Report – Cleaning Agent using Q-Learning & Deep Q-Networks (DQN)

Roll No – `24CS60R70`
Project Title: Intelligent Grid Cleaning Agent using Reinforcement Learning

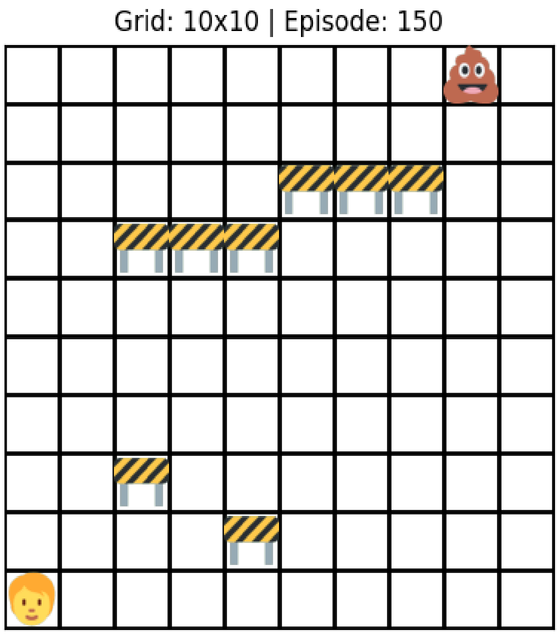
---

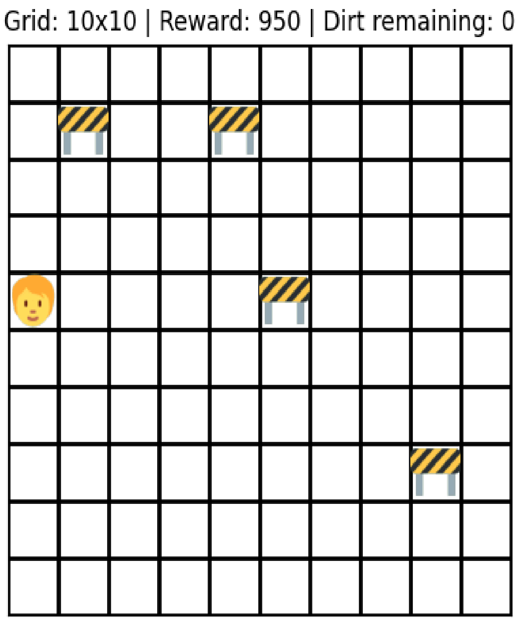
Introduction

This project explores the use of **Q-Learning** and **Deep Q-Learning (DQN)** to train an agent to clean a grid-based environment by navigating to dirt cells while avoiding obstacles. The performance of both algorithms is tested and evaluated under different grid sizes and settings, including a bonus extension with multiple dirt cells.

Environment Design

- Square grid world: ⬜⬜⬜ with walls and random obstacles 🧱
- Agent (🤖) starts at a random position
- Dirt (💩) can appear at random locations
- Actions: `UP`, `DOWN`, `LEFT`, `RIGHT`, `CLEAN`
- Rewards:
    - +10 for cleaning
    - -1 for step penalty
    - -10 for hitting wall/obstacle





Algorithms Overview

Q-Learning (Tabular RL)

- Discrete state-action value table
- Exploration handled via ε-greedy strategy
- Trained via temporal-difference update rule

DQN (Deep Q-Network)

- State represented as input to neural network
- Approximates Q-values using PyTorch model
- Uses replay buffer, batch training, and target network
- Suitable for large/continuous state spaces

Experimental Setup

Grid Sizes Tested

- 🟩 `10x10`
- 🟨 `100x100`
- 🟧 `1000x1000`
- 🟥 `10000x10000`

Execution Time Results

Q-Learning Execution Time

Grid Size	Time Taken (seconds)
10 × 10	2.27
100 × 100	0.01
1000 × 1000	0.06
10000 × 10000	9.36

DQN Execution Time

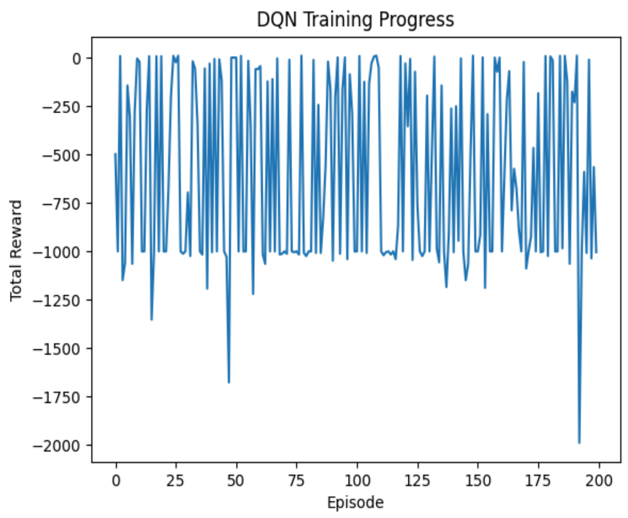
Grid Size	Time Taken (seconds)
10 × 10	14.35
100 × 100	14.95
1000 × 1000	16.71
10000 × 10000	35.22

