Reinforcement learning

Uma Maheswara Reddy
UBIT: usannapu (50539298)

Shraddha Meduri Kishore
UBIT: smedurik(50560510)

Part 2:

1)Discuss the benefits of:

• Using experience replay in DQN and how its size can influence the results

- In DQN the experience replay works by learning and storing the past experiences and it also works by randomly sampling it helps to reduce the correlations between consecutive samples and improve the stability.
- It will give an advantage to the agent to learn both the past experiences and recent most experiences and it will ensure the use of knowledge and it will out change the smoothing in distribution of updates of the policies.
- While coming to the small buffers it will lead to the problem of overfitting by concentrating on the recent events and it will reducing the diversity and hindering generalization while coming to the large buffers it will lead to give the more diverse experiences and it will improve the learning and it may include the information that was outdated and that can be less relevant to the objectives which are current. And balancing the buffer size is more important to maximize the learning efficiency without the overfitting or diluting the relevant information

• Introducing the target network

- Target network in DQN will improve the training and stability by decoupling the calculation of target q value from networks frequent updates which it comes from online .

- It works alongside the Q network which is online and serves as a fixed reference point for several updates and it minimizes the oscillations and divergence in Q value estimates .The network is updated periodically by copying the network wrights which are from online and it will smoother and the learning will be more stable .This will prevents the fluctuations which are large and it will helps to stabilize the process of training

• Representing the Q function as qˆ(s, w)

- The representation of the q function as q(s,w) where the w stands for the parameters as weights and states of the neural network and it will reflect the approximation of the value function Q in the reinforcement learning. The goal of the process is learning to adjust the parameters such as weights and state and it accurately approximates the q value which is true and it will represent the future rewards which are expected for taking the actions specified in the state. To handle a large and complex state spaces which are using by a neural network to approximate the values of q instead of it relaying on the q tables which are traditional

---

2)Briefly describe 'CartPole-v1', the second complex environment, and the grid-world environment that you used (e.g. possible actions, states, agent, goal, rewards, etc). You can reuse related parts from your Assignment 1 report.

CartPole -V1:

Agent : In this Cartpole V1 agent is a cart that can have the ability to move left or right along a 1D track and it has pole attached to its top

States :The states position of the cartpole v1 is defined by the position of the cart and velocity of the cart and pole angle and the pole angular velocity

Actions: Left or Right

Goal : The main of objective or goal of the cart is to balance the pole upright on the cart which is moving as long as possible

Reward : The reward of +1 is given for every step the pole remains balanced. The episode will be terminated if the pole falls beyond the certain angle or if the cart moves out of bounds

Cliff Walking Environment:

Agent : In this a walker will navigate through a grid kind of environment with a cliff that are dangerous

States : Each states will represents the agents positions on the grid Actions : Agent can move up , down , left or right

Goal : It will reach the goal state while avoiding falling off from the cliff

Rewards: Positive rewards are given for reaching the goal, and penalties (large negative rewards) are applied for falling off the cliff.

Warehouse Robot Environment:

Agent: The robot operating will be in 6 * 6 grid warehouse environment

States : Every state will represents the robots current position and it will tells us where it will hold the item or not

Possible actions : The robot has the actions up , down , left , right , pickup and drop off items.

Goal : The main of the robot need to pick up the items from specific locations and it will deliver then it drop at the particular item at drop off points and it will navigate around obstacles

Rewards : The positive rewards for pickup and dropping the items successfully and it will get a negative reward for the colliding actions.

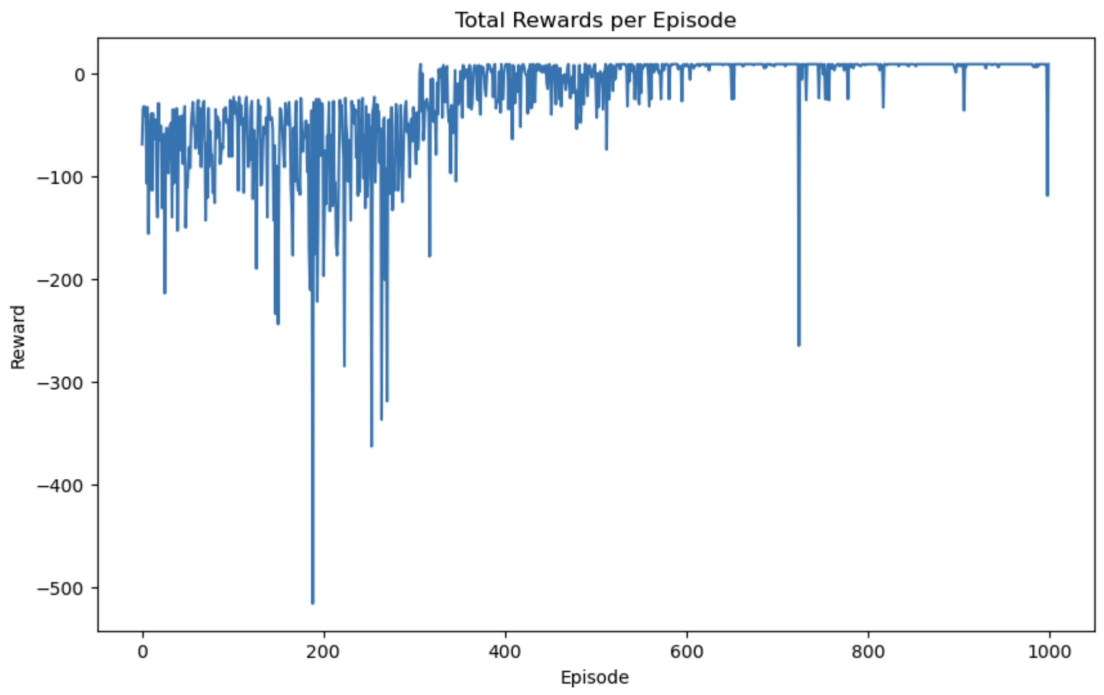
---

3) Show and discuss your results after applying your DQN implementation on the three environments. Plots should include epsilon decay and the total reward per episode.

WareHouse Robot

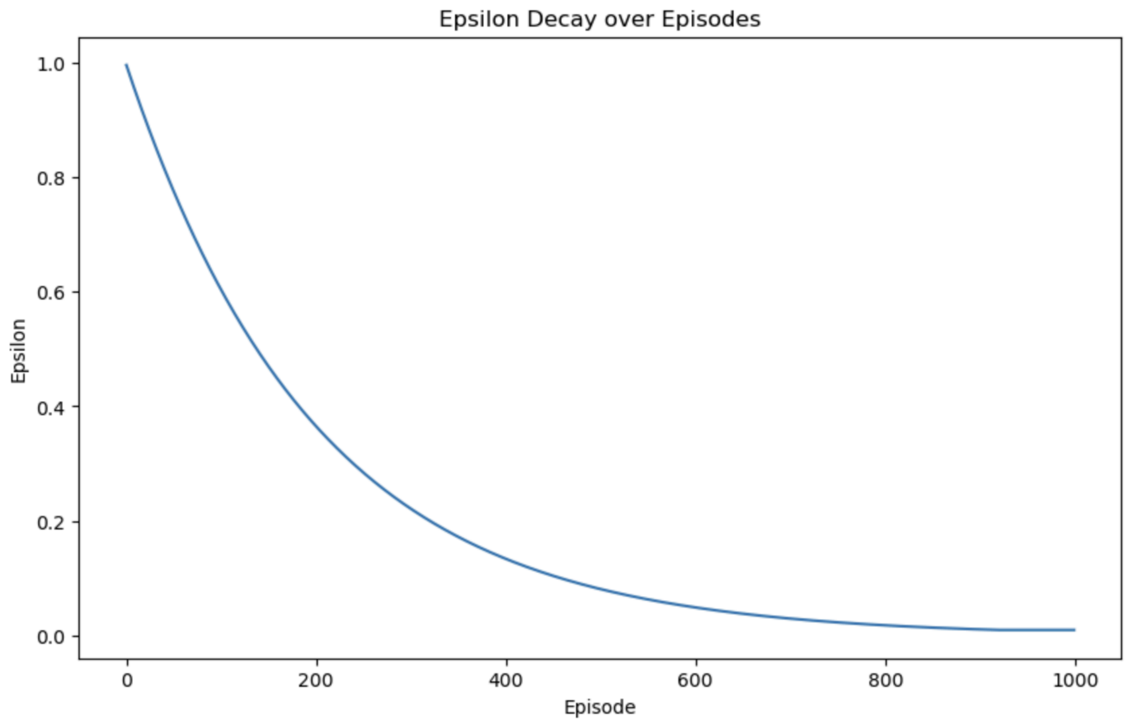
Total reward per episode

The below graph plot is the total rewards per episode and it will demonstrate the learning progress of the agent of DQN . Meanwhile in early episodes the agent is experiences the variables which are the rewards with the frequent large negative values and it will indicate exploration and suboptimal actions. As the training process particularly after the episode of 300 the rewards are stabilized and it will reflect decisions which are improvised and learning . After that the rewards plateau at the higher values and it is showing the agent has likely converged on optimal policy. Occasional drops may still occur due to exploration as epsilon decays. Overall, the agent becomes more consistent in maximizing rewards as training advances.



