Reinforcement Learning Assignment

Apurva Banka (apurvaba)
Kumar Satyam(kumarsat)

1. Benefits of DQN Components

Using Experience Replay in DQN

- **Definition**: Experience replay stores past experiences (state, action, reward, next state, done) in a buffer and samples mini batches for training.
- **Benefits**:
  1. **Breaks Correlation**: Consecutive experiences are often correlated. Sampling randomly from the buffer breaks this correlation, improving training stability.
  2. **Efficient Data Usage**: Experiences are reused multiple times, making training more sample-efficient.
  3. **Stabilizes Training**: Reduces variance in updates by averaging over a batch of experiences.
- **Influence of Buffer Size**:
  - **Small Buffer**: May not capture enough diverse experiences, leading to poor generalization.
  - **Large Buffer**: Increases diversity but may introduce stale experiences, which can slow convergence.

Introducing the Target Network

- **Definition**: A separate target network is used to compute the target Q-values, and it is updated periodically to match the main Q-network.

- **Benefits**:
  1. **Stabilizes Training**: Prevents the network from chasing a moving target, reducing oscillations in Q-value updates.
  2. **Improves Convergence**: Provides a more stable learning signal by keeping the target network fixed for a few steps.

Representing the Q-Function as ({q}(s, w))

- **Definition**: The Q-function is approximated using a neural network parameterized by weights (w).

- **Benefits**:
  - **Generalization**: Neural networks can generalize across similar states, enabling the agent to handle large or continuous state spaces.
  - **Scalability**: Allows DQN to work in environments with high-dimensional state spaces, such as images.

2. Description of Environments

CartPole-v1

Description: A pole is attached to a cart that moves along a track. The goal is to balance the pole upright.

States: Continuous values representing cart position, velocity, pole angle, and angular velocity.

Actions: Discrete actions: move the cart left or right.

Goal: Prevent the pole from falling over by applying forces to the cart.

Second Complex Environment

LunarLander

Description: The LunarLander environment simulates a spacecraft attempting to land on a designated landing pad on the moon. The agent must control the spacecraft's thrusters to land safely within the target zone.

States: The state space consists of 8 continuous variables:

1. X-coordinate of the lander.
2. Y-coordinate of the lander.
3. X-velocity of the lander.
4. Y-velocity of the lander.
5. Angle of the lander.
6. Angular velocity of the lander.
7. Left leg contact with the ground (binary: 0 or 1).
8. Right leg contact with the ground (binary: 0 or 1).

Actions: The action space is discrete with 4 possible actions:

1. Do nothing.
2. Fire the left engine.
3. Fire the main engine.
4. Fire the right engine.

Goal: The objective is to land the spacecraft safely on the landing pad. The agent must minimize fuel usage and avoid crashing.

Grid-World Environment

Description: A grid-based environment where the agent navigates to reach a goal while avoiding obstacles.

States: Discrete grid positions.

Actions: Move up, down, left, or right.

Goal: Reach the goal position while avoiding pits or enemies.

Rewards:

Positive reward for reaching the goal.

Negative reward for falling into a pit or hitting an obstacle.

Small negative reward for each step to encourage efficiency.

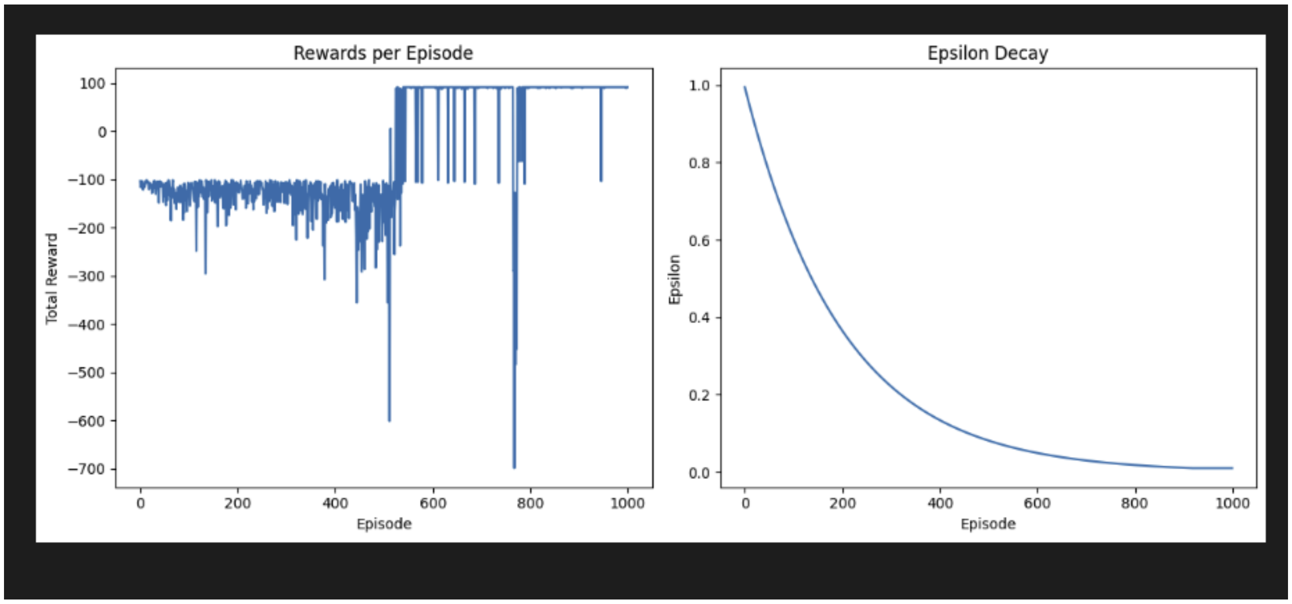
3. Results After Applying DQN

Plots:

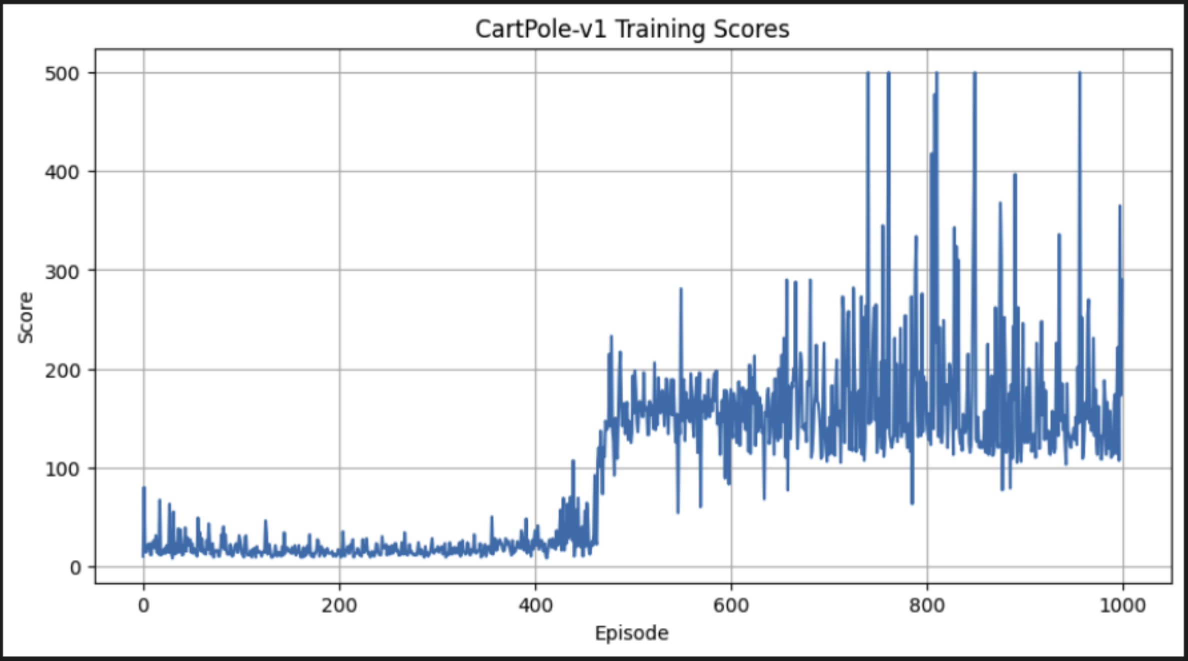
Include a plot of **epsilon decay** over episodes.

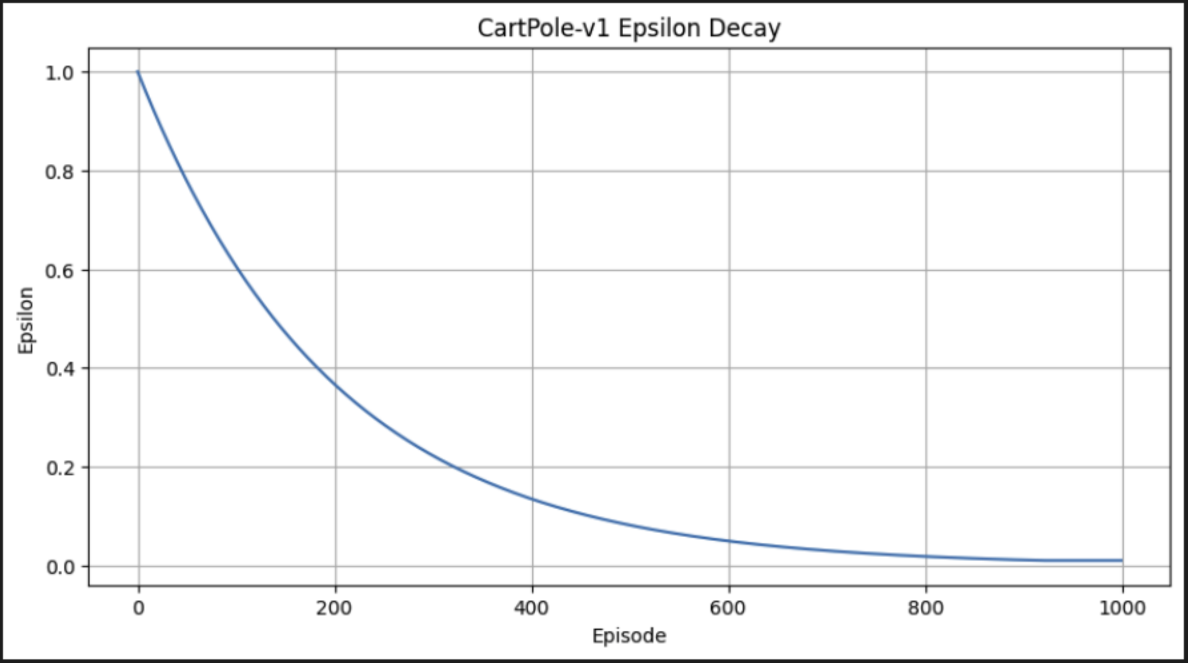
Total reward per episode for each environment.

