Design Defense: Treasure Hunt Deep Q-Learning Agent

Introduction

This project implements a Deep Q-Learning (DQN) agent (the pirate) that must find the treasure in a grid maze. The design blends practical deep-RL techniques (experience replay, a target network, epsilon-greedy exploration, and reward shaping) to produce a stable, fast-converging policy. Below I analyze how humans and machines approach this task, describe each solver's step sequence, compare the two approaches, discuss exploitation vs. exploration, explain how reinforcement learning (RL) discovers paths, evaluate algorithm choice for complex problems, and describe the implementation details of the DQN used for this game. Where relevant I cite contemporary reviews and foundational work in RL.

Human vs. Machine Approaches: High-level analysis

Humans and machines adopt different computational strategies. Humans typically combine model-based planning, spatial reasoning, and learned habits. We form mental maps, simulate future moves, use heuristics (move toward the goal when unobstructed), and learn from a few examples (fast learning) by using prior knowledge. Machine agents (especially DQN agents) rely on numerical function approximation and iterative, trial-and-error learning. They map states to action-value estimates and update parameters using gradient descent on temporally collected experience (slow, data-hungry learning). Recent reviews highlight that modern deep RL systems can exhibit both “fast” (behavioral) and “slow” (gradual parameter) learning dynamics, akin in some ways to human dual processes but realized differently (Botvinick et al., 2019).

How a human solves this maze (stepwise)

1. Perceive the board and identify obstacles and the goal.
2. Mentally compute or visualize candidate paths (shortest or promising).
3. Apply heuristics (follow corridors, avoid dead ends) or perform search (breadth/depth limited).
4. Try a path. If blocked, backtrack and revise plan.
5. Learn from failed attempts by updating mental model or strategy (fewer future backtracks). Humans typically use a mix of planning, heuristics, and episodic memory to converge quickly.

How the intelligent agent solves the problem (stepwise)

1. Observe the maze state as a flattened vector input.
2. Select an action by epsilon-greedy policy, with probability ε pick a random valid action (explore), else select the action with maximum predicted Q-value (exploit).
3. Execute the action in the environment and receive next state, scalar reward, and terminal flag.
4. Store the transition (s, a, r, s′, done) in experience replay.

5. Periodically sample minibatches from replay and compute targets. When using a target network, targets are computed with the frozen target model to stabilize updates (Mnih et al., 2015).
6. Update the online network via gradient descent on mean-squared TD (temporal-difference) error.
7. Occasionally copy weights from the online network to the target network. Repeat until performance stabilizes.

Similarities and differences

Similarities are that both humans and agents iterate on behavior using feedback. Both can exploit prior knowledge (humans via long-term memory and agents via learned network weights). Differences are that humans often rely on explicit planning and symbolic reasoning (few trials), whereas the DQN is model-free, learns a value function through many interactions, and lacks explicit symbolic planning. Humans generalize from very small data using priors. Deep RL generalizes via representation learning but needs many experience samples (Botvinick et al., 2019; Sutton & Barto, 2018).

Purpose of the intelligent agent in pathfinding

The agent's role is to learn a policy $\pi(s)$ that maps observed states to actions maximizing expected cumulative reward (reach treasure quickly, avoid dead ends). The agent formalizes pathfinding as a sequential decision problem and discovers strategies robust to stochasticity or partial observability. Compared with algorithmic search (A*, BFS), RL produces a compact policy that reacts online (no recomputation), adapts with more experience, and can incorporate complex state inputs.

Exploitation vs. Exploration: definition and ideal proportion

Exploitation uses the current policy to choose the best estimated action. Exploration chooses novel actions to discover new information. The ideal proportion depends on task complexity and data budget. For small mazes, rapid decay from high initial exploration to low exploration accelerates convergence because the environment is low dimensional and valid paths are reachable with moderate exploration. In practice, start with $\varepsilon \approx 1.0$ and decay to $\varepsilon \approx 0.05$–0.1 within the first 50–150 epochs to prioritize discovering paths early and then exploit thereafter. This scheduling is consistent with practices in deep RL and with “fast then slow” learning regimes (Botvinick et al., 2019; Sutton & Barto, 2018).

How reinforcement learning determines the path

RL estimates action-value $Q(s, a)$ reflecting expected future reward. Actions are selected to maximize Q. Through repeated transitions and Bellman updates, the Q network propagates terminal reward (treasure) back along antecedent states, shaping the value surface so the network learns higher Q for actions that lead toward the treasure. Experience replay stabilizes updates by breaking correlations between sequential transitions, and target networks provide stable targets for bootstrapping (Mnih et al., 2015; Sutton & Barto, 2018).

Evaluating algorithm choice for complex problems

For simple mazes, classical search (A*, Dijkstra) is deterministic and sample efficient. RL offers advantages when the state is high dimensional (images), rewards are delayed, or when adaptability is needed across varied mazes. DQN scales to complex perceptual inputs and non-stationary or stochastic environments, but requires more data and careful engineering (replay, target networks, reward design). Combining model-based planning with learned value functions can yield human-like efficiency for broader task classes (Botvinick et al., 2019).

Implementation of deep Q-learning in this project

The final implementation extends a basic DQN with stability and acceleration techniques:

- **Network architecture & optimization**: a small feedforward model maps flattened maze states to Q(s,·).
- **Experience Replay**: transitions appended to a replay buffer and sampled as minibatches to decorrelate data and increase sample efficiency.
- **Target Network**: a cloned target_model whose weights are periodically synchronized from the online model. Target network stabilizes TD targets (Mnih et al., 2015).
- **Epsilon schedule**: an aggressive decay from high ε down to ε_min within a controlled epoch window, enabling rapid exploration early and exploitation later.
- **Reward shaping**: amplified win reward (large positive terminal reward) and increased loss penalty accelerate credit assignment and speed convergence.

- **Training schedule**: minibatch training per episode, with target updates at a fixed frequency to maintain stability.

These choices follow best practices from deep RL literature and produce rapid convergence in this discrete maze domain when tuned appropriately (Mnih et al., 2015; Sutton & Barto, 2018).

Conclusion

Humans and machines approach pathfinding differently. Humans rely on planning and prior knowledge, while DQN agents learn value functions from experience. The implemented DQN combines experience replay, target networks, tuned epsilon decay, and reward shaping to find treasure paths reliably and rapidly. For small, visually simple mazes, search algorithms are efficient. However, DQN's generalization and ability to operate from learned representations make it a strong approach when inputs or environments scale in complexity.

References

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019). Reinforcement Learning, Fast and Slow. *Trends in Cognitive Sciences*, *23*(5), 408–422. https://doi.org/10.1016/j.tics.2019.02.006

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & 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. (2018). *Reinforcement learning: An introduction* (2nd ed.). The Mit Press.