Epsilon Decay:

The decay of the epsilon graph follows the expected exponential decline across all environments. This indicates that agent which will start from a high level of exploration and trying out different actions and gradually will shifts to rely more on the learned policy on the process of the training progress and this will effectively balances the exploration and exploitation throught the process of the learning

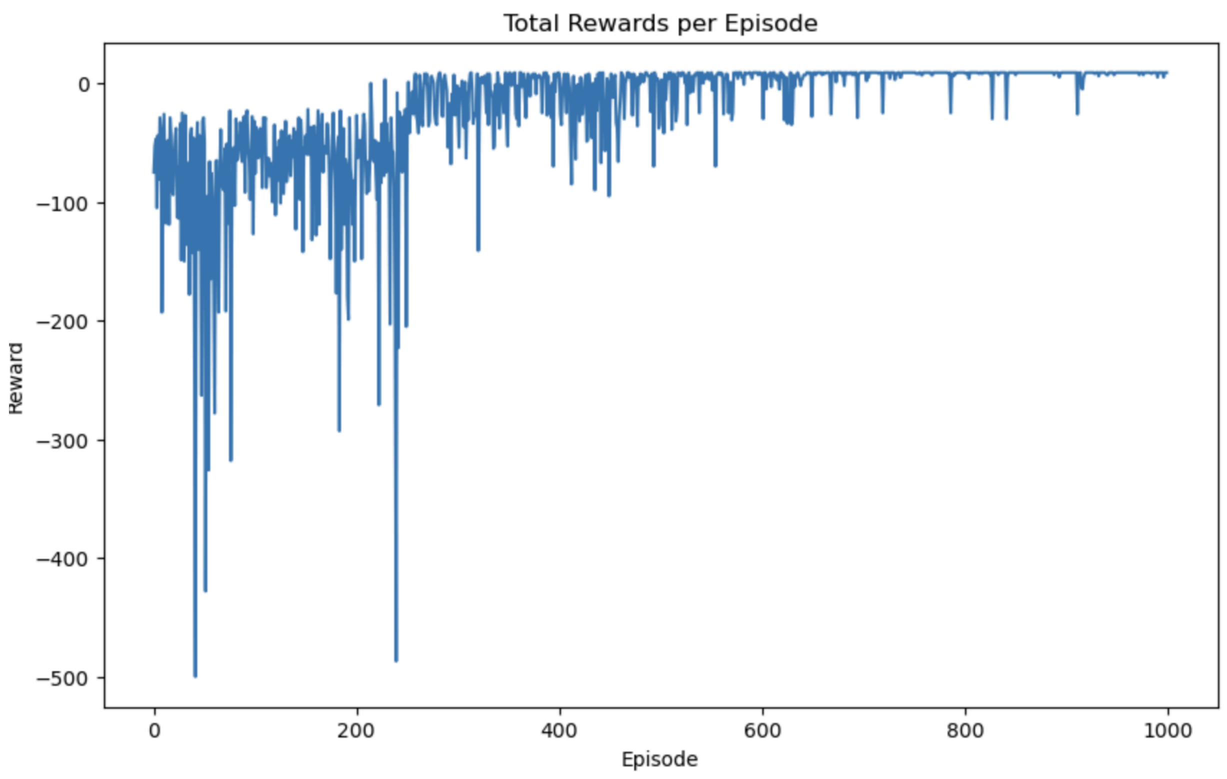


Cliffwalking

Total reward per episode

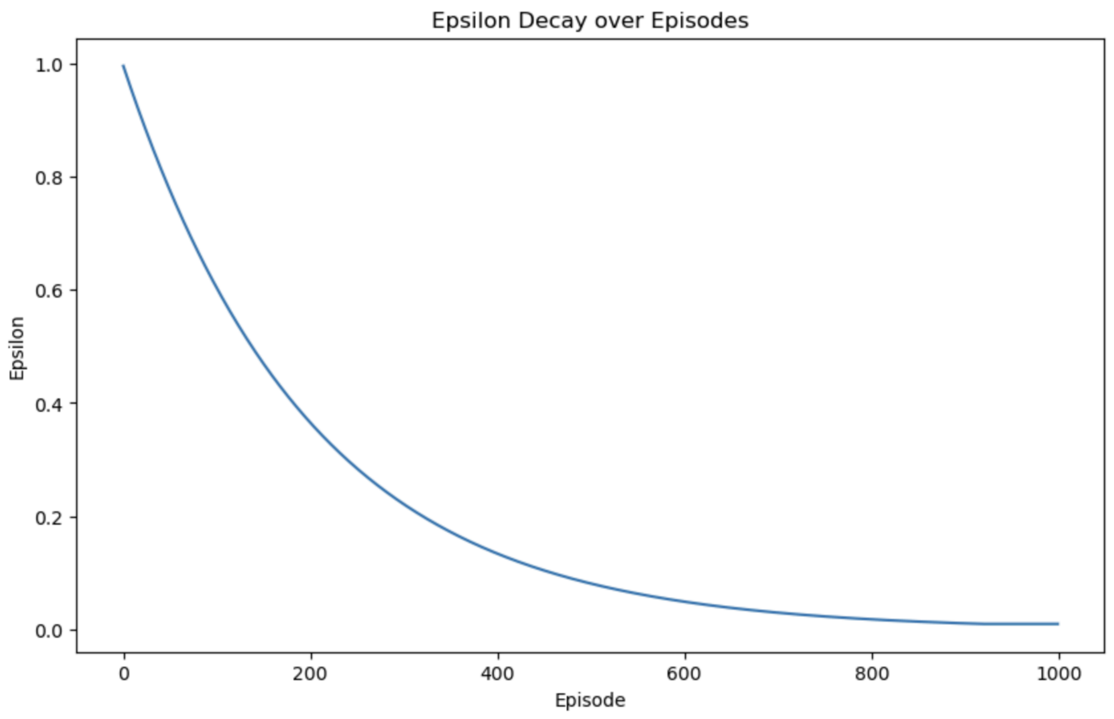
The below graph will show the total reward per episode ans with the early episodes displaying the high volatility and having the large negative rewards and over the time the rewards will stabilize and it will approach zero and it will indicate the improved performance.

```
Episode: 1000, Reward: 9, Epsilon: 0.01
```



Epsilon Decay :

The second image illustrates the epsilon decay starting high and decreasing smoothly to 0.01 and it will reflect a shift from exploration to exploitation and this balance will helps the agent learn and effective policies while exploring the environment adequately .