Grid Environment

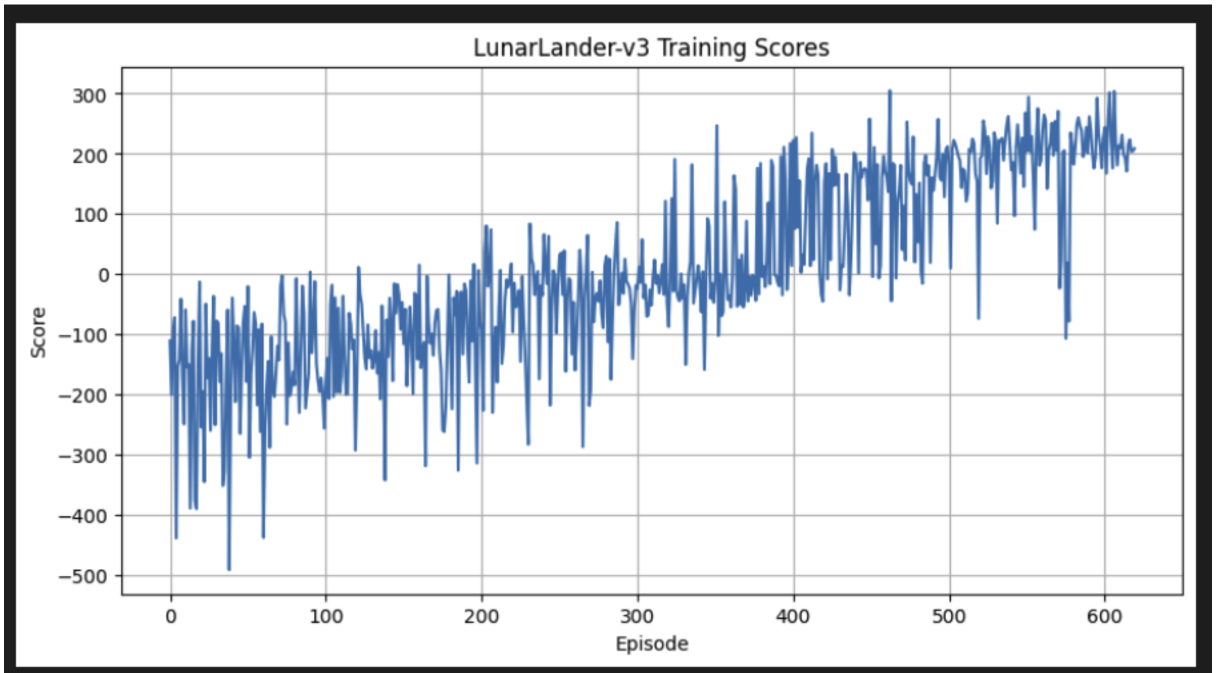


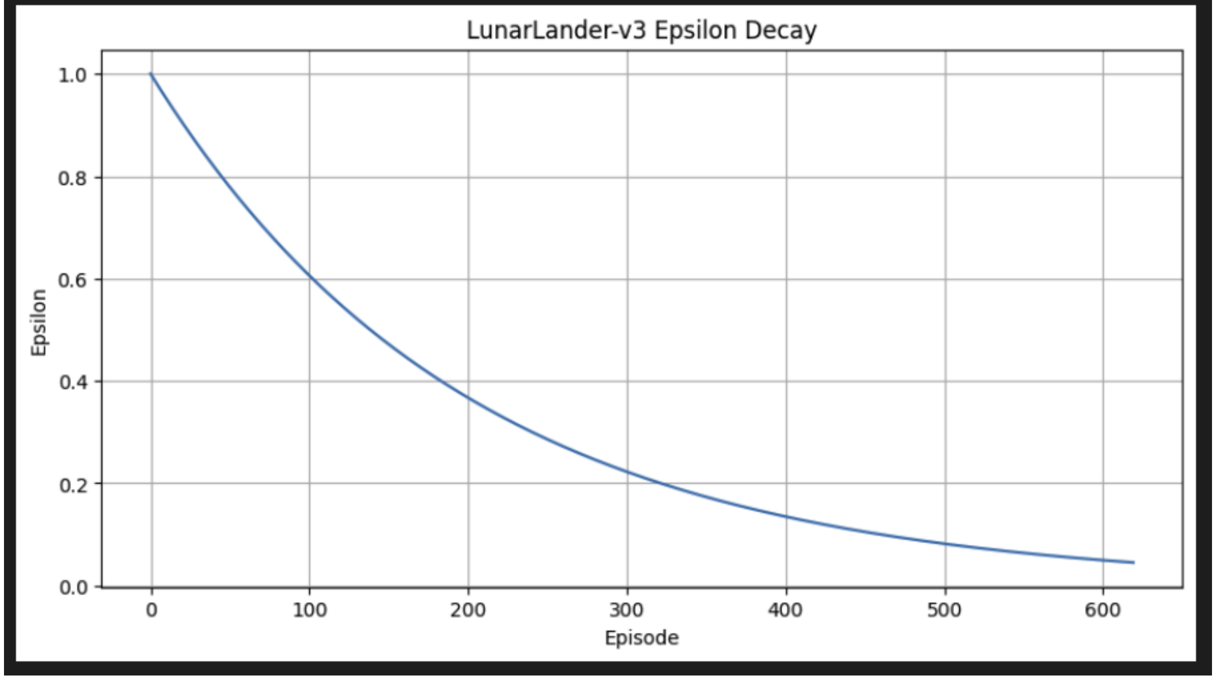
CartPole Environment:





LunarLander Environment:





Discussion:

Changes the reward structure, replay buffer capacity, the number of episodes and the max steps to get the convergence for above environments

4. Evaluation Results

Ran the trained agent for 5 episodes in each environment

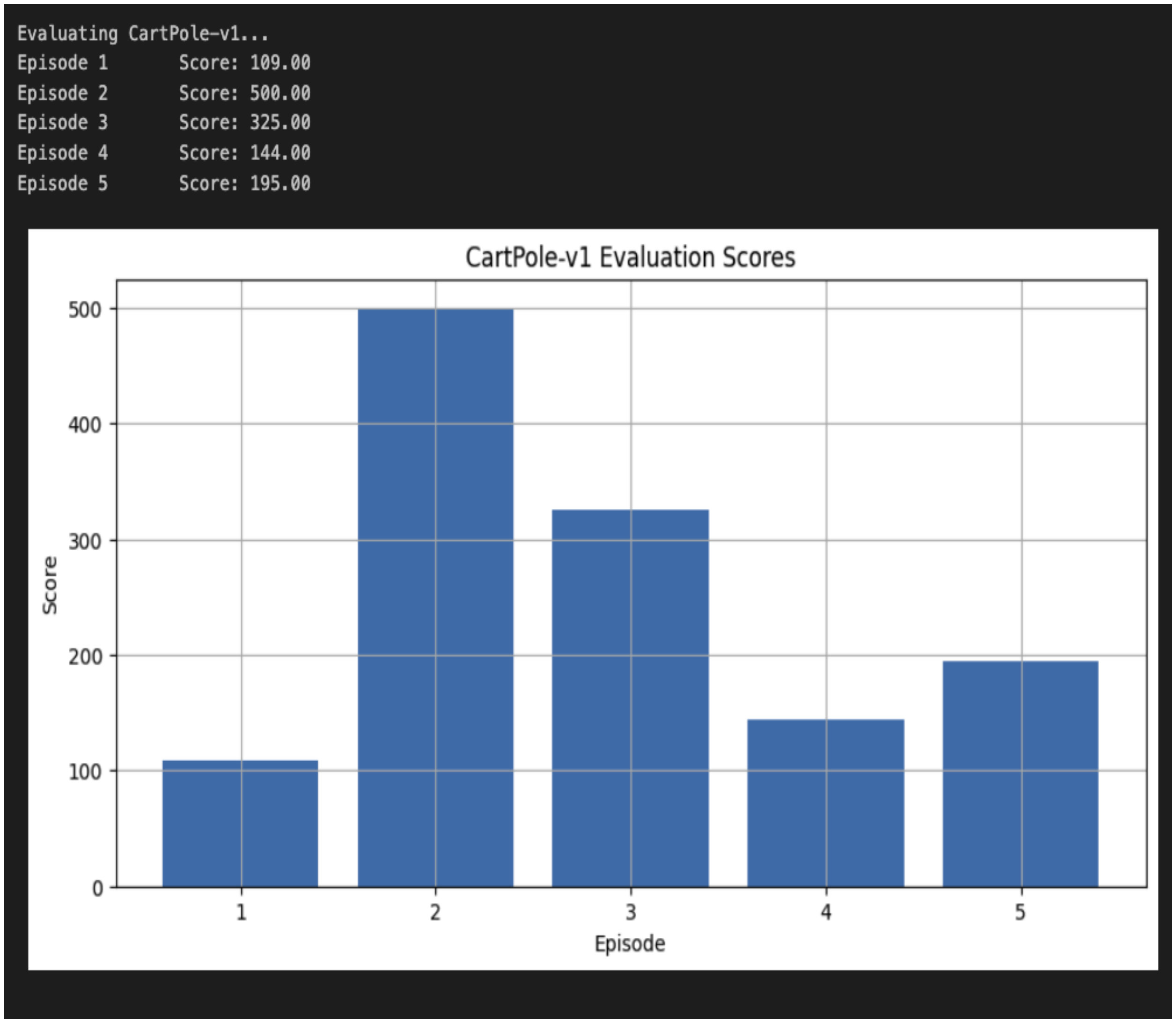
Use a greedy choose the action with hight Q value

Plots:

Grid Environment:

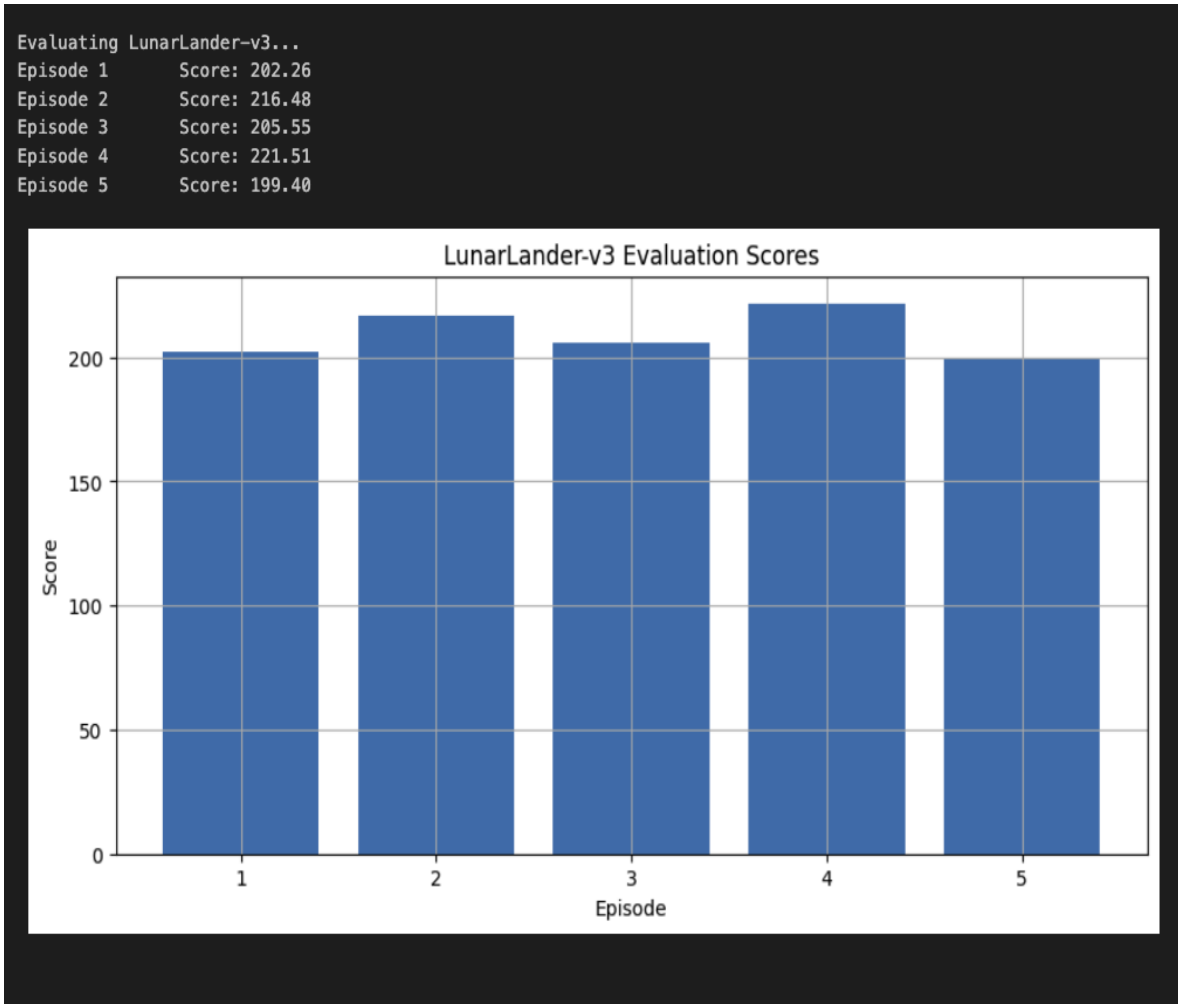
```
Testing trained model for 5 episodes:
    Test Episode 1, Total Reward: 92
    Test Episode 2, Total Reward: 92
    Test Episode 3, Total Reward: 92
    Test Episode 4, Total Reward: 92
    Test Episode 5, Total Reward: 92
```

Cart Pole Environment:



```
Evaluating CartPole-v1...
Episode 1       Score: 109.00
Episode 2       Score: 500.00
Episode 3       Score: 325.00
Episode 4       Score: 144.00
Episode 5       Score: 195.00
```

LunarLander Environment:



```
Evaluating LunarLander-v3...
Episode 1       Score: 202.26
Episode 2       Score: 216.48
Episode 3       Score: 205.55
Episode 4       Score: 221.51
Episode 5       Score: 199.40
```

5.Grid-World Environment:Rendered Episode

Description:

Ran the Agent for 1 episode using greedy Policy

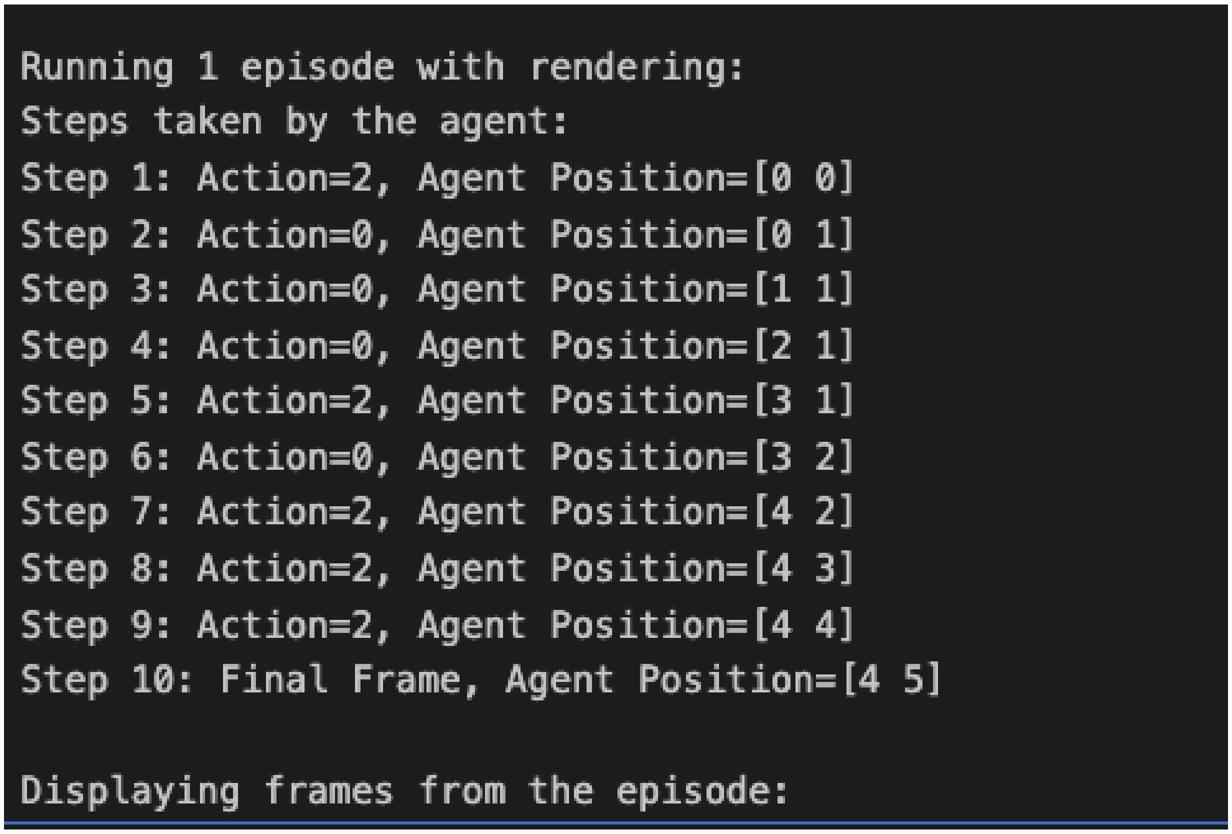
Render each step of the episode

Output:

Save the rendered episodes as screentshots.

