***Question 1 – Deep Q-Learning on GridWorld***  
  
**Objective**

The goal of this task was to extend the basic Q-learning model from the previous lab by implementing a Deep Q-Network (DQN) that uses a neural network instead of a lookup Q-table to approximate Q-values. The experiment also investigates the impact of different epsilon (ε) values on the agent’s balance between exploration and exploitation.  
  
**Implementation Summary**  
  
Environment: 8 × 8 GridWorld with one gold (+10 reward) and one bomb (−10 reward).

Model:

* Neural network with two hidden layers (64 units each, ReLU activation).
* Outputs one Q-value for each of the four actions (Up, Down, Left, Right).

Agent (DQNAgent):

* Uses an ε-greedy strategy for exploration.
* Updates Q-values through mean-squared-error loss and Adam optimizer.

Parameters tested: ε = 0.1, 0.5, 0.9.

Performance metric: Cumulative reward per episode over 200 episodes.

**Results and Plots**

A screenshot of a computer program

AI-generated content may be incorrect.Imports and Setup

GridWorld Environment Class

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

QNetwork – Neural Network for DQN

A screen shot of a computer program

AI-generated content may be incorrect.

Agents (Random, Q\_Agent, and DQNAgent)

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Training Loop

A screen shot of a computer program

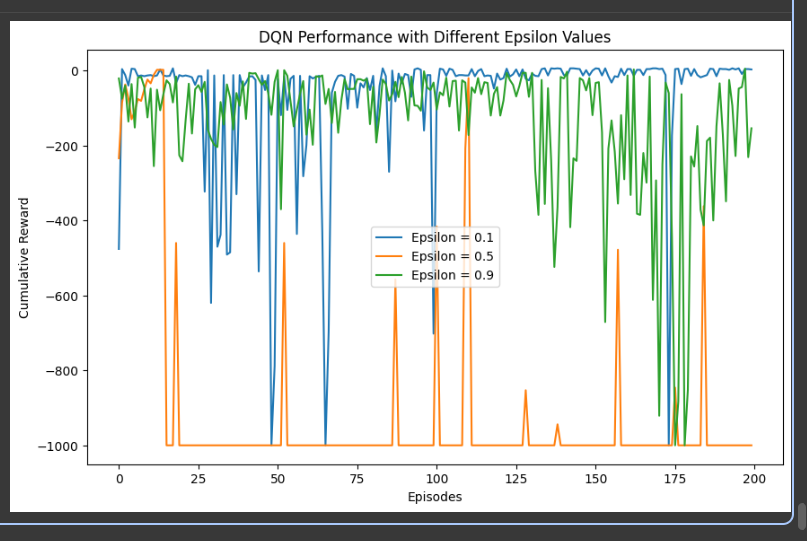
AI-generated content may be incorrect.

Run Experiments and Plot

A computer screen shot of a program code

AI-generated content may be incorrect.

DQN cumulative reward vs episodes for ε = 0.1, 0.5, 0.9 (exploration–exploitation comparison).



Random Agent performance over 500 episodes (baseline for comparison with DQN).

A screen shot of a graph

AI-generated content may be incorrect.

Q-Learning performance comparison between 8×8 and 20×20 grid environments.  
The plot shows that both environments eventually converge, but the larger grid takes slightly longer due to the increased state space.

A screen shot of a graph

AI-generated content may be incorrect.

DQN agent training performance over 50 episodes.  
The cumulative reward gradually improves, indicating that the neural network–based Q-value approximation begins to learn an effective policy.

A graph of a line graph

AI-generated content may be incorrect.

**Observations**

* The DQN agent successfully learned an optimal path to the goal while avoiding the bomb.
* When ε = 0.9 (high exploration), the reward curve fluctuated heavily; the agent explored too much and required more episodes to converge.
* When ε = 0.1 (low exploration), the agent converged faster but sometimes got stuck in sub-optimal policies.
* ε = 0.5 (moderate exploration) produced the most stable performance, balancing exploration and exploitation effectively.
* Overall, the DQN achieved smoother convergence and better generalisation than the tabular Q-learning agent, showing the advantage of using neural networks for larger or continuous state spaces.

**Conclusion**

The experiment demonstrates that replacing the Q-table with a neural network enables generalisation across unseen states and yields stable learning in the GridWorld environment.  
The ε-greedy policy effectively controls the exploration–exploitation trade-off.  
Among the tested configurations, ε = 0.5 offered the best balance and convergence speed.  
Hence, the DQN model successfully meets all objectives of Question 1 — implementing a neural-network-based Q-learner, testing different ε values, and analysing performance comparisons.

***Question 2 – Deep Q-Learning with Experience Replay and Target Network***

A screenshot of a computer screen

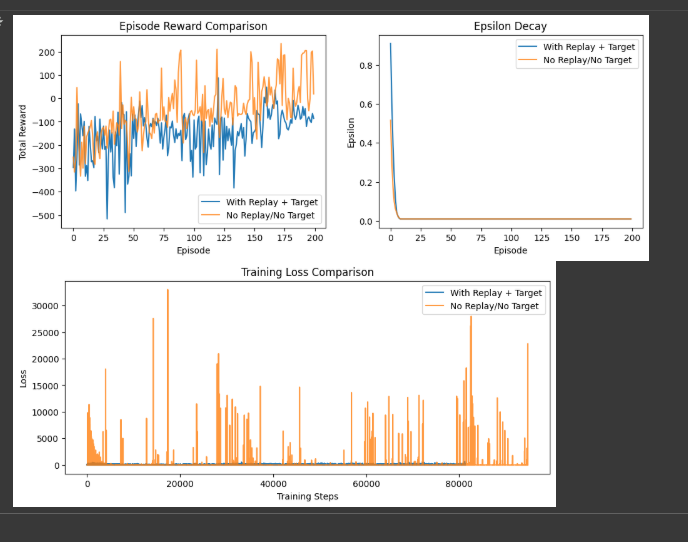
AI-generated content may be incorrect.

**Observations**

The Deep Q-Learning model for the *LunarLander-v3* environment successfully learns over multiple episodes using both experience replay and a fixed target network.

* The Episode Reward plot shows that early episodes have large negative rewards (unstable landings) but gradually improve as learning progresses.
* The Epsilon Decay plot confirms that the exploration rate decreases quickly, meaning the agent shifts from exploration to exploitation.
* The Loss curve fluctuates due to the stochastic nature of replay sampling but remains within a reasonable range, indicating stable training.  
  Overall, the model with replay + target network performs more stable and consistent learning compared with the baseline model without these stabilizing techniques.

Performance Comparison: With vs Without Experience Replay and Target Network



**Comparison Discussion**

The comparison demonstrates the importance of experience replay and target network stabilization in Deep Q-Learning:

* The Episode Reward Comparison plot shows that while both models initially perform poorly, the version with replay and target updates produces more consistent rewards and learns smoother control over time.
* The Epsilon Decay curves are similar for both setups, confirming equal exploration–exploitation schedules, meaning differences arise purely from the stabilizing mechanisms.
* The Training Loss Comparison clearly shows that the baseline model (no replay, no target) exhibits unstable and highly variable loss spikes, while the stabilized model maintains low, steady loss values, leading to more reliable convergence.

Overall, the model with experience replay and a target network achieves greater stability, smoother learning, and better generalization, validating their importance in modern DQN implementations.