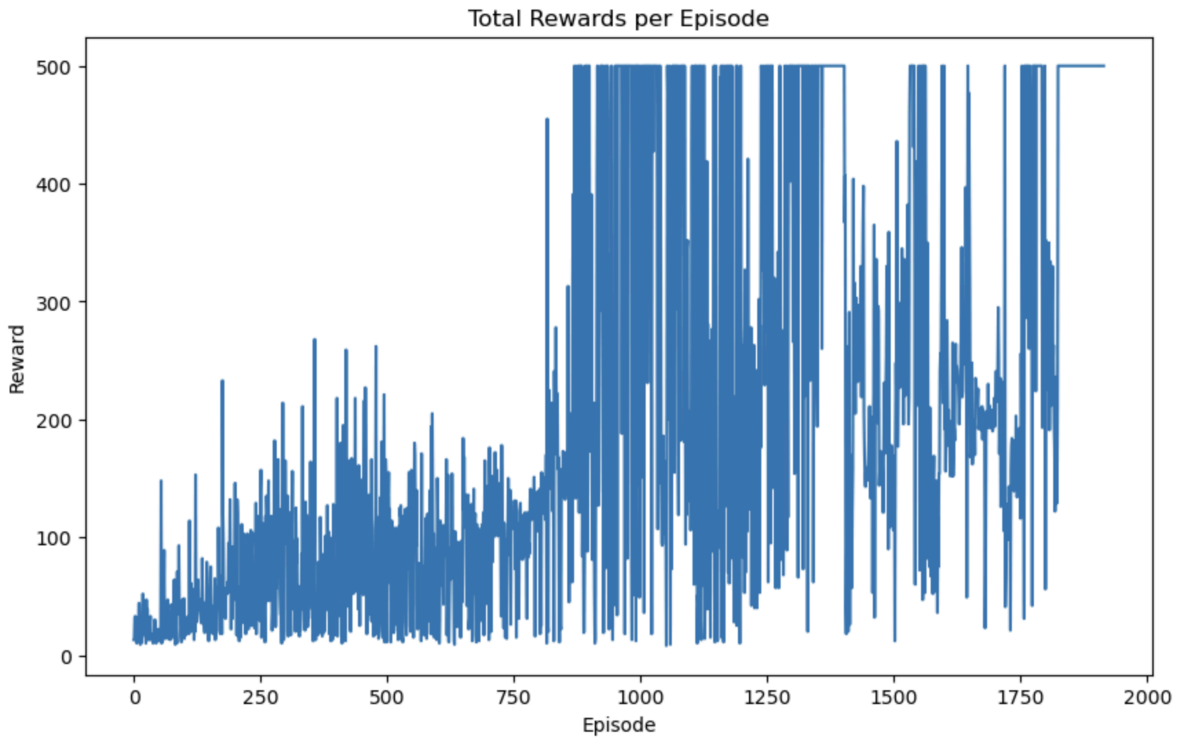


CARTPOLE - V1

Total reward per episode

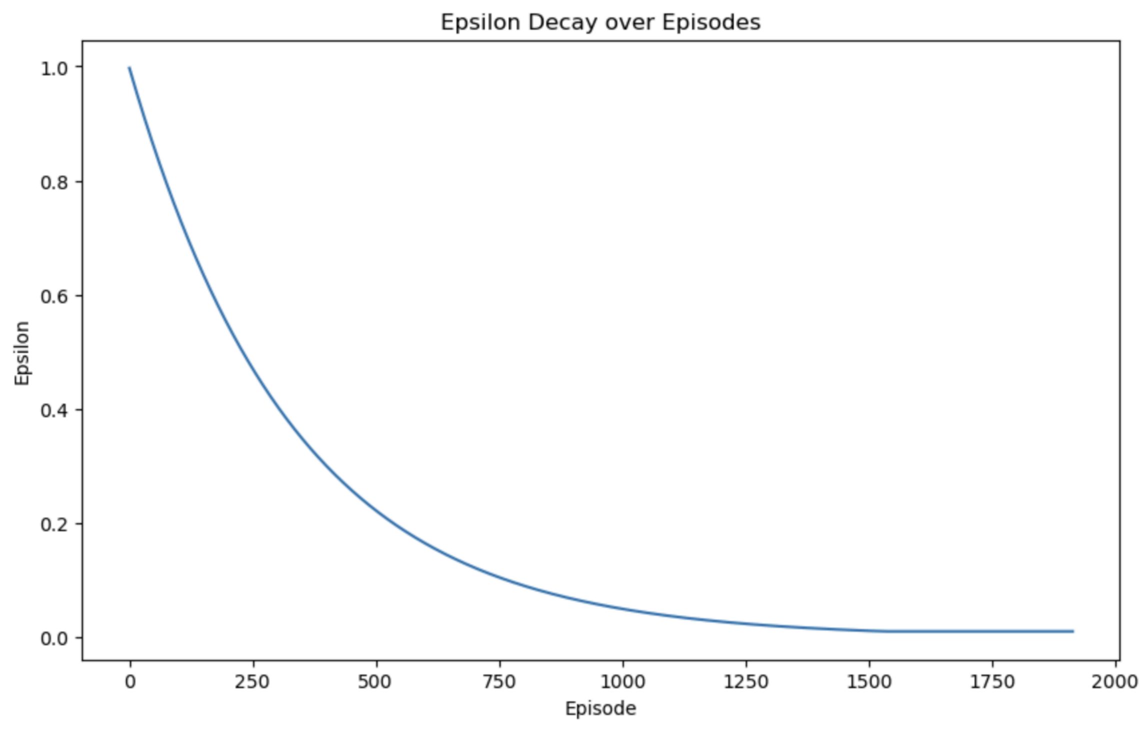
The agent's performance fluctuates and improves gradually. Around episode 1000, there's a noticeable increase in rewards, indicating learning progress. After this point, the agent frequently achieves higher rewards, suggesting it has learned effective strategies. Early stopping occurred at episode 1915 with an average reward of 472.02, showing stable performance.

```
Early stopping at episode 1915 with average reward 472.02
```



Epsilon Decay:

This graph will be illustrates a curve that begins at 1 and gradually decreases toward 0 over time and this will signifies the agent transitions from randomly explore the environ to exploit the knowledge it has acquire through learned policy

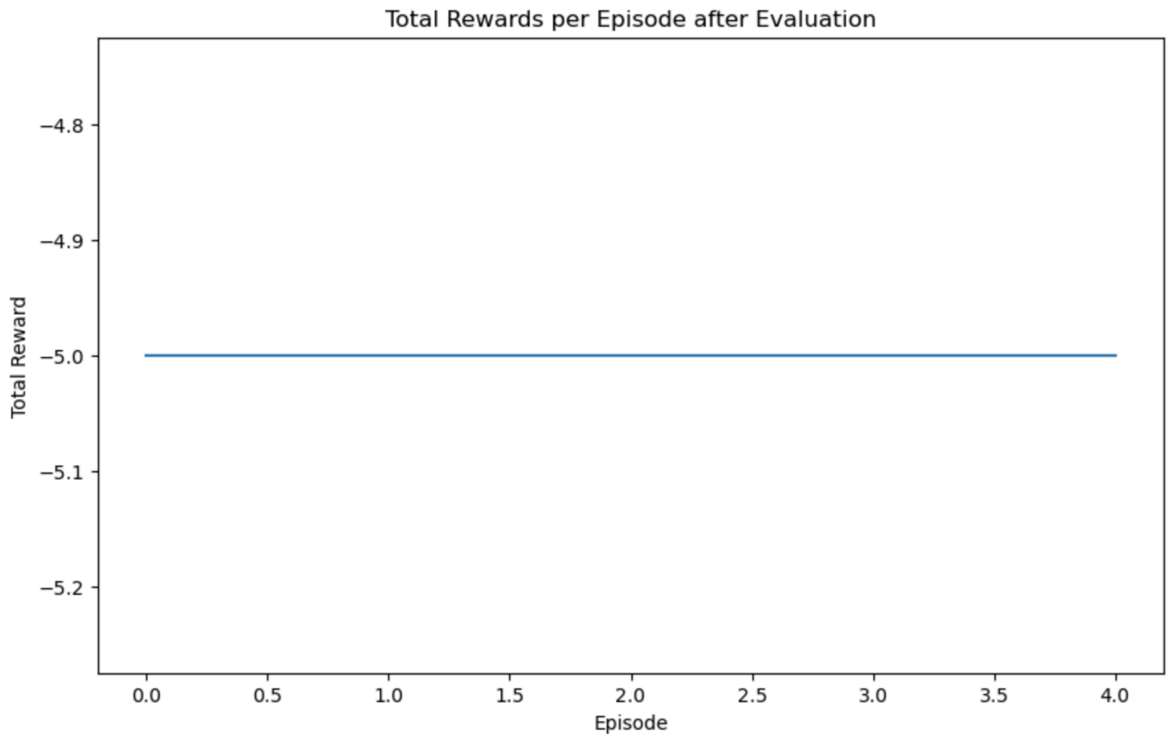


---

4) Provide the evaluation results. Run your agent on the three environments for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