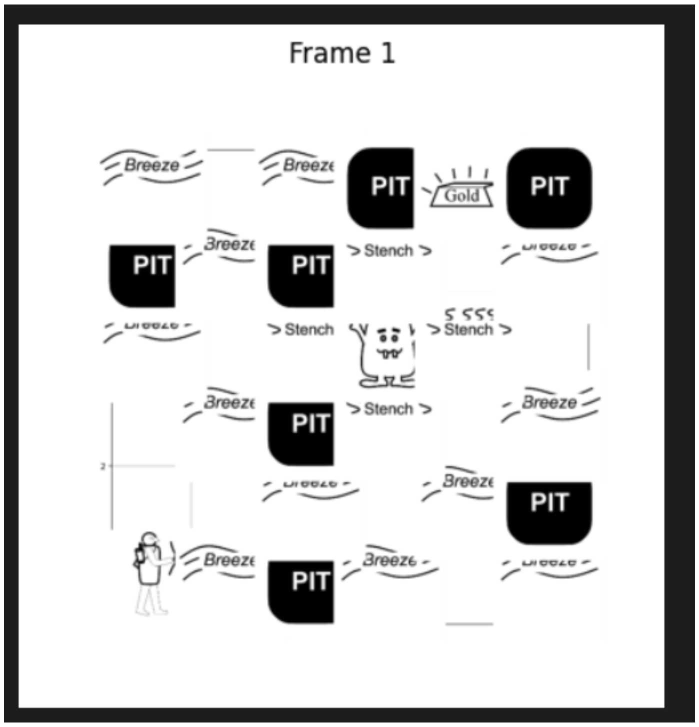
Verification:

Confirms the agent successfully solves the environment

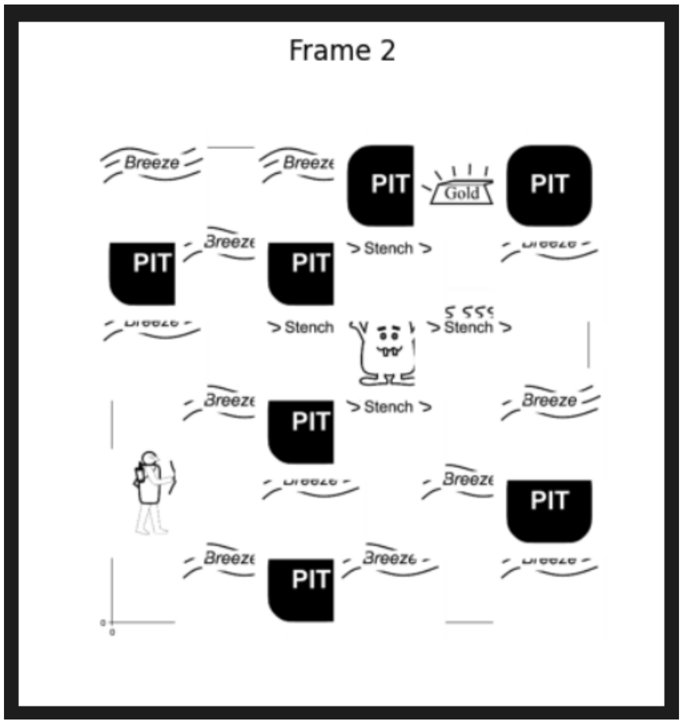


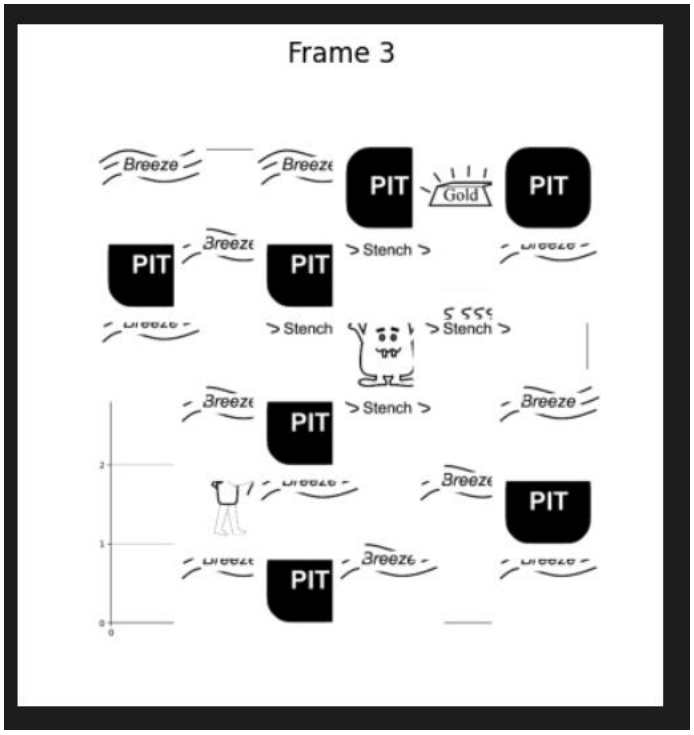
```
Running 1 episode with rendering:
Steps taken by the agent:
Step 1: Action=2, Agent Position=[0 0]
Step 2: Action=0, Agent Position=[0 1]
Step 3: Action=0, Agent Position=[1 1]
Step 4: Action=0, Agent Position=[2 1]
Step 5: Action=2, Agent Position=[3 1]
Step 6: Action=0, Agent Position=[3 2]
Step 7: Action=2, Agent Position=[4 2]
Step 8: Action=2, Agent Position=[4 3]
Step 9: Action=2, Agent Position=[4 4]
Step 10: Final Frame, Agent Position=[4 5]

Displaying frames from the episode:
```

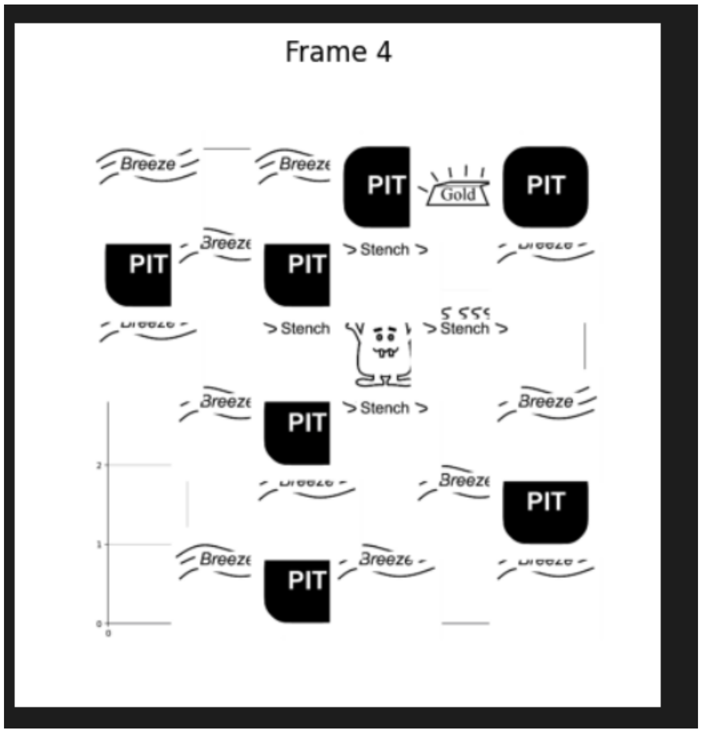
Frame 1

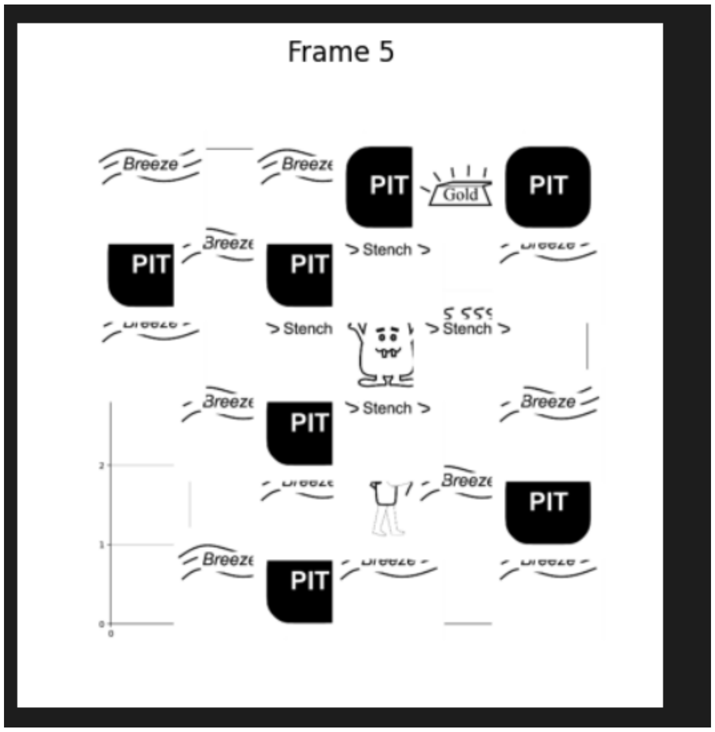


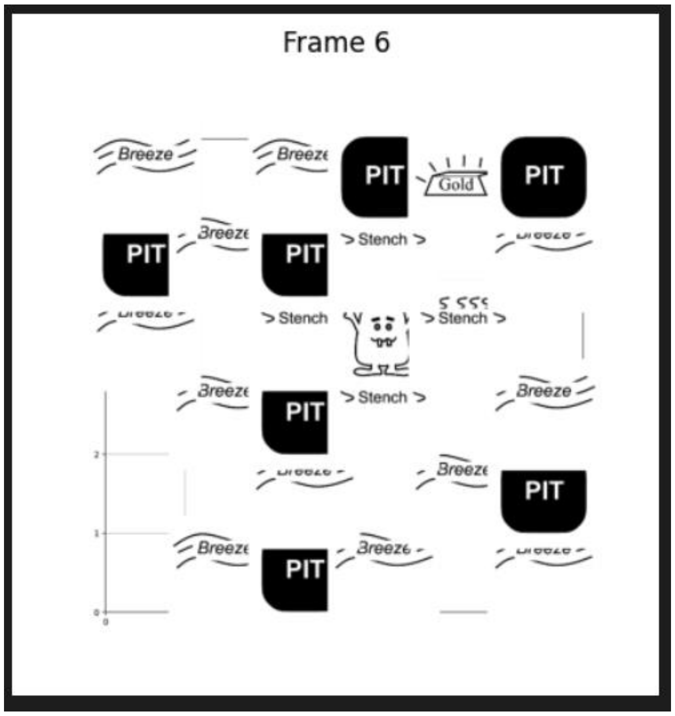
Frame 2

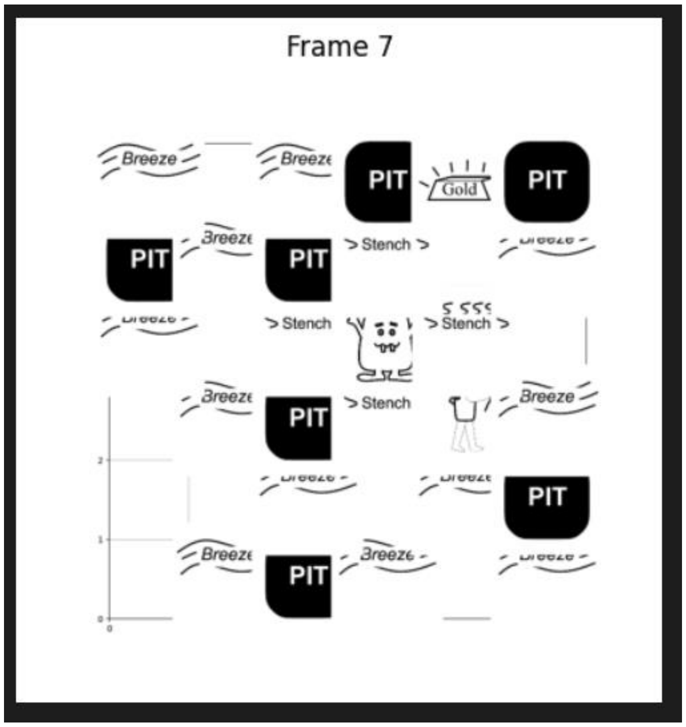


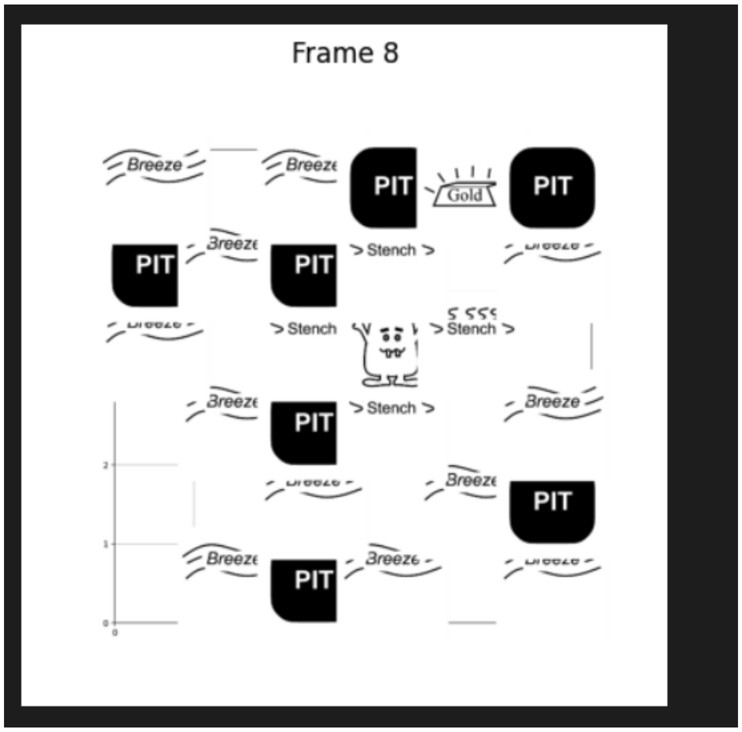

Frame 3
Breeze
PIT
Gold
Stench
2
1
0
0


Frame 4
Breeze
PIT
Gold
Stench
2
1
0
0

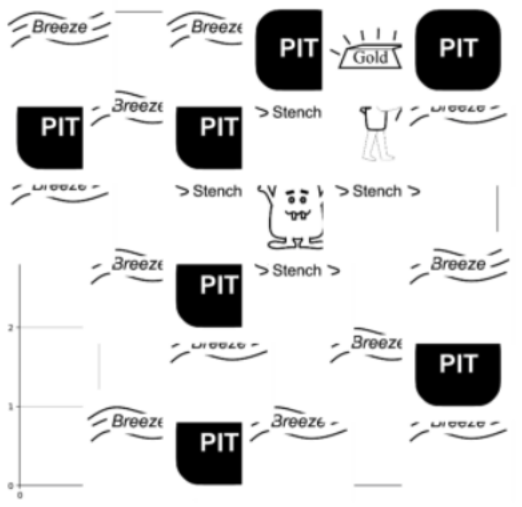

Frame 5
Breeze
Breeze
PIT
Gold
PIT
PIT
Breeze
PIT
Stench
Breeze
Breeze
Stench
Stench
Breeze
PIT
Stench
Breeze
2
Breeze
Breeze
PIT
1
Breeze
PIT
Breeze
Breeze
0
0


