**Design Defense Report**

**Human vs. Machine Problem Solving**

When a human faces a maze or treasure hunt problem, the approach usually begins with observation, intuition, and trial and error navigation. A person might analyze the structure visually, look for recognizable patterns, and rely on spatial memory to recall routes that lead to dead ends or success. Humans naturally draw on experience - for instance, remembering that turning toward open pathways or avoiding previously blocked areas increases the chance of finding the exit. A human’s strength lies in contextual reasoning and adaptive learning: understanding the “why” behind a path’s success or failure, rather than just memorizing outcomes.

By contrast, a machine approaches the same problem mathematically and programmatically. My intelligent agent doesn’t “see” the maze but interprets it as a grid of numerical values where each cell holds information about possible moves, rewards, and penalties. Instead of emotion or intuition, the algorithm uses reinforcement learning - specifically, deep Q-learning - to maximize cumulative rewards by selecting the best possible action in each state. The model doesn’t start with knowledge; it learns patterns over thousands of trials, eventually forming an optimal path based purely on data driven experience (Mnih et al., 2015).

While both human and machine approaches involve exploration and learning from mistakes, their reasoning differs. A human learns conceptually and can transfer insight to new mazes, while a machine focuses on local optimization within its training environment. Humans also tend to use less repetition, often reaching conclusions faster by applying intuition, whereas a neural network must iterate through immense trial cycles to “understand” which actions produce the best outcomes.

**How the Agent Solves the Pathfinding Problem**

My intelligent agent, modeled as a pirate searching for treasure, operates within a grid based environment called TreasureMaze. It interprets each cell in the maze as a possible state. Each move - up, down, left, or right - represents an action that transitions it from one state to another. The agent begins without any understanding of which paths lead to success or failure. Through reinforcement learning, it gradually builds a Q-table (a collection of state action values) by interacting with the environment.

When the agent takes an action, it receives a reward or penalty: positive if it reaches the treasure, negative if it hits an obstacle, and small penalties for inefficient wandering. These outcomes adjust the agent’s decision making model through backpropagation in the neural network. Over time, the Q-values converge toward an optimal policy, meaning the agent learns the best move to make from any given position (Sutton & Barto, 2018). The trained model then tests itself by playing “greedy” runs, selecting the highest valued actions to reach the treasure consistently.

The key difference from human reasoning is that the machine does not visualize or plan consciously; instead, it mathematically evaluates every possible move. This process is far more exhaustive but incredibly efficient once trained.

**Purpose of the Intelligent Agent in Pathfinding**

The intelligent agent’s purpose is to simulate autonomous problem solving in a dynamic environment. In pathfinding, the agent must not only navigate but also learn to navigate optimally. Traditional algorithms like A\* or Dijkstra’s search rely on predefined heuristics. However, deep Q-learning gives the agent adaptability - it can adjust its strategy based on environmental feedback without needing an exact map or predefined rules.

The TreasureMaze agent models real decision making systems used in robotics, self driving vehicles, and drone navigation. In these cases, an intelligent agent must continuously analyze its environment and adapt to changing conditions, just as my pirate must reorient when blocked by obstacles. This reflects how modern AI systems function in uncertain or partially observable contexts.

**Exploration vs. Exploitation**

Exploration and exploitation are two fundamental forces in reinforcement learning. Exploration means taking risks and trying new actions to discover unknown paths, while exploitation means choosing the best known actions based on current knowledge (Sutton & Barto, 2018). My agent begins with high exploration - randomly testing directions to collect data about which paths yield the highest rewards. As training progresses, exploration gradually decays, allowing exploitation to dominate. This shift ensures the agent balances learning new information with using what it already knows.

For this pathfinding problem, the ideal balance leaned toward about 10% exploration and 90% exploitation after several hundred epochs. This proportion allowed the agent to continue improving slightly while primarily reinforcing the most successful paths. Too much exploration caused erratic wandering, while too much exploitation led to premature convergence on suboptimal routes. By the end of training, the epsilon value (exploration rate) decayed to 0.01, meaning the pirate had nearly perfected its treasure hunting strategy.

**Reinforcement Learning and the Optimal Path**

Reinforcement learning (RL) enables the agent to learn through feedback - rewarding success and penalizing failure - rather than explicit instruction. Each interaction updates the neural network’s internal representation of the environment. In my project, RL guided the pirate through thousands of maze iterations, reinforcing efficient behavior with higher Q-values. The result was a model that could predict, with high confidence, the shortest or safest route to the goal without any direct human guidance.

The process mirrors behavioral psychology, where repeated rewards shape habits. Over many episodes, the agent internalized the concept of “effort leads to treasure,” transforming trial based randomness into a consistent decision making system. This demonstrates the power of reinforcement learning in environments where the correct path cannot be preprogrammed but must be discovered.

**Implementing Deep Q-Learning**

To implement deep Q-learning, I first built a neural network in Keras with an input layer equal to the maze’s total cells and an output layer representing four possible actions. The model used Rectified Linear Unit (ReLU) activations and the Adam optimizer to update weights efficiently. The agent stored past experiences in a replay memory buffer, which allowed it to train on mini batches of previous episodes. This technique stabilized learning by reducing correlation between consecutive samples (Mnih et al., 2015).

The training loop consisted of 3,000 epochs, where the agent began each run from a random cell. It predicted the best next move, executed the action, recorded the resulting reward, and updated its network through backpropagation. The system continuously adjusted its Q-values until achieving a success rate above 90%. Once complete, the model was tested on greedy runs from multiple start positions, each successfully reaching the treasure - demonstrating convergence and mastery of the maze.

**Conclusion**

This project revealed the fascinating contrast between human intuition and machine learning precision. While humans solve problems through abstract reasoning, an intelligent agent like my pirate learns by experience and data. The deep Q-learning algorithm successfully transformed random exploration into optimized decision-making, mirroring the same behavioral reinforcement patterns that guide learning in humans.

Ultimately, the agent’s purpose was not only to find a path but to learn how to learn - a principle at the core of modern artificial intelligence

**References**

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.