Ware House Robot

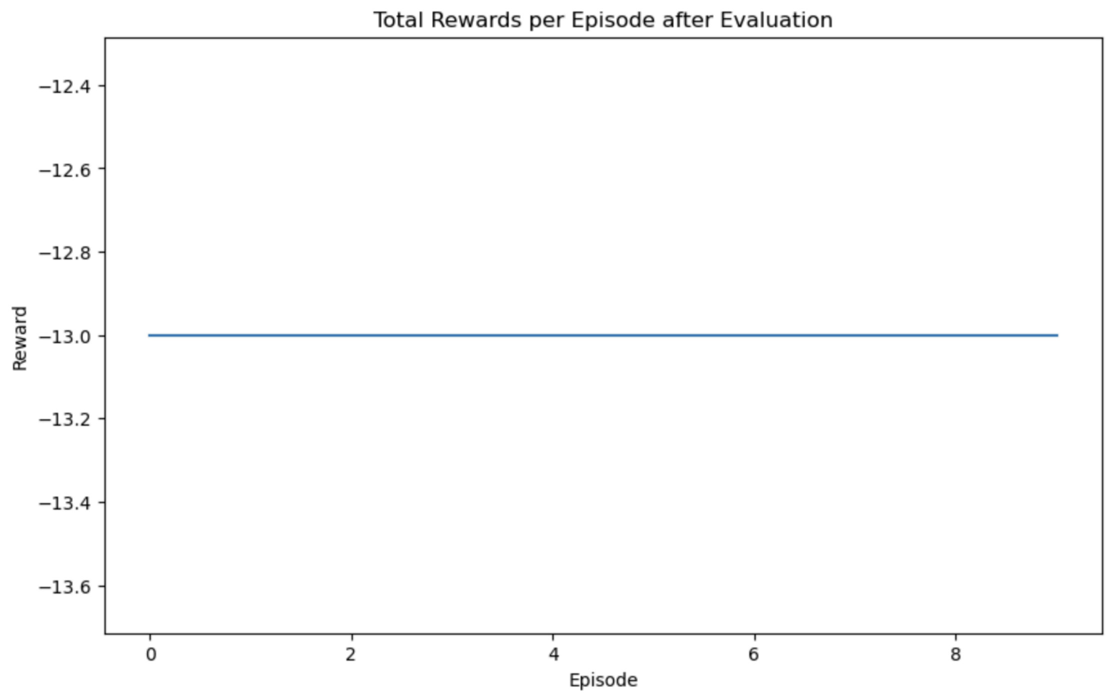
The agent's performance will fluctuate during the episodes early and it will stabilize the learning progress with fewer sharp drops and after the 500 episodes the agent can earn high rewards which are consistent and variability remains. During the evaluation the agent received a constant reward of -5 over five episodes indicating limited improvement or challenges in adapting to the test environment.the standard epsilon decay approach likely contributed to the agent's gradual shift from exploration to exploitation. This is for custom environment or warehouse robot.



```
Average Reward over 5 episodes: −5.00
```

Cliffwalking

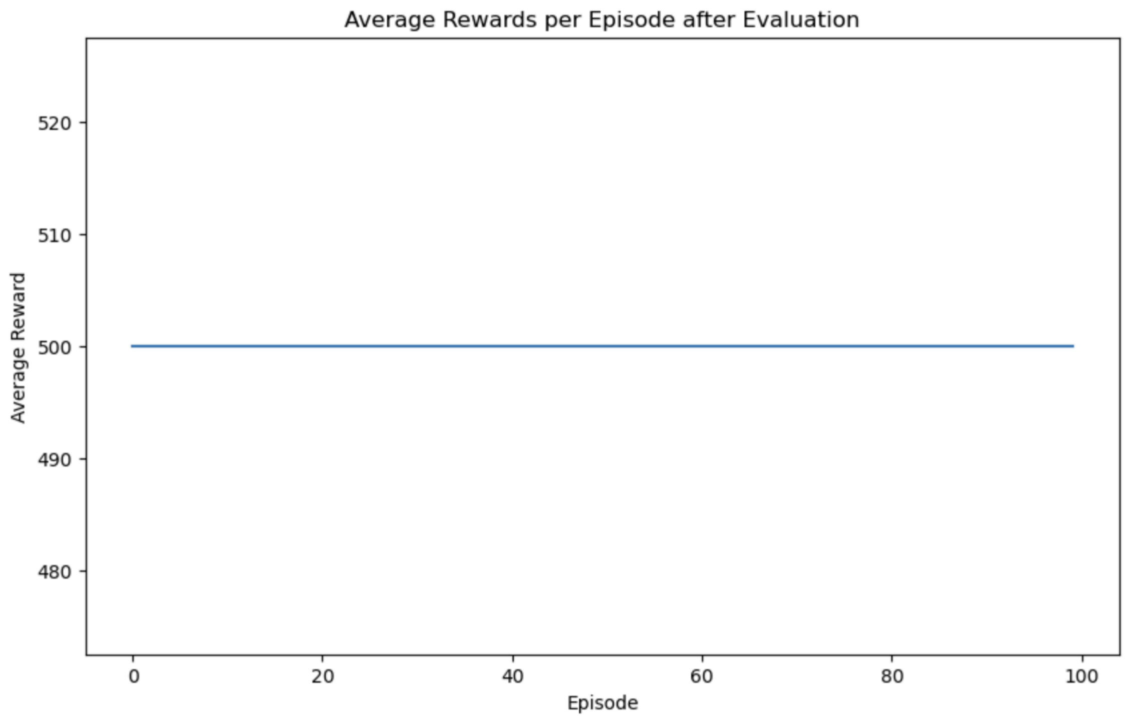
This graph represents the total rewards per episode during an evaluation phase. The reward remains constant at approximately -13 across all 10 evaluation episodes, suggesting that the agent's performance is stable. The average reward over these episodes is recorded as -13.00, which is the greedy action



```
Average Reward over 10 Evaluation Episodes: −13.00
```

Cartpole

This graph shows the average reward per episode across 100 evaluation episodes. The reward remains consistently at 500, indicating that the agent has achieved optimal. The stable line suggests that the agent's learned policy performs reliably and achieves maximum reward in each episode.



```
Overall Average Reward over 100 Evaluation Episodes: 500.00
```

---

5)Provide your interpretation of the results. E.g. how the DQN algorithm behaves on different envs.

- DQN setup allows the robot to learn optimal navigation strategies, item handling, and goal completion actions. Through repeated episodes, the DQN can effectively help the robot learn high-reward paths, avoid obstacles, and perform tasks efficiently. The provided hyperparameters and training loop allow the robot to progressively improve its behavior, as seen in the graphs from previous evaluations, where the rewards stabilize at optimal levels, reflecting a successfully learned policy.

CartPole-V1:

Training: The Deep Q-Network (DQN) algorithm is maintaining a pole's balance on a cart by shifting the cart to the left or right in the CartPole environment. The increasing rewards per episode reflect the agent's learning to balance the pole for greater lengths of time during training. The rewards plot has a consistent rising trend throughout the episodes, suggesting that the agent's performance is becoming better with time.

Epsilon Rate: As training goes on, the epsilon rate, which controls the degree of exploration, progressively drops. The Epsilon Rate Decay figure illustrates this tendency. The agent moves from exploring to using the learnt tactics more as the epsilon number decreases.

Evaluation: The model undergoes five episodes of testing after the training phase is over. The trained agent continuously receives high rewards, indicating that it has successfully learnt to balance the pole, according to the Total Rewards per Episode plot after evaluation.

Cliff Walking :

Training : The growing total rewards every episode should be a reflection of the DQN algorithm's effectiveness in the CliffWalking environment during training. Due to the agent's random activities while exploring the state space, the rewards will initially fluctuate dramatically as it learns. However, the overall rewards should gradually rise as the agent gains experience and begins to comprehend the best way to traverse the grid. The agent is effectively learning to avoid falling into the cliff area while maximizing its score by achieving the goal, as evidenced by this upward trend in incentives.

Epsilon Rate: The epsilon rate is crucial for balancing the agent's exploration and exploitation. It starts at a high value (1.0), allowing the agent to explore a wide range of actions. As the epsilon rate decreases, the agent gains confidence in its learned strategies and begins to exploit them more frequently. If the epsilon decay is well-tuned, we can expect to see the epsilon rate gradually decrease and stabilize at a lower bound, such as 0.01. This shift indicates that the agent is becoming more strategic in its actions, using the knowledge it has gained during training to maximize rewards while still occasionally exploring new actions to refine its policy further.

Evaluation: After training, evaluating the DQN agent across multiple episodes should show how well it navigates the CliffWalking environment. If the total rewards per episode during evaluation are consistently high, it suggests that the agent has learned a solid strategy for reaching the goal without falling off the cliff. A stable and elevated total reward indicates that the agent has effectively balanced its path toward the goal while avoiding risky actions.

6)Include all the references that have been used to complete this part

https://paperswithcode.com/method/dqn
https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
https://medium.com/@shruti.dhumne/deep-q-network-dqn-90e1a8799871
https://www.gymlibrary.dev/environments/classic_control/cart_pole/
https://towardsdatascience.com/reinforcement-learning-cliff-walking-implementation-e40ce98418d4
https://www.gymlibrary.dev/environments/toy_text/cliff_walking/
https://gymnasium.farama.org/environments/classic_control/cart_pole/
https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
https://medium.com/@hkabhi916/mastering-deep-q-learning-with-pytorch-a-comprehensive-guide-a7e690d644fc
https://wandb.ai/safijari/dqn-tutorial/reports/Deep-Q-Networks-DQN-With-the-Cartpole-Environment--Vmlldzo4MDc2MQ

Part 3:

1. Discuss the algorithm you implemented.

The Double Deep Q-Network , an improvement on the classic Deep Q-Network , is the approach we used to solve the problem of overestimation bias in Q-value estimates. The following are the main elements of the implementation:

DQNetwork Class: This class defines the neural network architecture for approximating Q-values. It consists of three fully connected layers with ReLU activation functions, transforming the input state dimensions to output Q-values corresponding to possible actions.