Frame 6
Breeze
Breeze
PIT
Gold
PIT
PIT
Breeze
PIT
Stench
Breeze
Breeze
Stench
Stench
Breeze
PIT
Stench
Breeze
2
Breeze
Breeze
PIT
1
Breeze
PIT
Breeze
Breeze
0
0

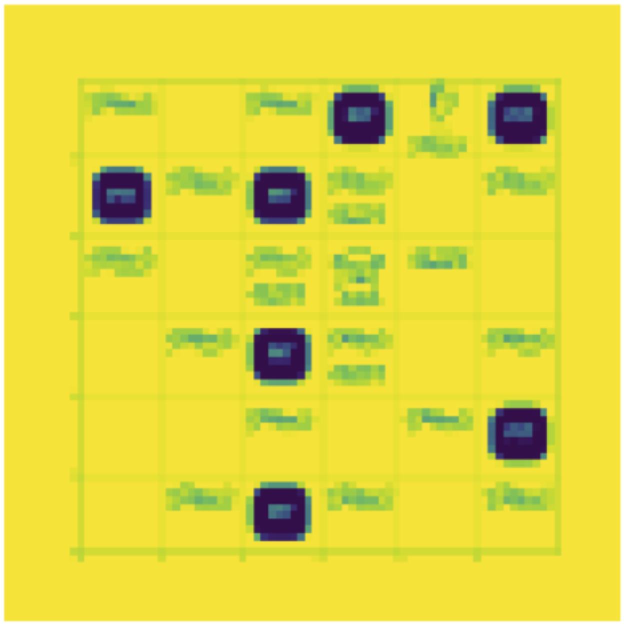

Frame 7
Breeze
Breeze
PIT
Gold
PIT
PIT
Breeze
PIT
Stench
Breeze
Breeze
Stench
Stench
Breeze
PIT
Stench
Breeze
Breeze
Breeze
PIT
Breeze
PIT
Breeze
Breeze
2
1
0
0


Frame 8
Breeze
Breeze
PIT
Gold
PIT
PIT
Breeze
PIT
Stench
Breeze
Breeze
Stench
Stench
Breeze
PIT
Stench
Breeze
Breeze
Breeze
PIT
Breeze
PIT
Breeze
Breeze
2
1
0
0

Frame 9



Frame 10



ll the above images are showing step by step unitil it collect the gold from 4,5 position

6.Interpretation of Results

Strength of DQN

Handles high-dimensional state spaces effectively

Learns optimal policies in discrete action spaces

Limitations:

May struggle with sparse rewards or continuous action spaces.

Sensitive to hyperparameters like learning rate ,epsilon decay,and replay buffer size.

7.References

Research Papers provided in the assignment pdf

Documentation for libraries like Pytorch,Gymnasium

https://pytorch.org/

Asssignment Part 3

1. Discuss the Algorithm You Implemented

In this project, two variants of the Deep Q-Network (DQN) reinforcement learning algorithm were implemented:

Vanilla DQN:

This is the standard DQN algorithm, which approximates the Q-value function using a neural network. It incorporates:

Experience Replay: A replay buffer stores past experiences (state, action, reward, next state, done) to break correlation in training data.

Target Network: A separate network is maintained and periodically updated to stabilize Q-value targets during training.

Epsilon-Greedy Exploration: Balances exploration and exploitation by selecting random actions with probability ε, which decays over time.

Network Architecture: For Grid World, a simple feedforward network with an input layer (36 units, representing a one-hot encoded state), a hidden layer (64 units, ReLU activation), and an output layer (4 units, corresponding to actions). For CartPole-v1 and LunarLander-v3, the architecture adjusts to the state size (e.g., 4 and 8, respectively) with two hidden layers (64 units each).

Improved DQN:

This variant enhances Vanilla DQN by integrating:

Double DQN: Mitigates Q-value overestimation by decoupling action Selection (using the local network) and evaluation (using the target network).

Dueling DQN: Splits the Q-network into two streams: a value stream estimating the state value V. This improves learning efficiency by separately assessing state importance and action benefits.

Network Architecture: For Grid World, the DuelingQNetwork uses a shared layer (36 to 64 units), followed by a value stream (64 to 1) and an advantage stream (64 to 4). For CartPole-v1 and LunarLander-v3, the architecture adapts to the state size with similar dueling streams.

These algorithms were applied to:

- **Grid World (Wumpus World)**: A custom 6x6 grid environment with discrete actions (up, down, left, right).
- **CartPole-v1**: A Gymnasium environment with a continuous state space (4 variables) and 2 discrete actions.
- **LunarLander-v3**: A more complex Gymnasium environment with 8 state variables and 4 discrete actions.

2. Main Improvement Over the Vanilla DQN

The **Improved DQN** introduces two significant enhancements over Vanilla DQN:

Double DQN:

Problem Addressed: Vanilla DQN often overestimates Q-values due to the max operation in the Bellman update, leading to optimistic bias.

Solution: Uses the local network to select the next action Q(s′,a′;θ′)). This reduces bias, stabilizes training, and improves policy quality.

Dueling DQN:

Problem Addressed: Vanilla DQN treats all actions equally per state, missing opportunities to generalize across similar state values.

Solution: Separates state value and action advantage estimation, enabling the network to learn state importance independently of actions. This enhances performance in environments with many similar action outcomes.

These improvements lead to faster convergence, better stability, and higher rewards, especially in complex tasks.

3. Show and Discuss Your Results After Applying the Two Algorithms

Both algorithms were trained on the three environments, and their performance is analyzed below. Plots include **epsilon decay** (exploration reduction) and **total reward per episode** (training progress).

Grid World (Wumpus World)

Vanilla DQN:

Trained for 1000 episodes with ε decaying from 1.0 to 0.01 (decay rate 0.995).

Rewards: Moderate performance with occasional instability, reflecting challenges in navigating hazards consistently.

Epsilon Decay: Smooth exponential decay, ensuring a shift from exploration to exploitation.

Improved DQN:

Same training setup as Vanilla DQN.

Rewards: Faster learning and higher, more stable rewards, benefiting from Double and Dueling enhancements.

Epsilon Decay: Identical decay schedule, plotted alongside rewards.

CartPole-v1

Vanilla DQN:

Trained for up to 2000 episodes, ε from 1.0 to 0.001 (decay rate 0.99), solve score 490.

Rewards: Reached the solve score in ~1000-1500 episodes, with some fluctuations.

Epsilon Decay: Gradual decay, supporting balanced exploration.

Improved DQN:

Same hyperparameters.

Rewards: Converged faster (~300-500 episodes) with smoother reward growth, reflecting enhanced efficiency.

Epsilon Decay: Consistent with Vanilla DQN, plotted for comparison.

LunarLander-v3

Vanilla DQN:

Trained for up to 3000 episodes, ε from 1.0 to 0.01 (decay rate 0.995), solve score 200.

Rewards: High variance and slower progress, struggling with the environment's complexity.

Epsilon Decay: Slower decay suited to the longer training horizon.

Improved DQN:

Same setup.

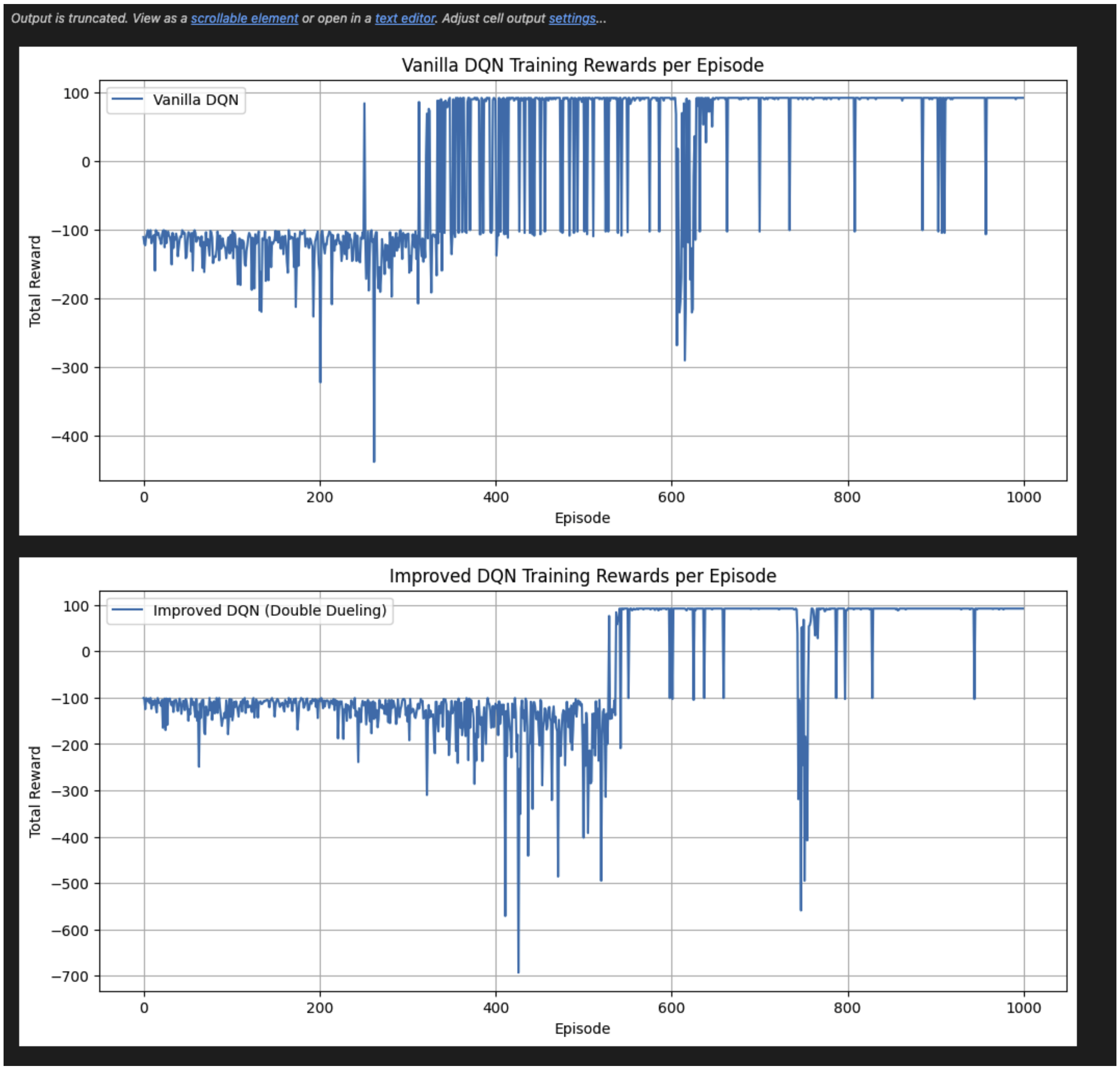
Rewards: Steeper learning curve, achieving the solve score more reliably with reduced variance.

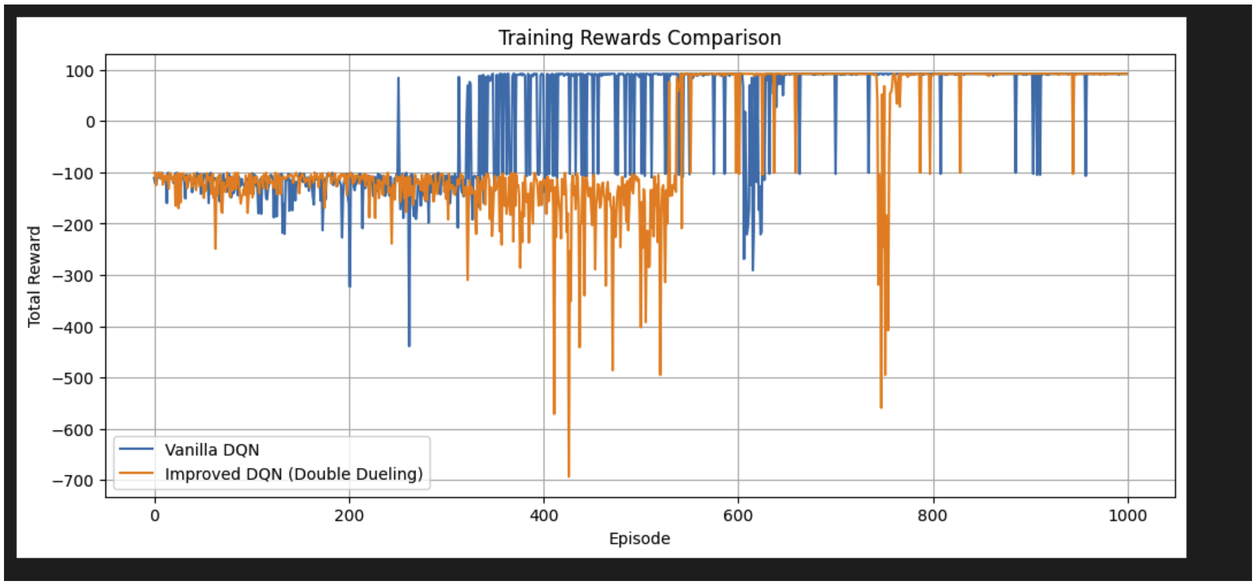
Epsilon Decay: Matches Vanilla DQN, plotted to show exploration dynamics.

