation to avoid accidentally reinitializing weights (which would show flat, uniformly negative Q-values and make the agent appear stationary). Evaluation then proceeds with ε = 0 and a greedy action selection loop to verify goal attainment within the step budget.

**Personal Reflection and Conclusion**

Developing the pirate intelligent agent was both a technical and personal learning journey. Much like the agent itself, I often found myself exploring uncharted paths adjusting parameters, retracing steps, and occasionally running into walls. There were moments of real frustration when the model appeared to stall, when visualizations failed to render, or when running a cell out of order caused the network to lose all of its learned weights. These experiences mirrored reinforcement learning in the truest sense: every setback was feedback. Through trial, error, and persistence, I began to understand not only *how* deep Q-learning functions but *why* it behaves the way it does.

The process taught me to read training signals as a form of dialogue between the model and the data. I learned how critical it is to maintain exploration early on to prevent premature convergence, and how even a small hyperparameter change such as modifying the training cadence, replay memory size, or epsilon decay—can significantly affect learning efficiency. The diagnostic tools I added, such as “bump\_left” and “saw\_left,” became essential windows into the agent’s internal behavior, translating abstract reward trends into meaningful feedback.

I also came to appreciate the fragility of complex systems: one misplaced reset or model reinitialization could undo hours of progress, reinforcing the importance of checkpoints, documentation, and controlled experimentation. As Gulli and Pal (2017) explain, deep learning systems thrive on iteration and incremental refinement; small improvements, consistently applied, create measurable advances in learning efficiency and generalization. Likewise, Beysolow II (2018) emphasizes that reinforcement learning depends on maintaining a careful balance between exploration and exploitation—a lesson I internalized as I tuned the pirate agent’s epsilon schedule and reward thresholds.

Despite occasional rabbit holes, this experience deepened my understanding of deep reinforcement learning and strengthened my confidence in designing and defending intelligent systems that learn through experience. Like the pirate agent itself, I learned to trust the process—each misstep was not failure, but feedback guiding me toward mastery.

**References**

Beysolow II, Y. (2018). *Applied reinforcement learning with Python: With real-world examples using TensorFlow and Keras.* Apress.

Gulli, A., & Pal, S. (2017). *Deep learning with Keras.* Packt Publishing.