Experience Replay: The ExperienceReplay class manages the storage and sampling of experience tuples state, action, reward, next state, done to break correlations in the training data. This allows for more stable learning by providing a diverse set of experiences for the training process.

Epsilon-Greedy Policy: The algorithm employs an epsilon-greedy strategy for action selection, balancing exploration (choosing random actions) and exploitation (selecting the best-known action). Epsilon starts at a high value and decays over episodes, reducing the exploration rate as the training progresses.

Training Loop: During training, the algorithm iterates through episodes, collecting rewards while interacting with the environment. For each step, it chooses an action based on the current state and updates the replay buffer with the resulting experience.

Double DQN Update: The algorithm uses two networks—an online Q-network and a target Q-network. The online network selects actions, while the target network evaluates the value of those actions. This reduces the overestimation bias of the Q-values by ensuring that the action value estimation is based on a stable target, improving the accuracy of the learning process.

Target Network Update: Periodically, the weights of the target network are updated to match the online network, enhancing stability in the learning process.

2. What is the main improvement over the vanilla DQN?

The main improvement of the Double Deep Q-Network over the vanilla Deep Q-Network lies in addressing the overestimation bias commonly associated with Q-learning.

Decoupling Action Selection and Evaluation: In vanilla DQN, the same network is used for both selecting the action to take and for estimating the Q-value of that action. This can lead to overestimation because the network might select an action with a high Q-value due to inherent noise in the Q-value estimates. In Double DQN, two networks are utilized: the **online Q-network** and the **target Q-network**. This separation reduces the overestimation bias since the target network evaluates the action chosen by the online network.

Stability and Convergence: By employing two distinct networks and periodically updating the target network with the online network's weights, Double DQN promotes stability and convergence during training, which dictates how often the target network is synchronized with the online network. This mechanism allows the target values to remain stable over multiple training steps, improving the overall learning process.

Experience Replay: Like vanilla DQN, Double DQN utilizes an experience replay buffer to sample experiences. However, the benefit of this combined with the decoupled action selection and evaluation ensures that the network learns from a diverse set of experiences, further enhancing training stability.

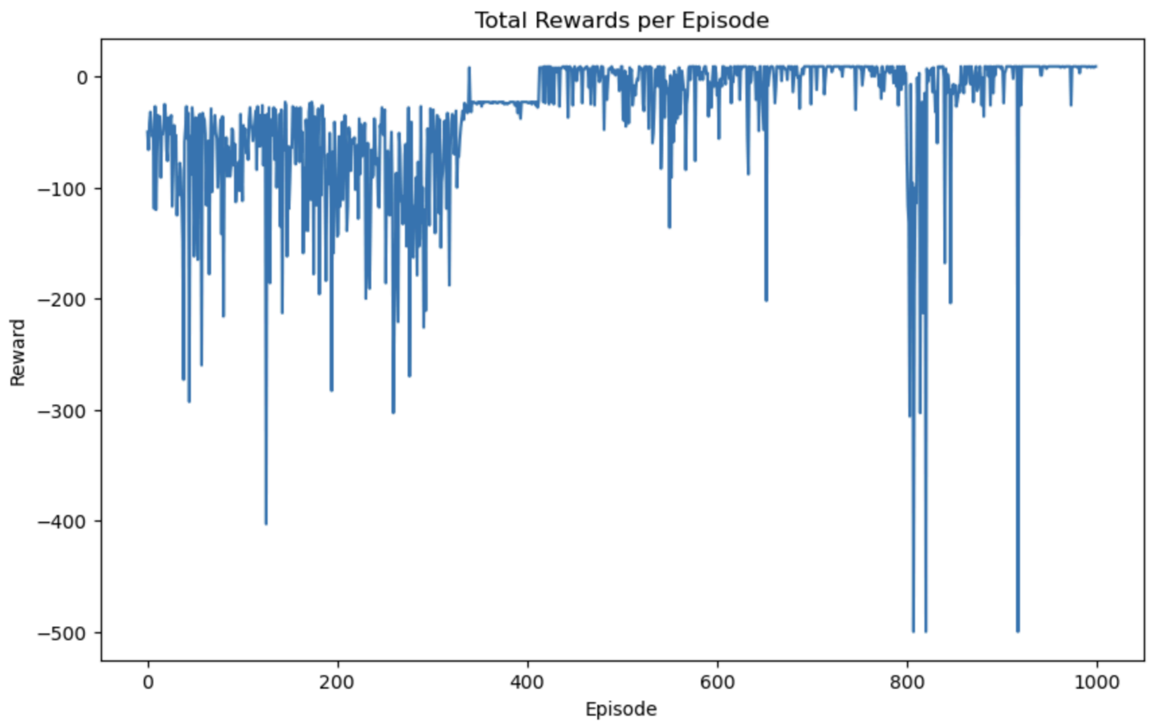
---

3) Show and discuss your results after applying your two algorithms implementation on the environment. Plots should include epsilon decay and the total reward per episode.

Warehouse robot

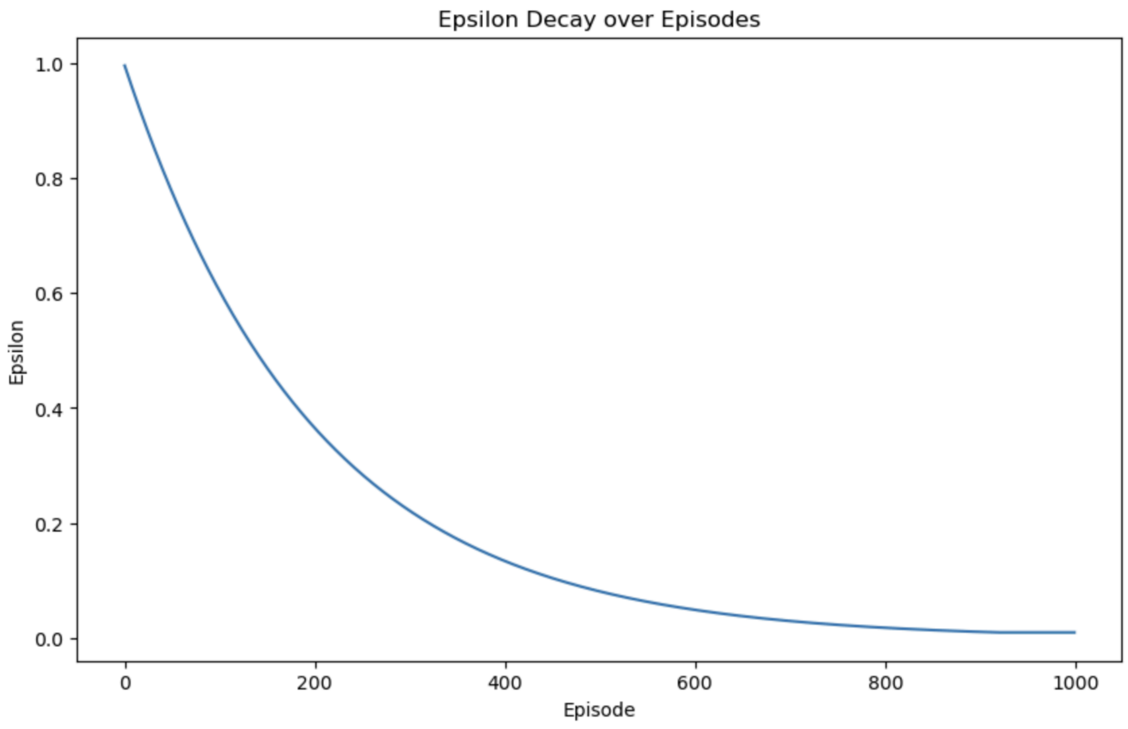
Total rewards per episode

This plot tracks the total rewards the agent earned in each of 1000 episodes. In the beginning, the agent’s performance is all over the place, with mostly negative rewards as it’s still figuring things out. Around episode 300, things start to settle down, showing it’s learned some useful strategies. There are still occasional big drops, meaning the agent sometimes hits tough spots or tries out new moves. By the end, the rewards are more stable and closer to zero, which suggests it’s getting better but hasn’t totally mastered the task yet.