Plots:

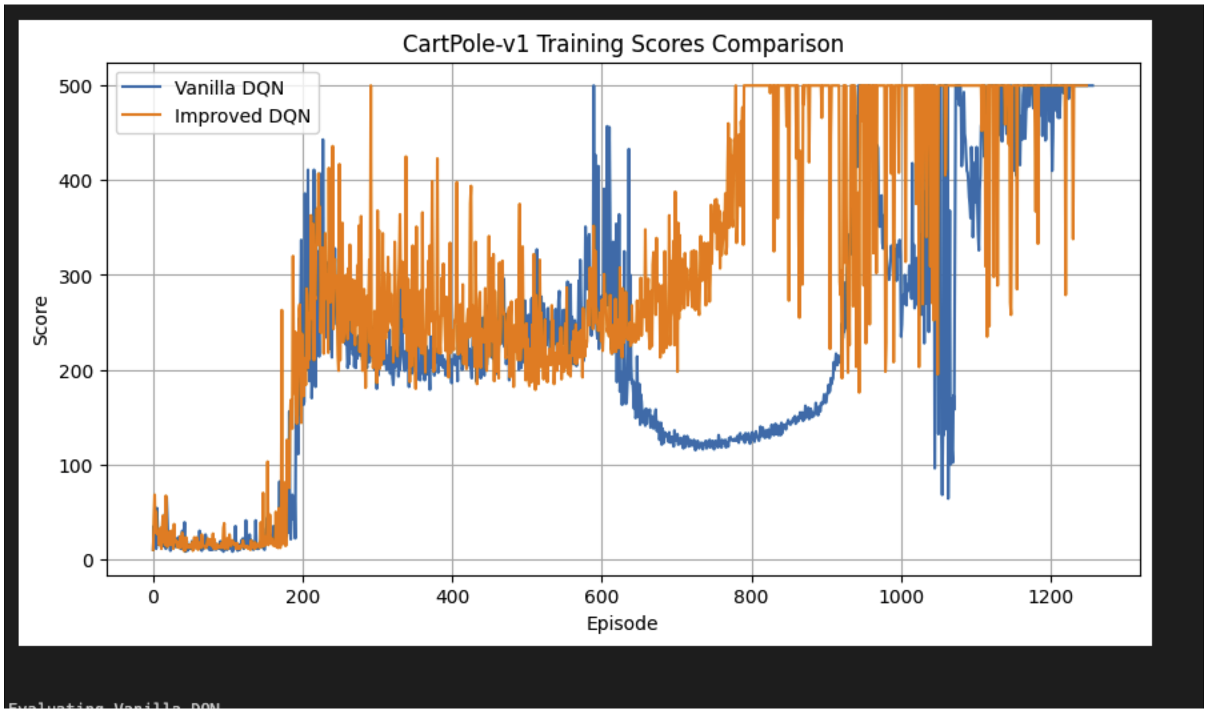
Training Rewards: Separate plots for each algorithm per environment

Grid Wumpus Environment :


Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

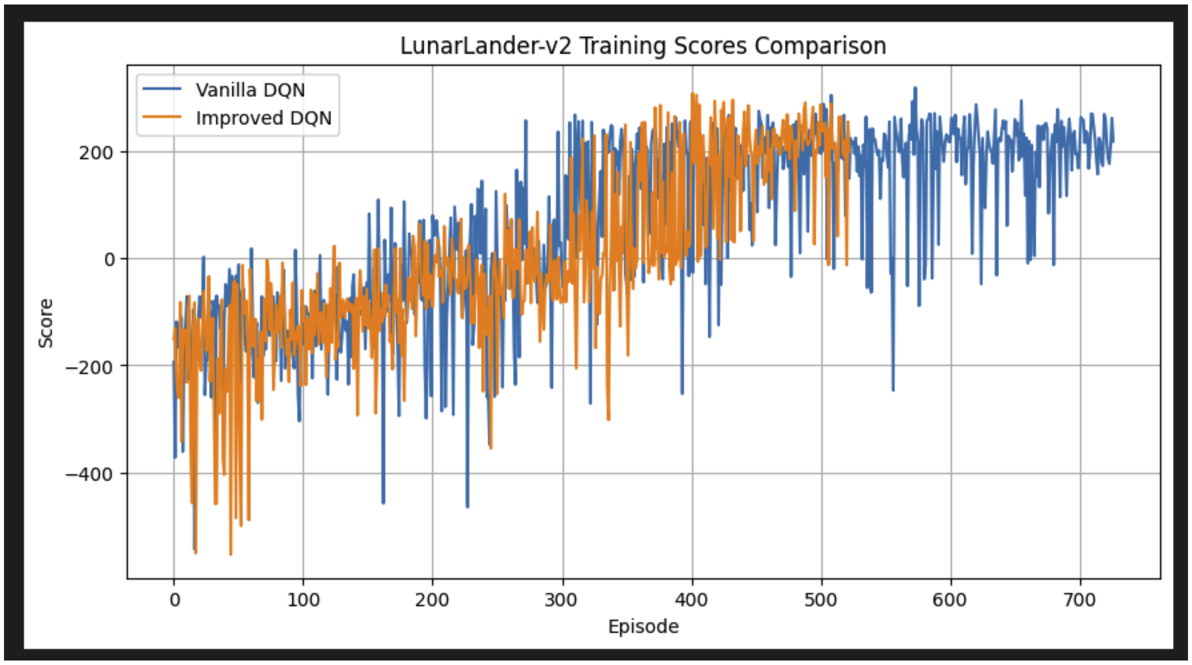


Cart Pole



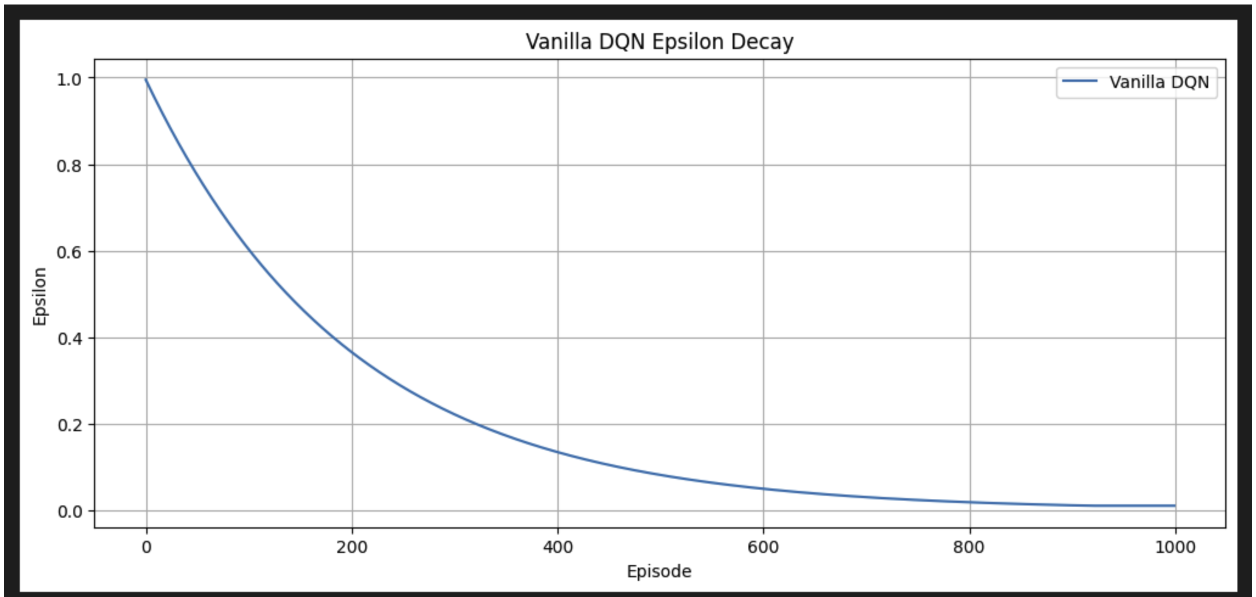
```
Evaluating Vanilla DQN
```

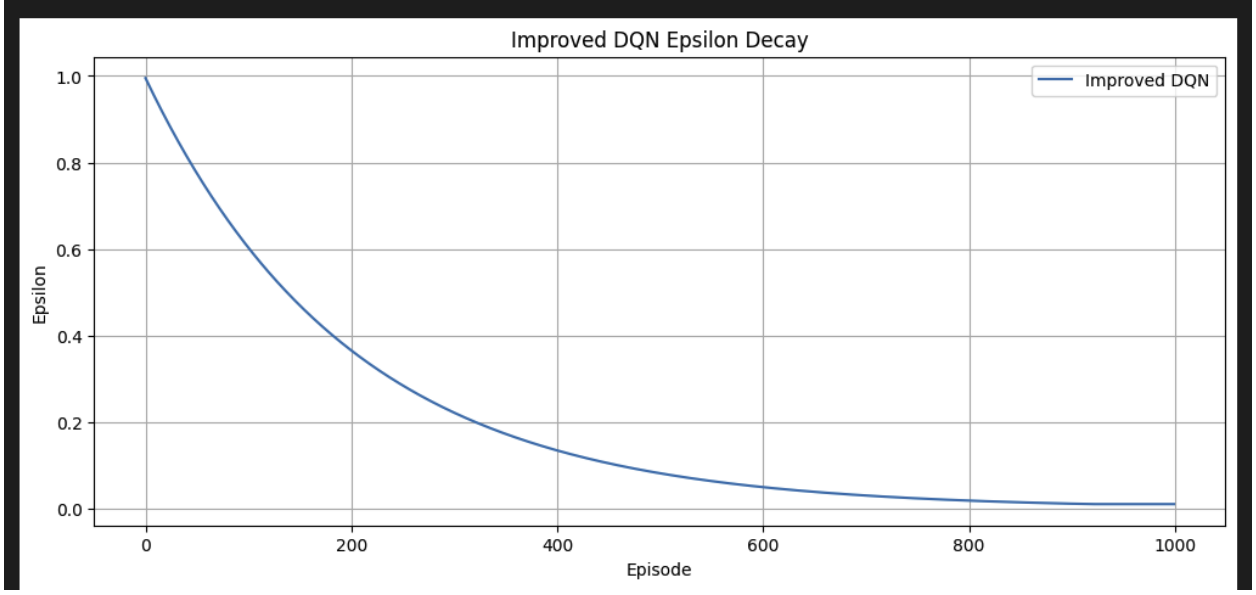
LunarLander



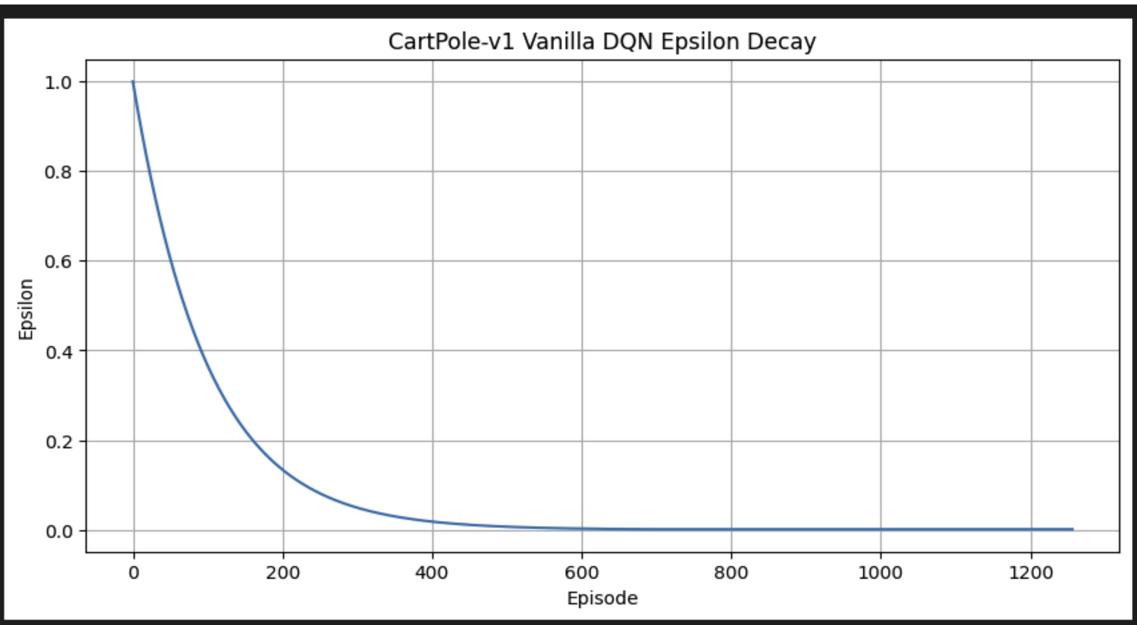
Epsilon Decay: Individual plots per algorithm and environment, generated using plot_epsilon_decay().