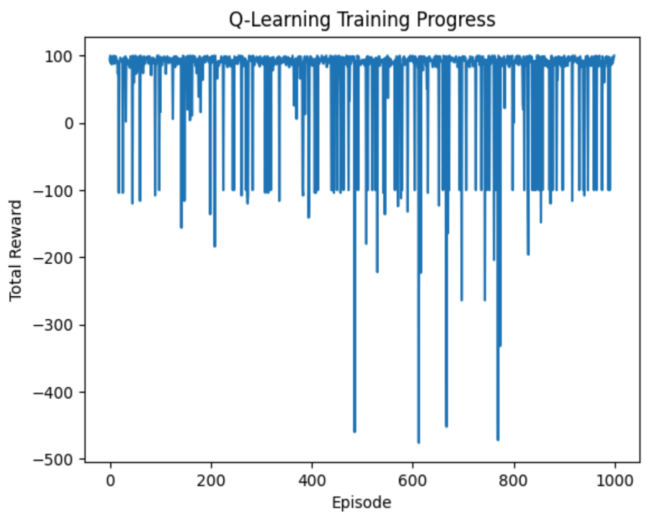
Hyperparameter Settings

Algorithm	Alpha	Learning Rate	Gamma	Epsilon	Batch Size	Target Update Freq	Max Steps	Episodes	Training Time	Throughput
Q-Learning	0.2	N/A	0.95	0.1	N/A	N/A	100	1000	15 sec	65.16 it/s
DQN	N/A	0.0005	0.9	0.1	64	2000	100	200	43 sec	4.59 it/s

Evaluation & Performance

Reward Trend (Training)





Insights & Comparison

1. Which algorithm performs better and why?

- **Q-Learning** performs very well in small to medium-sized environments .Its training output shows that it completes 1000 episodes in about 19 seconds (approximately 50.99 iterations per second).

- **DQN**, on the other hand, while being slower during training (200 episodes take around 6 minutes and 13 seconds, or 1.87 seconds per iteration), can handle larger and more complex state spaces because it uses a neural network to approximate the Q-values.

- In summary, for smaller grids, Q-Learning is faster and more efficient; however, DQN scales better when the state space is large or continuous.

2. Which one generalizes better?

- **DQN** is likely to generalize better since its neural network approximator can learn patterns over continuous states. This makes DQN more adaptable to unseen states in larger or more complex environments compared to the fixed table used in Q-Learning.

3. Discuss training vs. inference time.

Training Time:

- Q-Learning takes 19 seconds for 1000 episodes at a throughput of 50.99 it/s).

- DQN takes l6 minutes 13 seconds for 200 episodes, with 1.87 seconds per iteration

Inference Time:

- Q-Learning has nearly instantaneous inference via table lookup.

- DQN requires a forward pass through the network which is generally slower than a table lookup, but still acceptable for many applications if the network is lightweight.

4. How do the algorithms scale with grid size?

Q-Learning:

- Scales efficiently in small environments (e.g., 2.27 sec for a 10×10 grid) but the training time increases with grid size (up to 9.36 sec for a 10000×10000 grid) as the state-action table grows.

DQN:

- Scales more gracefully with grid size (training time increases from 14.35 sec for a 10×10 grid to 35.22 sec for a 10000×10000 grid) because the neural network size remains constant even as the state space expands.

5. Which one is more sample-efficient?

- **Q-Learning** appears to be more sample-efficient for small, discrete environments, achieving good performance with 1000 episodes.
- **DQN** converges with fewer episodes (200 in our case) but performs multiple updates per episode using experience replay, which may require more overall samples in complex tasks.
- Thus, while Q-Learning is sample-efficient for simpler tasks, DQN provides better performance and generalization in larger, more complex state spaces despite potentially needing more samples overall.

Conclusion

This project demonstrates the application of **Q-Learning** and **DQN** in a cleaning task, with analysis across environments of increasing complexity. The bonus extensions and heuristic-guided policies further improved agent performance.