Epsilon decay

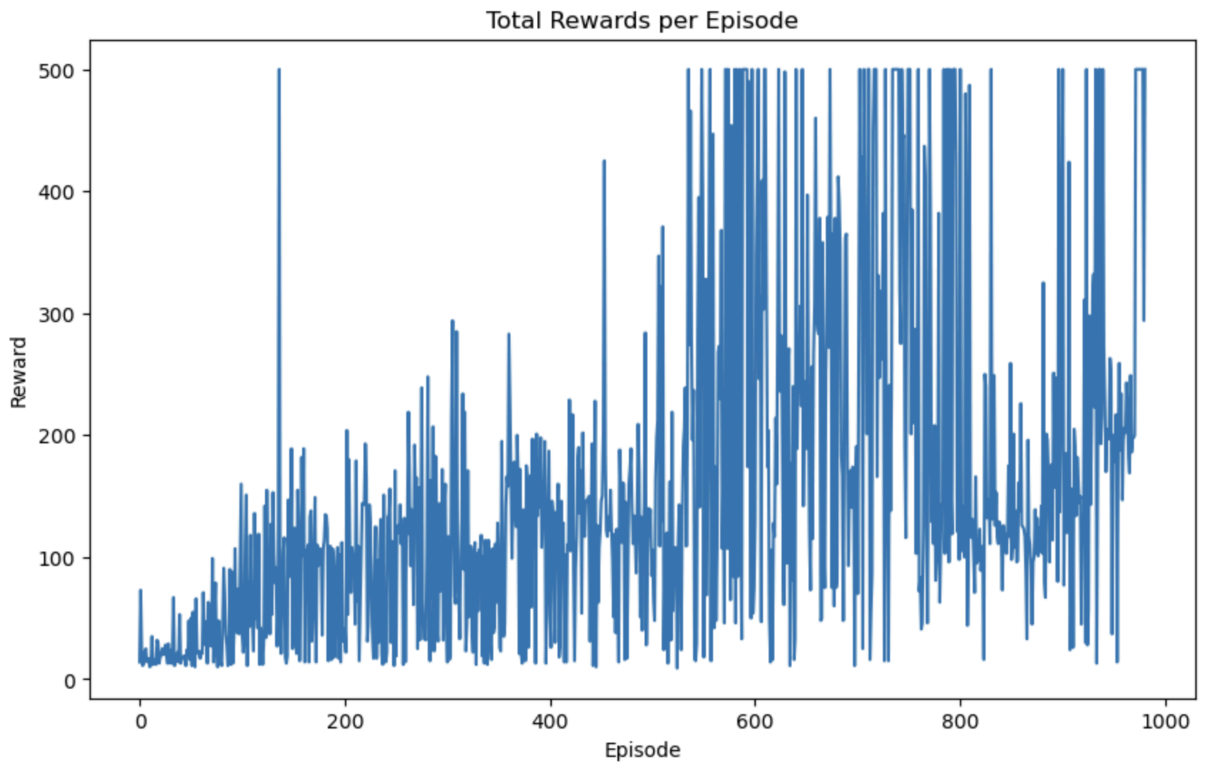
epsilon decreases over 1000 episodes. Epsilon controls how often the agent explores by picking random actions instead of following what it's learned. At the start, epsilon is high, meaning lots of exploration, but it gradually drops as the agent gets more confident in its choices. By the end, epsilon is close to zero, so the agent mostly relies on what it's learned instead of exploring randomly. This helps it settle into a more stable strategy.



Cartpole

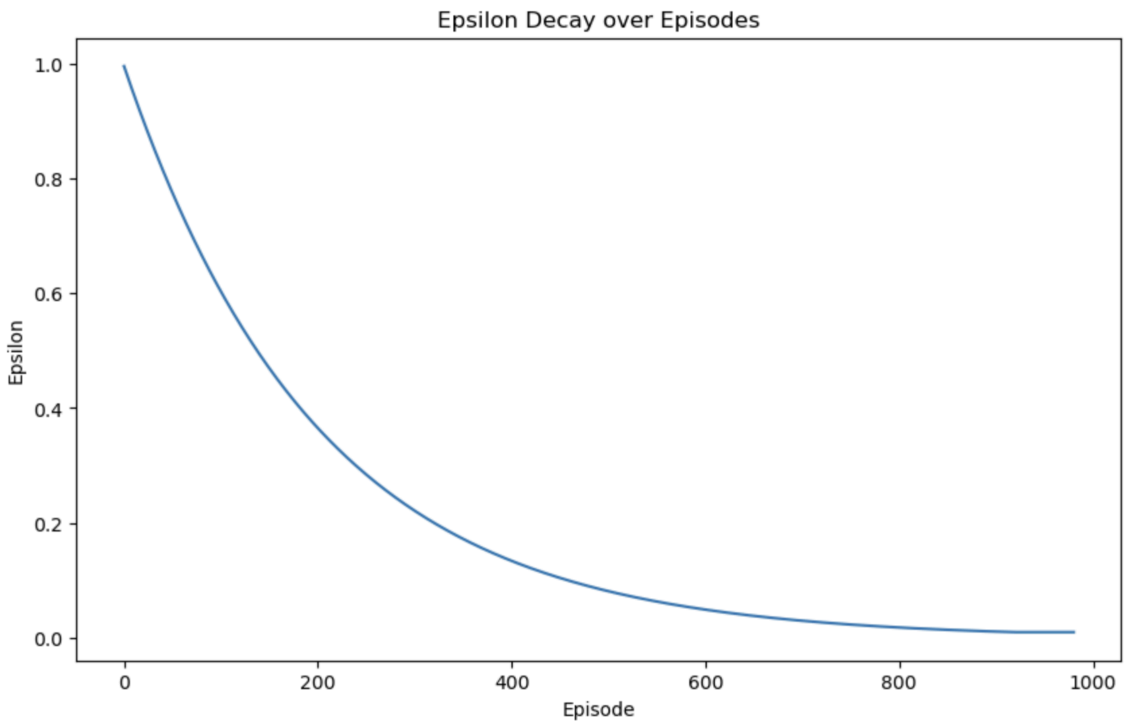
Total reward per episode

In the beginning, the rewards are low but gradually increase as the agent learns more effective strategies. Around the middle episodes, the rewards become higher but also more variable, showing both successes and mistakes. The spikes and drops indicate that the agent is still experimenting and adjusting its actions. By the later episodes, the rewards remain high but still fluctuate, suggesting the agent has improved significantly but earling stopping as avg reward is more than 470



Epsilon decay

epsilon decreases over 1000 episodes. Epsilon controls how often the agent explores by picking random actions instead of following what it's learned. At the start, epsilon is high, meaning lots of exploration, but it gradually drops as the agent gets more confident in its choices. By the end, epsilon is close to zero, so the agent mostly relies on what it's learned instead of exploring randomly. This helps it settle into a more stable strategy.