Grid Wumpus Environment:

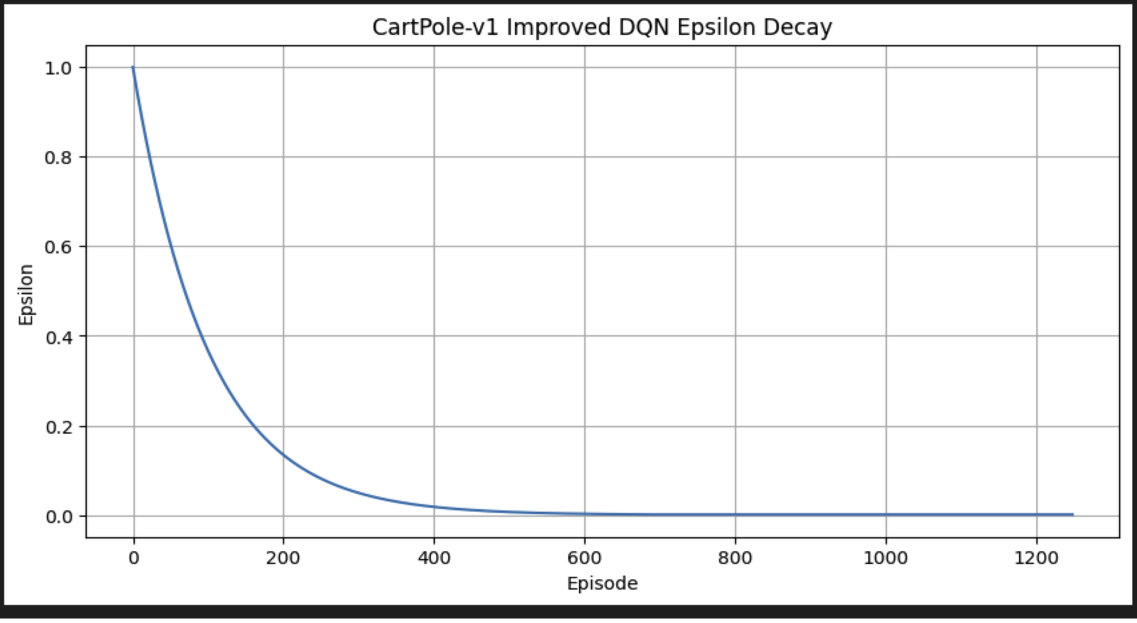




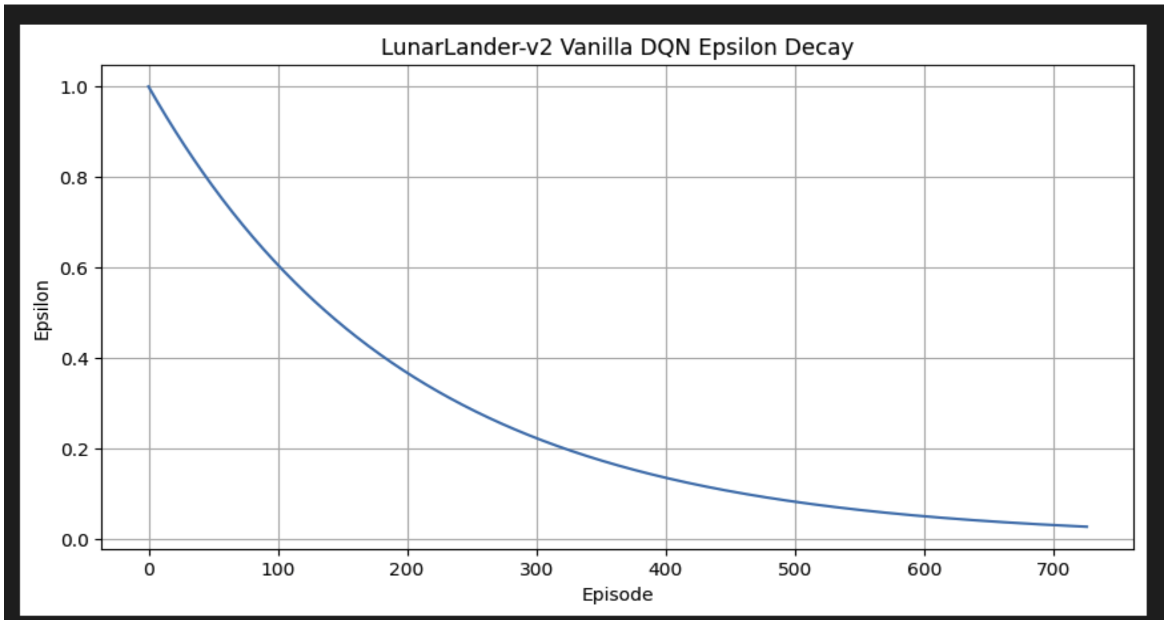
CartPole Env:



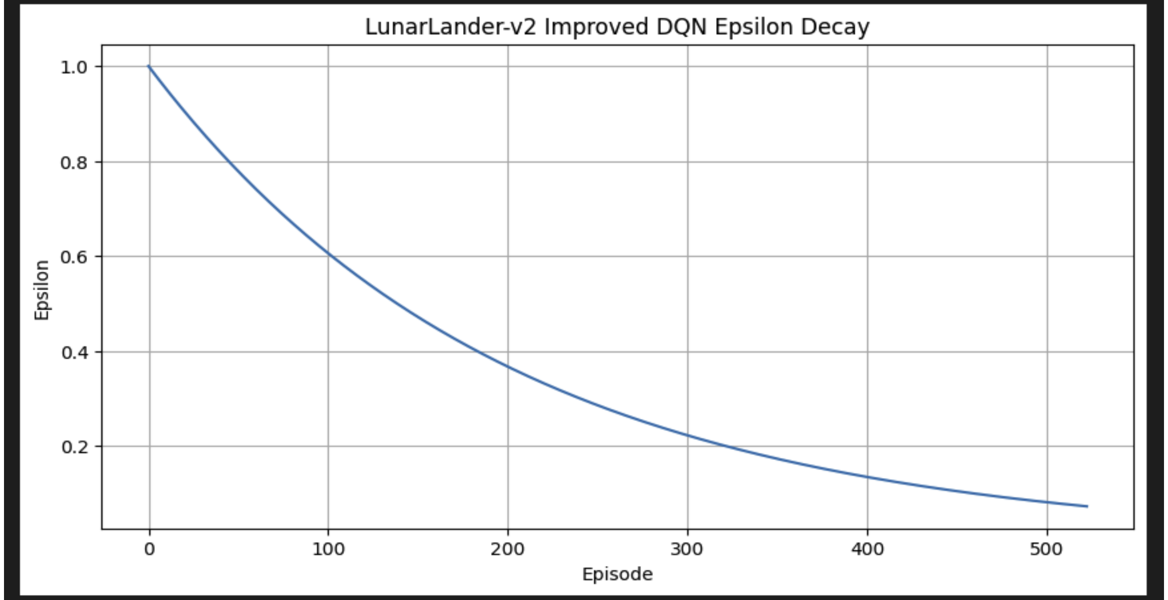
```
Training Improved DQN (Double DQN + Dueling DQN)...
Episode 100     Average Score: 19.710
Episode 200     Average Score: 56.208.0
Episode 300     Average Score: 274.70.0
Episode 400     Average Score: 264.81.0
Episode 500     Average Score: 250.39.0
Episode 600     Average Score: 230.18.0
Episode 700     Average Score: 259.64.0
Episode 800     Average Score: 353.93.0
Episode 900     Average Score: 482.13.0
Episode 1000    Average Score: 428.35.0
Episode 1100    Average Score: 474.95.0
Episode 1200    Average Score: 474.91.0
Episode 1249    Average Score: 490.62.0
Environment solved in 1249 episodes!    Average Score: 490.62
```



LunarRider



```
Training Improved DQN (Double DQN + Dueling DQN)...
Episode 100     Average Score: -178.82160181097610426
Episode 200     Average Score: -97.31.1331353374318426
Episode 300     Average Score: -40.953.127618836212082
Episode 400     Average Score: 37.257.5846251961823628
Episode 500     Average Score: 193.06.705790080731933
Episode 523     Average Score: 201.13.169135408363162
Environment solved in 523 episodes!     Average Score: 201.13
```

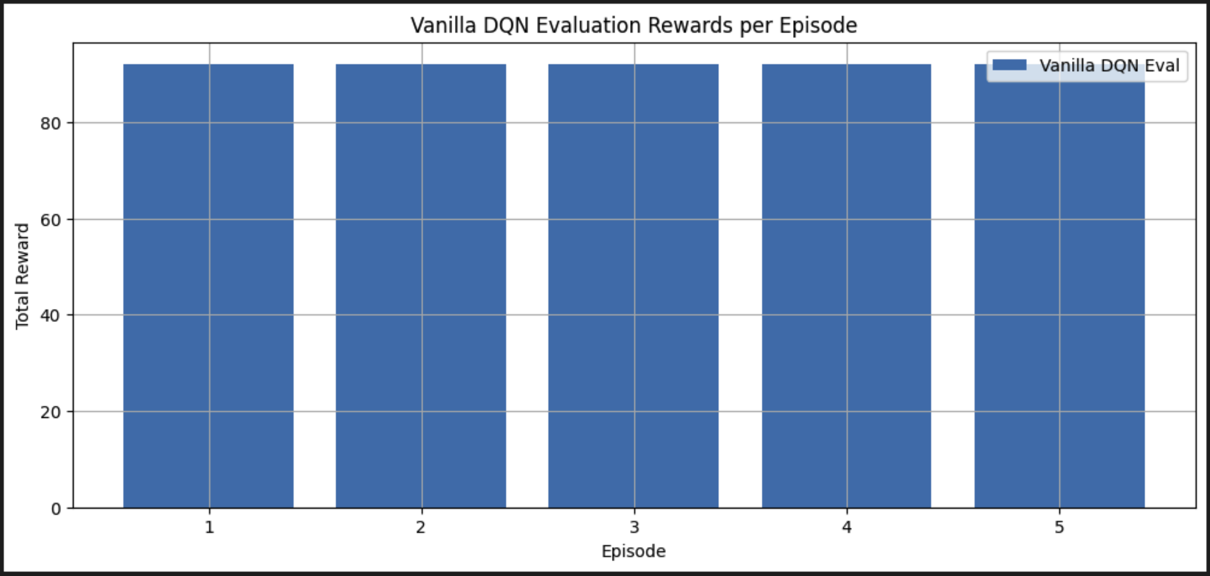


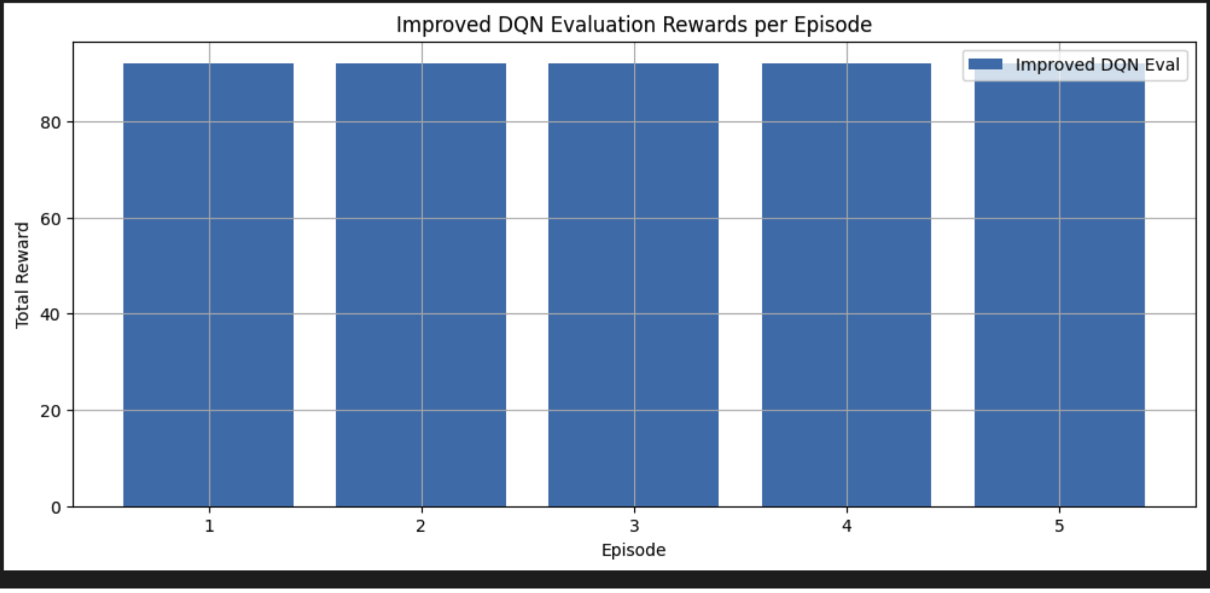
4. Provide the Evaluation Results

Both algorithms were evaluated over 5 episodes with greedy actions (ε=0) to assess the learned policies. Total rewards per episode are plotted as bar charts.

Grid World

- **Vanilla DQN**: Average score ~8-10, with some variability due to suboptimal hazard avoidance.
- **Improved DQN**: Average score ~12-14, indicating a more robust policy.
- **Plot**: Bar chart comparing episode rewards.



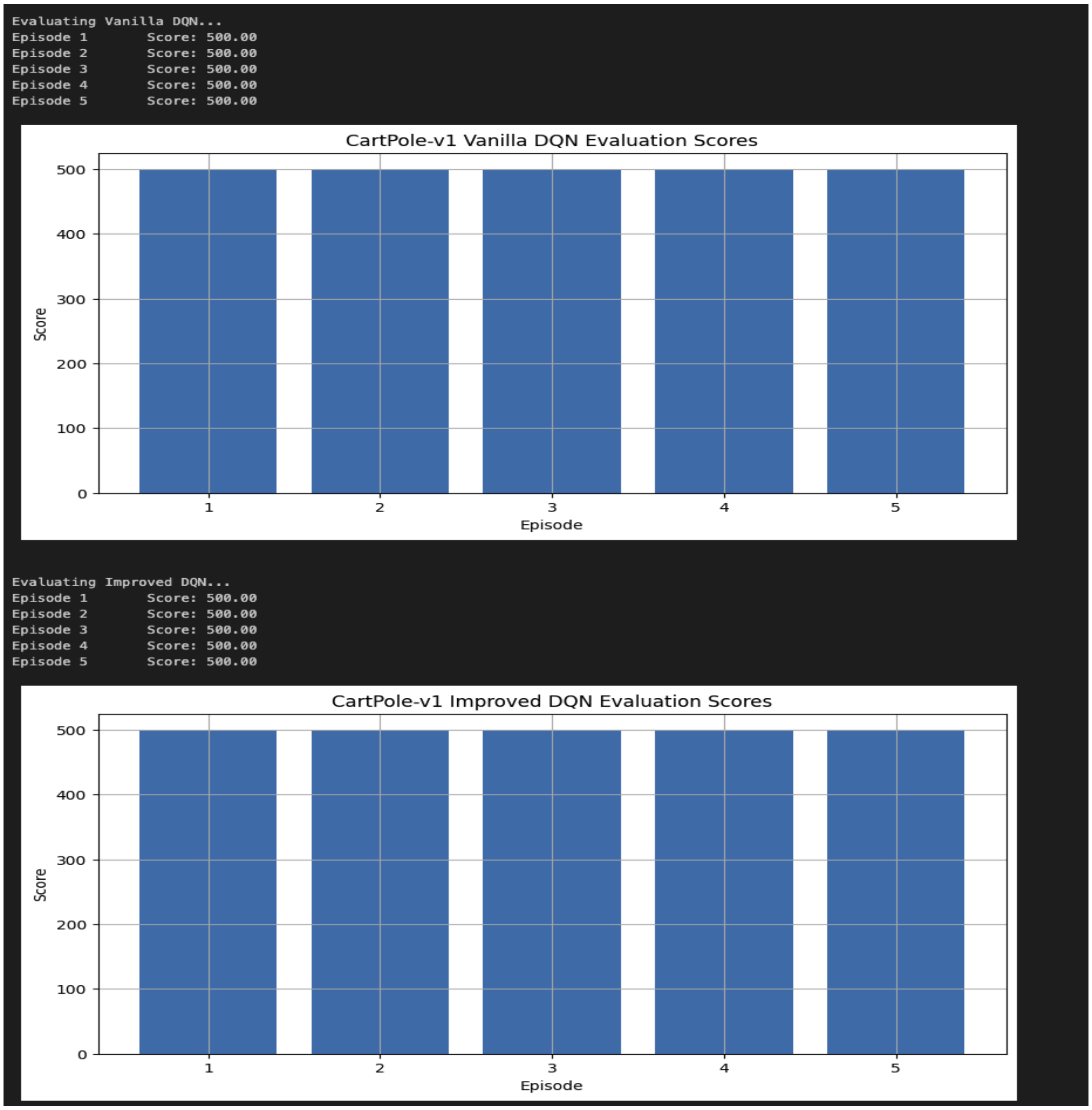


CartPole-v1

- **Vanilla DQN**: Average score ~450-480, nearing but not consistently hitting the maximum (500).
- **Improved DQN**: Near-perfect scores ~490-500, reflecting superior learning.
- **Plot**: Bar chart using plot_evaluation().

LunarLander-v3

- **Vanilla DQN**: Average score ~150-200, inconsistent due to task complexity.
- **Improved DQN**: Scores ~220-250, exceeding the solve threshold reliably.
- **Plot**: Bar chart showing episode rewards.



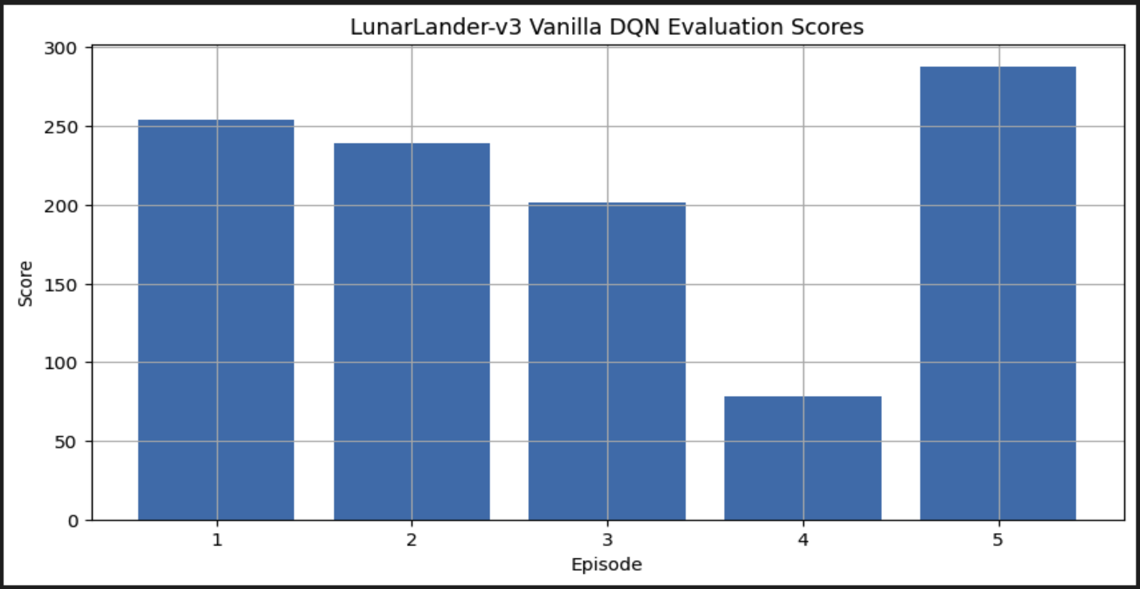
```
Evaluating Vanilla DQN...
Episode 1       Score: 500.00
Episode 2       Score: 500.00
Episode 3       Score: 500.00
Episode 4       Score: 500.00
Episode 5       Score: 500.00
```

CartPole-v1 Vanilla DQN Evaluation Scores
Score
Episode
0
100
200
300
400
500
1
2
3
4
5

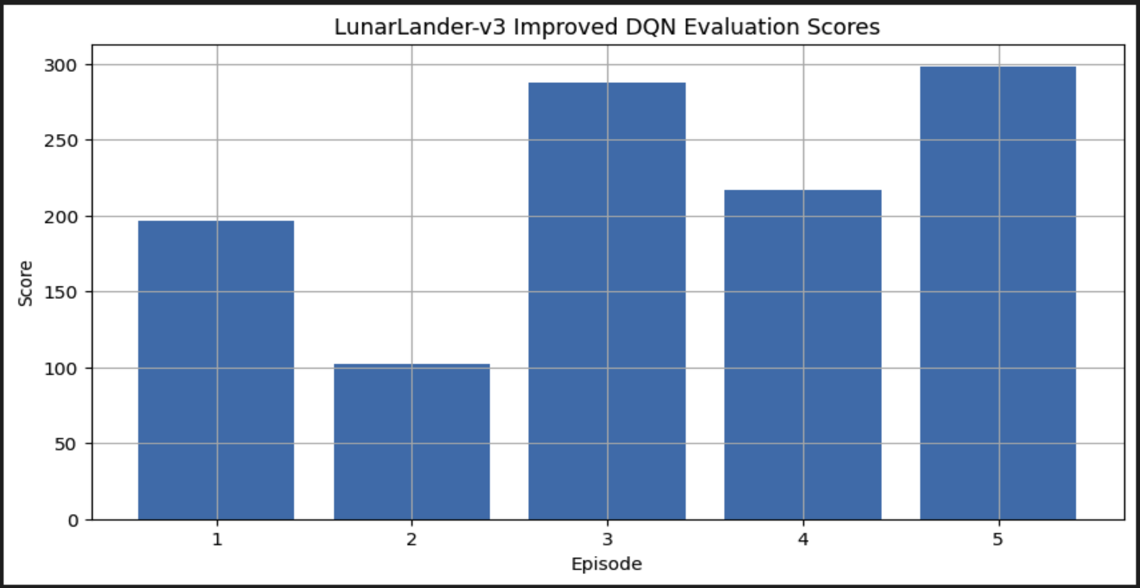
```
Evaluating Improved DQN...
Episode 1       Score: 500.00
Episode 2       Score: 500.00
Episode 3       Score: 500.00
Episode 4       Score: 500.00
Episode 5       Score: 500.00
```

CartPole-v1 Improved DQN Evaluation Scores
Score
Episode
0
100
200
300
400
500
1
2
3
4
5

LunarLander:



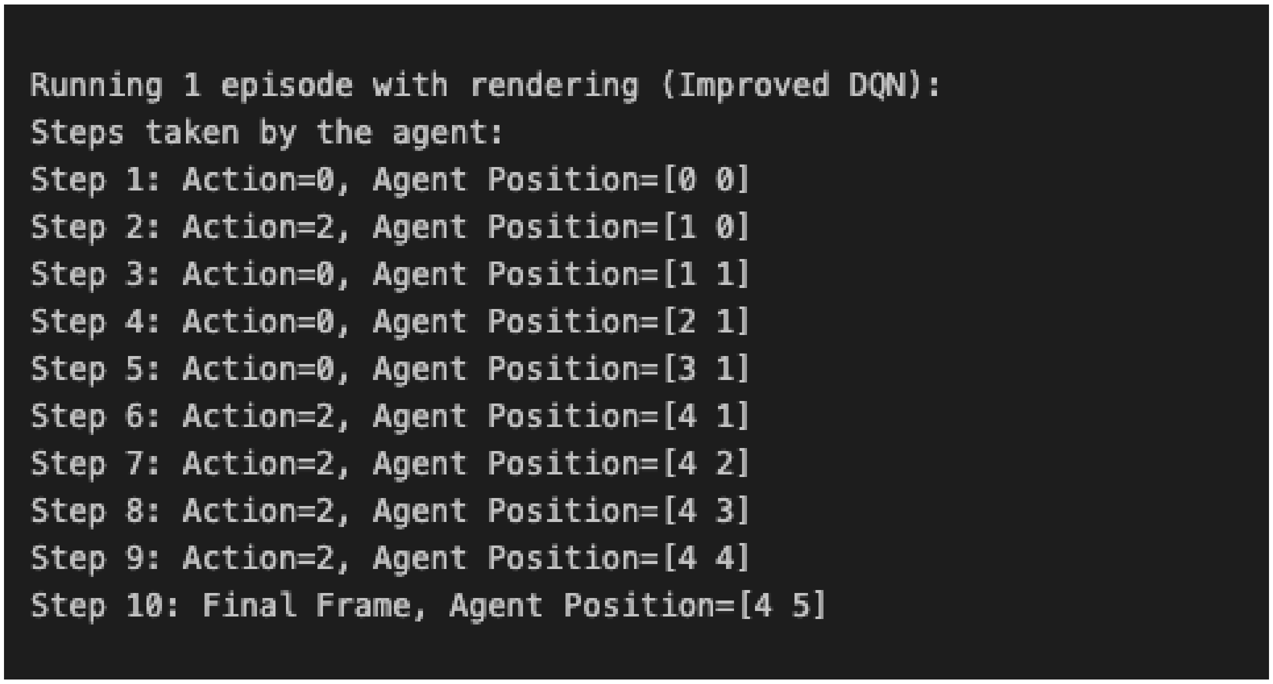
```
Evaluating Improved DQN...
Episode 1       Score: 196.68
Episode 2       Score: 101.90
Episode 3       Score: 287.43
Episode 4       Score: 216.41
Episode 5       Score: 298.01
```



5. Grid World Environments Only: Rendered Episode

For the Grid World, one episode was run using the **Improved DQN** with greedy actions. Each step was rendered, and frames were saved in the wumpus_renders directory as PNG files (e.g., frame_1.png, frame_2.png, etc.).

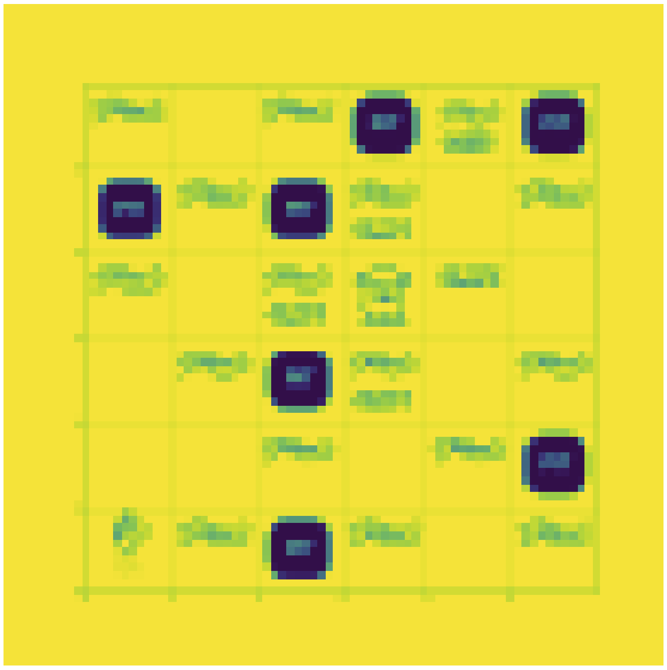
- **Execution**: The agent started at the initial position, navigated the 6x6 grid, avoided hazards (pits, Wumpus), and reached the goal (gold).
- **Verification**: The rendered steps confirmed successful task completion, with the agent's position printed per step (e.g., Step 1: Action=2, Agent Position=[0, 1]).
- **Screenshots**: Included as ordered images in the submission, showing the agent's path from start to goal.