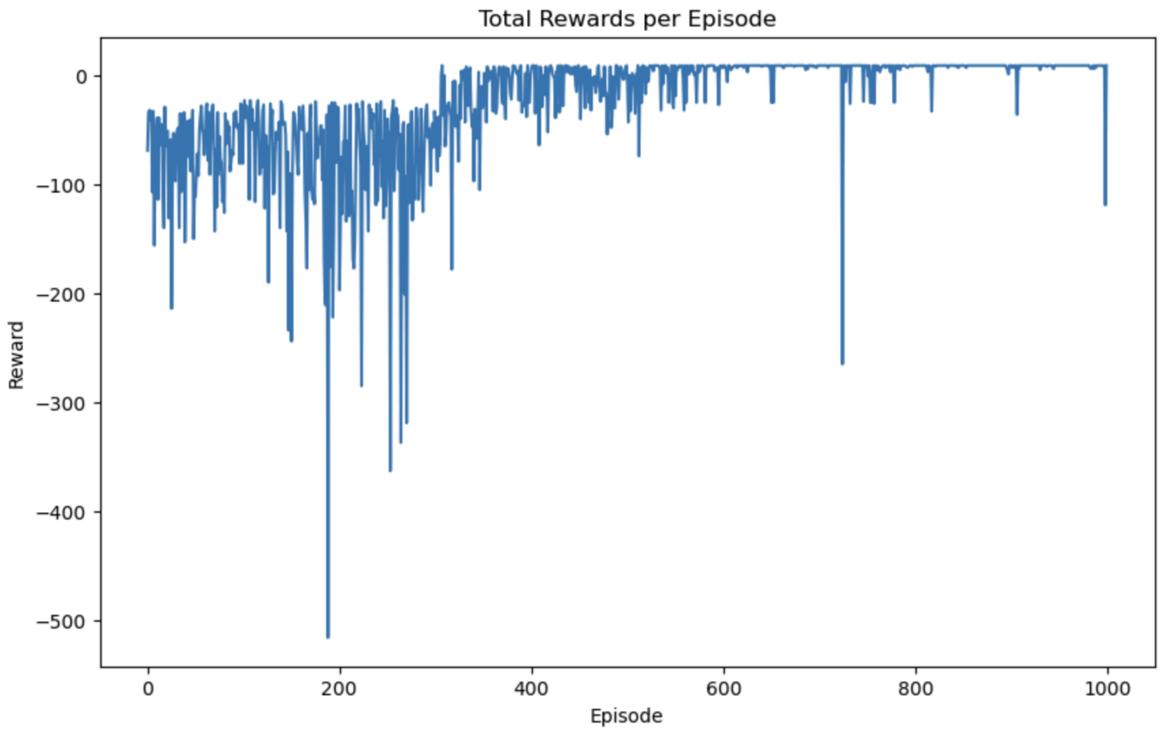


Cliffwalking

Total reward per episode

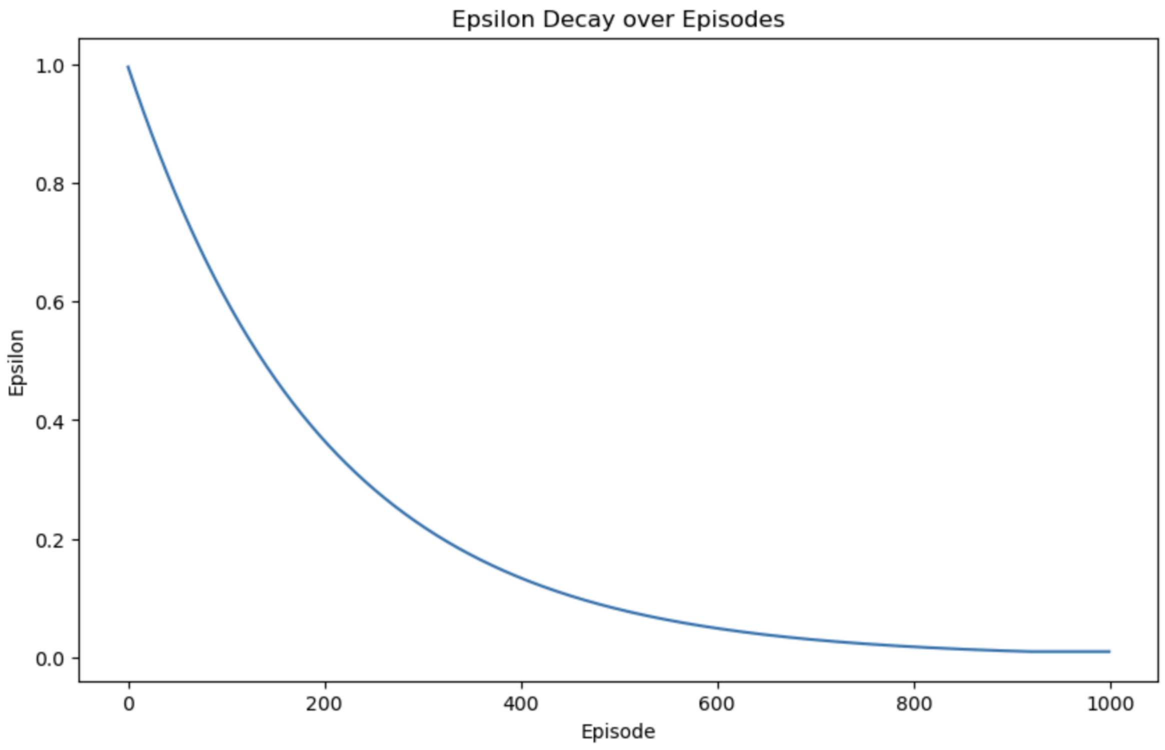
The below graph will show the total reward per episode ans with the early episodes displaying the high volatility and having the large negative rewards and over the time the rewards will stabilize and it will approach zero and it will indicate the improved performance.

```
Episode: 999, Reward: -13, Epsilon: 0.01
Episode: 1000, Reward: -13, Epsilon: 0.01
```



Epsilon decay

epsilon decreases over 1000 episodes. Epsilon controls how often the agent explores by picking random actions instead of following what it’s learned. At the start, epsilon is high, meaning lots of exploration, but it gradually drops as the agent gets more confident in its choices. By the end, epsilon is close to zero, so the agent mostly relies on what it’s learned instead of exploring randomly. This helps it settle into a more stable strategy.



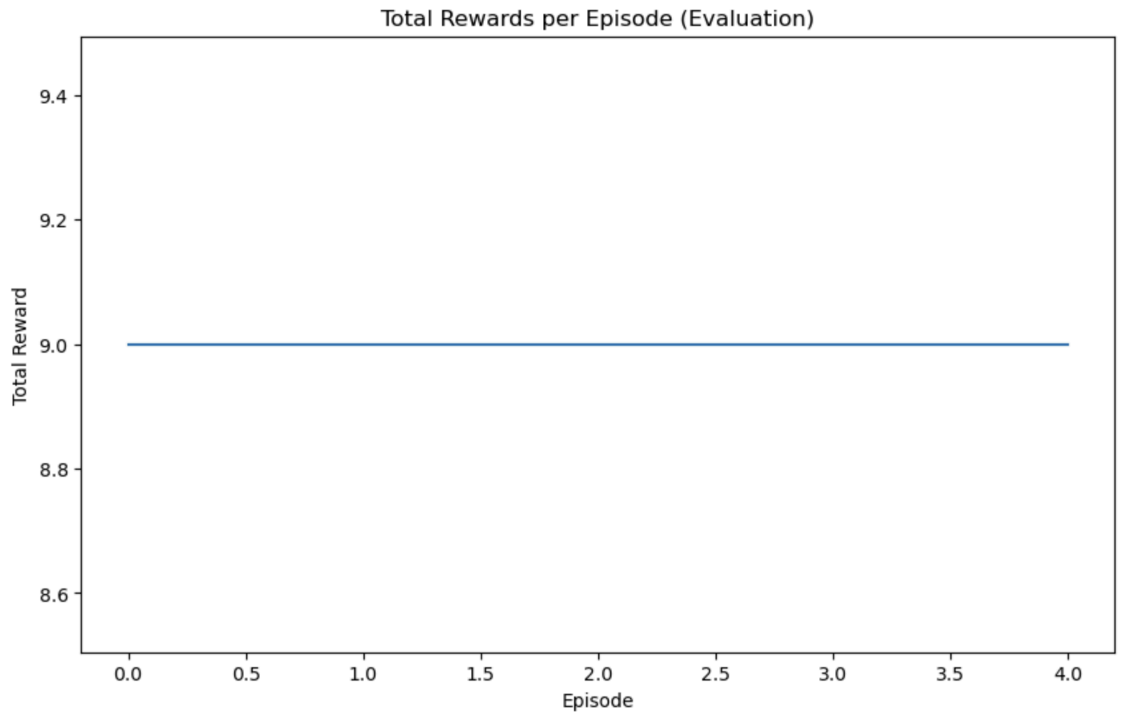
---

4) Provide the evaluation results. Run your environment for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