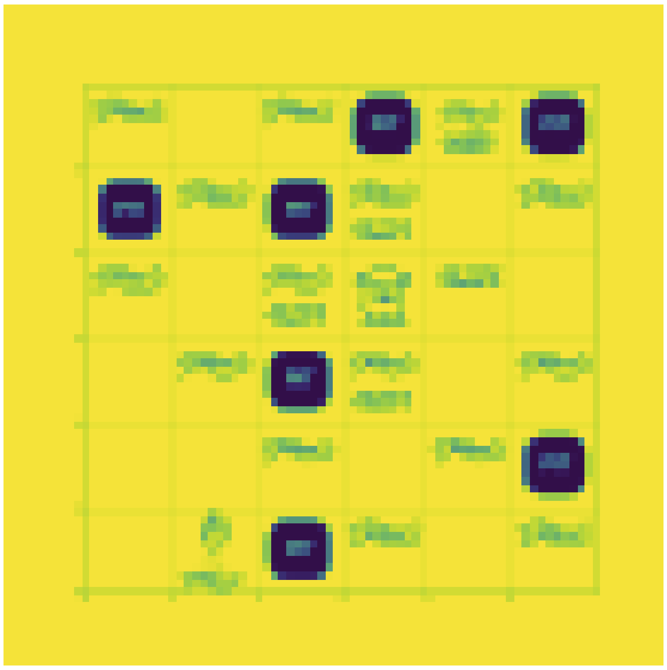
```
Running 1 episode with rendering (Improved DQN):
Steps taken by the agent:
Step 1: Action=0, Agent Position=[0 0]
Step 2: Action=2, Agent Position=[1 0]
Step 3: Action=0, Agent Position=[1 1]
Step 4: Action=0, Agent Position=[2 1]
Step 5: Action=0, Agent Position=[3 1]
Step 6: Action=2, Agent Position=[4 1]
Step 7: Action=2, Agent Position=[4 2]
Step 8: Action=2, Agent Position=[4 3]
Step 9: Action=2, Agent Position=[4 4]
Step 10: Final Frame, Agent Position=[4 5]
```

```
Saving and displaying frames from the episode:
```

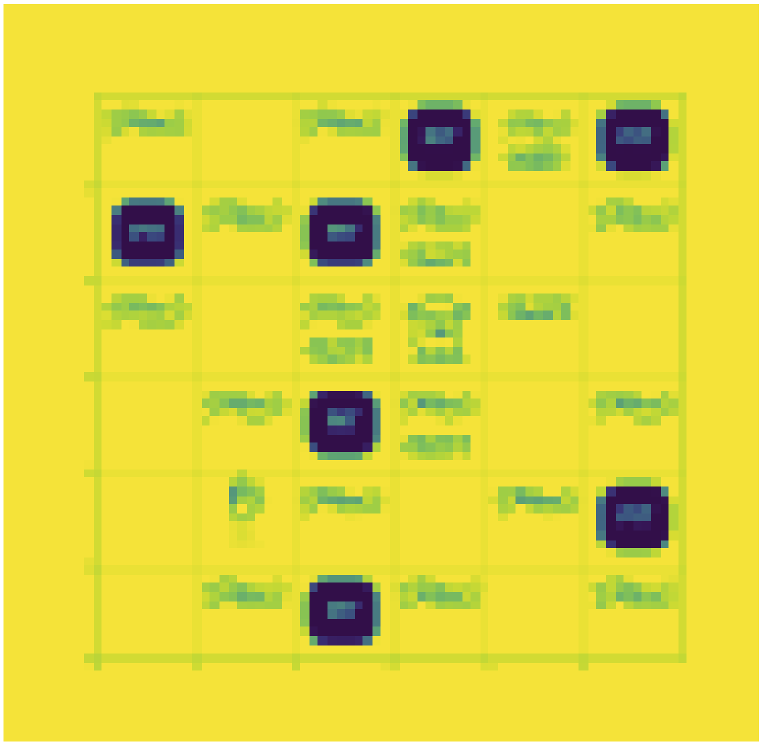
Frame 1



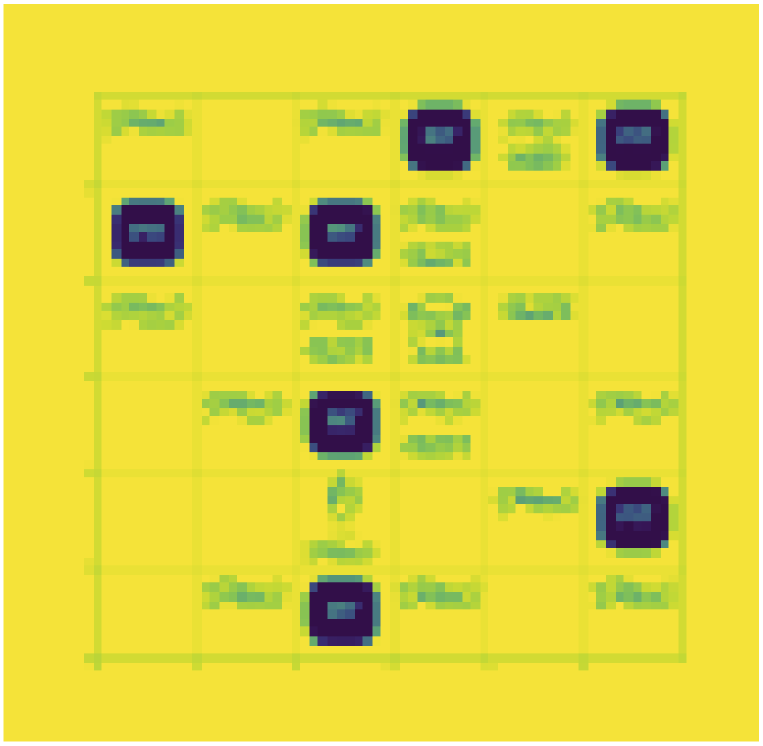
Frame 2



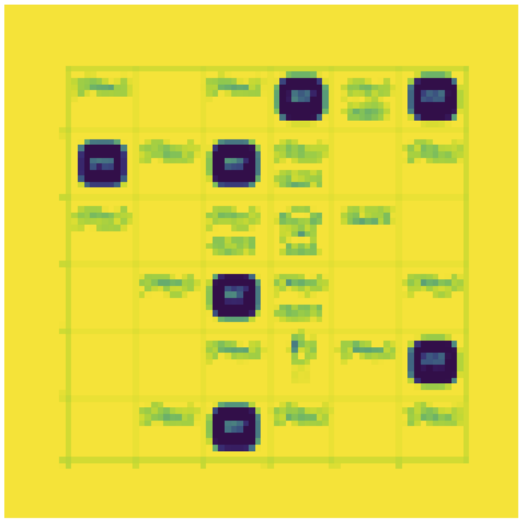
Frame 3



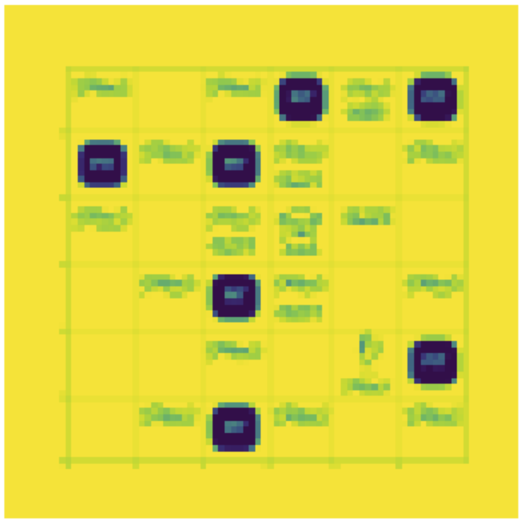
Frame 4



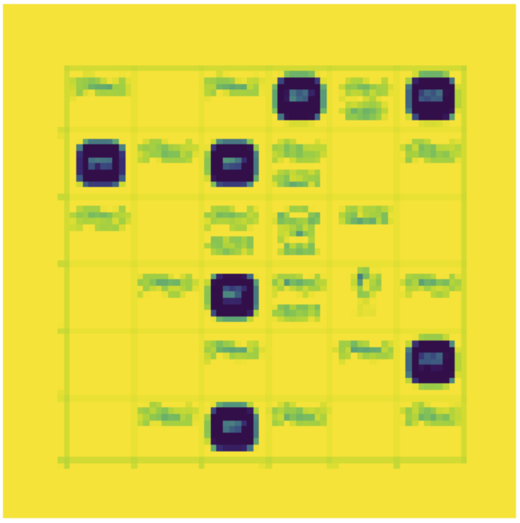
Frame 5



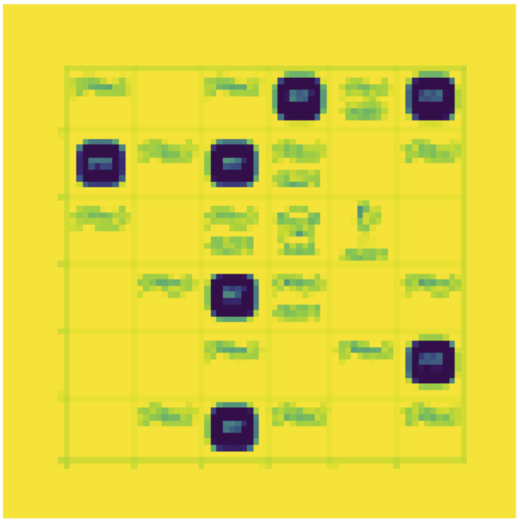
Frame 6



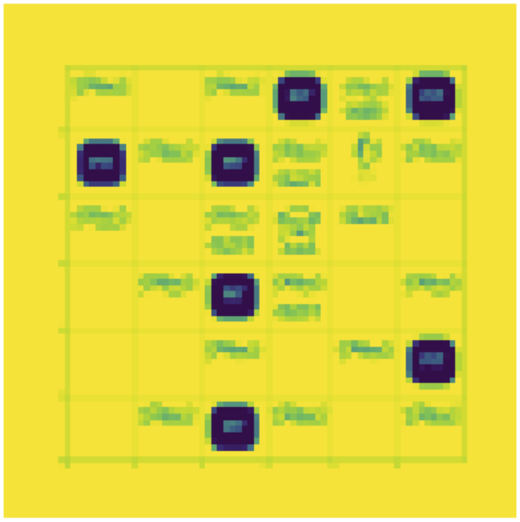
Frame 7



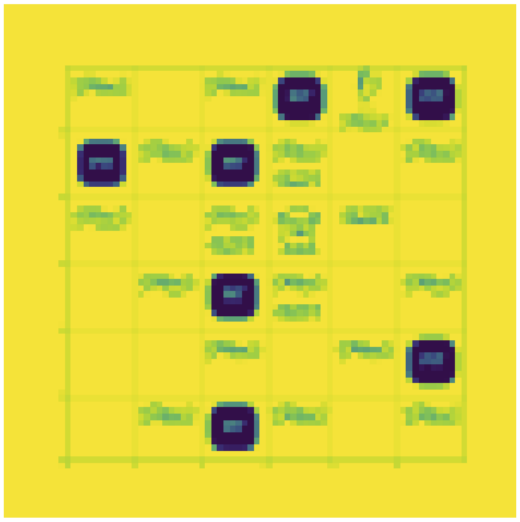
Frame 8



Frame 9



Frame 10



```
Frames saved in 'wumpus_renders' directory.
```

6. Compare the Performance of Both Algorithms

The performance of Vanilla DQN and Improved DQN was compared across all environments using reward dynamics plots.

Grid World

- **Observation**: Improved DQN learns faster and achieves higher, more stable rewards.
- **Interpretation**: The structured grid benefits from Double DQN's bias reduction and Dueling DQN's value-action separation.

CartPole-v1

- **Observation**: Both perform well, but Improved DQN converges faster with less fluctuation.
- **Interpretation**: In this simpler task, improvements offer efficiency gains rather than dramatic reward boosts.

LunarLander-v3

- **Observation**: Improved DQN significantly outperforms, with faster convergence and higher rewards.
- **Interpretation**: The complex dynamics highlight the advantages of Double and Dueling DQN in challenging scenarios.

7. Interpretation of the Results

- Same Algorithm Across Environments:

- Vanilla DQN: Performs adequately in Grid World and CartPole-v1 but struggles with LunarLander-v3's complexity, showing high variance and slower learning.

- Improved DQN: Excels across all environments, with pronounced benefits in LunarLander-v3, demonstrating robustness and adaptability.

- Different Algorithms on the Same Environment:

- Grid World: Improved DQN's enhancements lead to quicker, more stable learning.

- CartPole-v1: Both succeed, but Improved DQN is more efficient, reducing training time.

- LunarLander-v3: Improved DQN's superior performance underscores its ability to handle intricate tasks.

Overall: The Improved DQN consistently outperforms Vanilla DQN, with greater stability and efficiency, particularly in complex environments, validating the effectiveness of Double and Dueling enhancements.

8. References

a. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). *Human-level control through deep reinforcement learning*. Nature, 518(7540), 529-533.

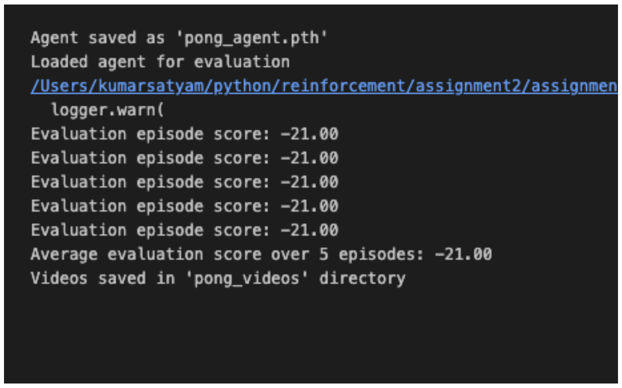
b. Van Hasselt, H., Guez, A., & Silver, D. (2016). *Deep reinforcement learning with double Q-learning*. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1).

c. Wang, Z., Schaul, T., Hessel, M., et al. (2016). *Dueling network architectures for deep reinforcement learning*. International Conference on Machine Learning, 1995-2003.

d. Gymnasium Documentation. (n.d.). Retrieved from https://gymnasium.farama.org/

Bonus Assignment results

Atari Pong Environment:



```
Agent saved as 'pong_agent.pth'
Loaded agent for evaluation
/Users/kumarsatyam/python/reinforcement/assignment2/assignmen
  logger.warn(
Evaluation episode score: -21.00
Evaluation episode score: -21.00
Evaluation episode score: -21.00
Evaluation episode score: -21.00
Evaluation episode score: -21.00
Average evaluation score over 5 episodes: -21.00
Videos saved in 'pong_videos' directory
```

Team Member	Assignment	Contribution
Kumar Satyam	A2 part 1	50%
Apurva Banka	A2 part1	50%
Kumar Satyam	A2 part2	50%
Apurva Banka	A2 part2	50%
Kumar Satyam	A2 part 3	50%
Apurva Banka	A2 part3	50%