Warehouse robot

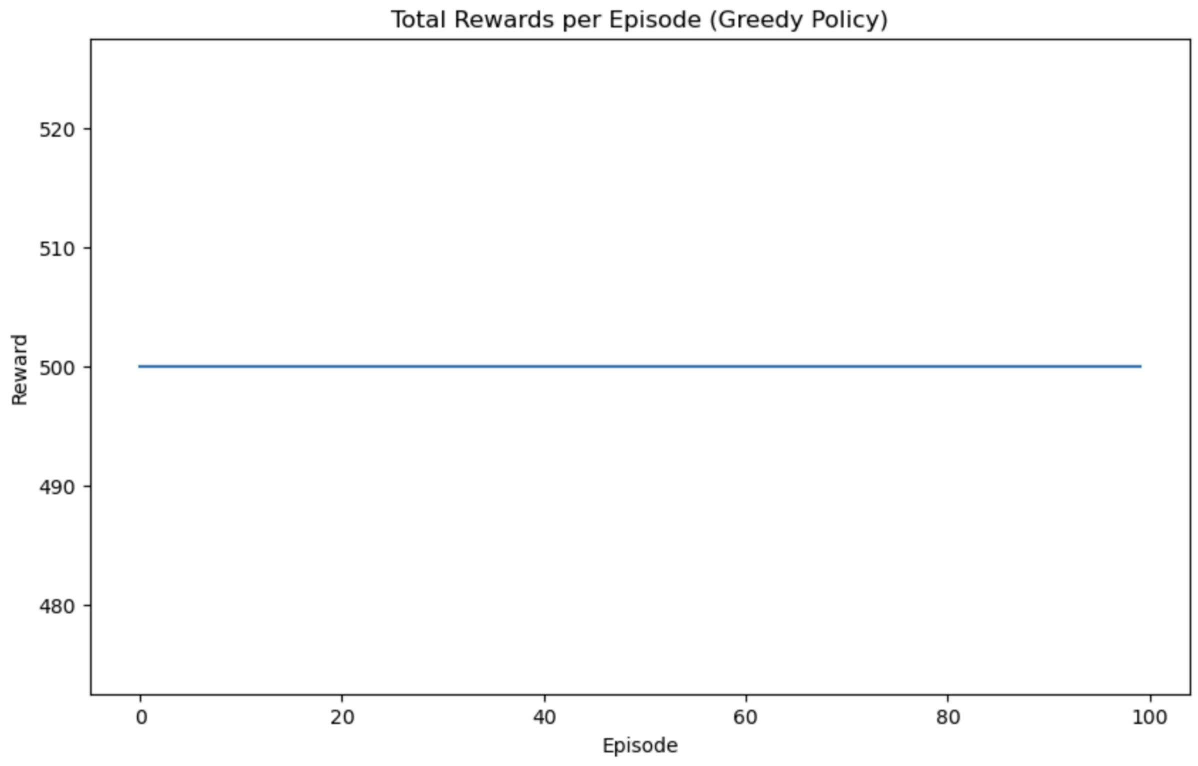
Each episode consistently receives a total reward of 9, indicating that the agent's performance is stable and it is choosing greedy actions

```
Episode: 1, Total Reward: 9
Episode: 2, Total Reward: 9
Episode: 3, Total Reward: 9
Episode: 4, Total Reward: 9
Episode: 5, Total Reward: 9
```



Cartpole

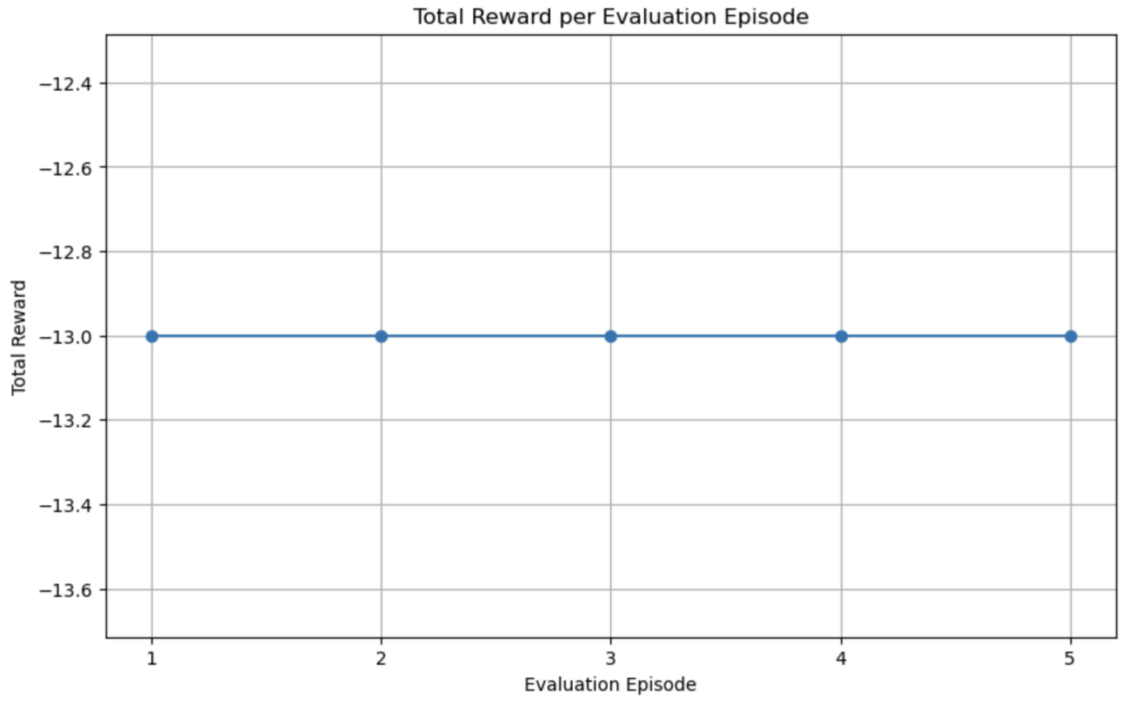
This graph shows the average reward per episode across 100 evaluation episodes. The reward remains consistently at 500, indicating that the agent has achieved optimal. The stable line suggests that the agent's learned policy performs reliably and achieves maximum reward in each episode.



Cliffwalking

The agent consistently receives a reward of -13 across all five evaluation episodes. The flat line at -13 suggests that the agent's performance is choosing greedy action

```
Evaluation Episode: 1, Reward: -13
Evaluation Episode: 2, Reward: -13
Evaluation Episode: 3, Reward: -13
Evaluation Episode: 4, Reward: -13
Evaluation Episode: 5, Reward: -13
```

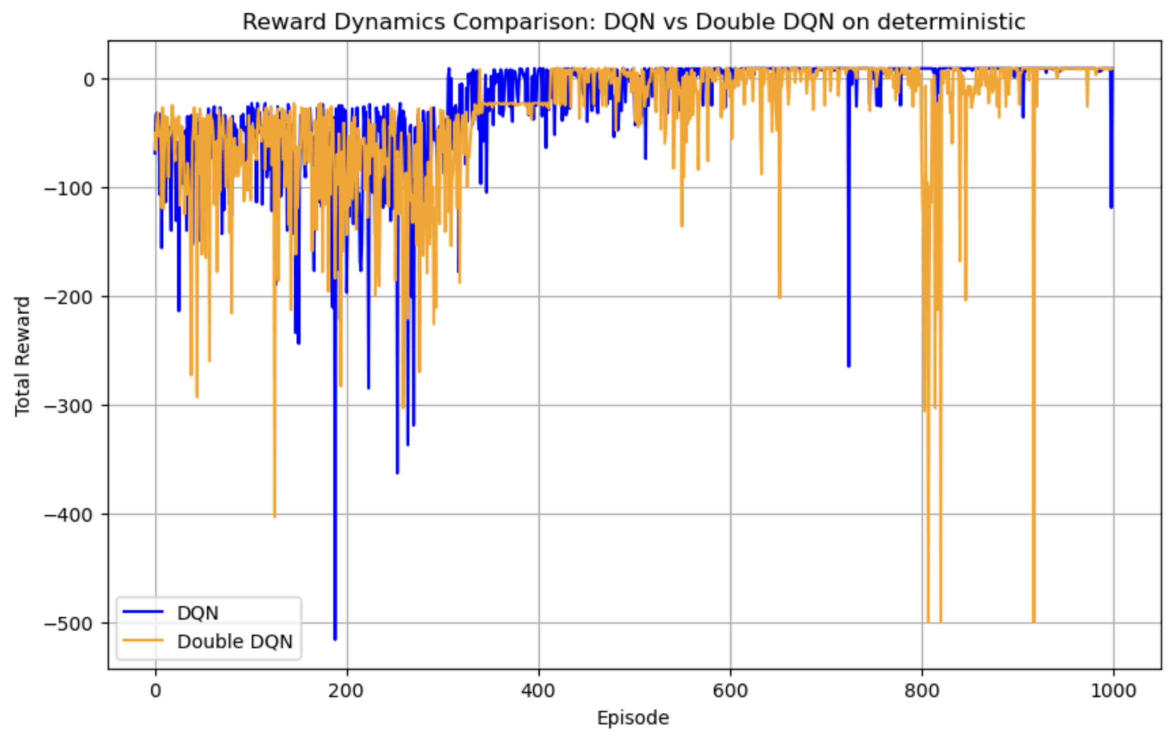


5) Compare the performance of both algorithms (DQN & Improved version of DQN) on the same environments (e.g. show one graph with two reward dynamics) and provide your interpretation of the results. Overall three rewards dynamics plots with results from two algorithms applied on:

- Grid-world environment
- 'CartPole-v1'
- Gymnasium envs or Multi-agent environment or any other complex environment

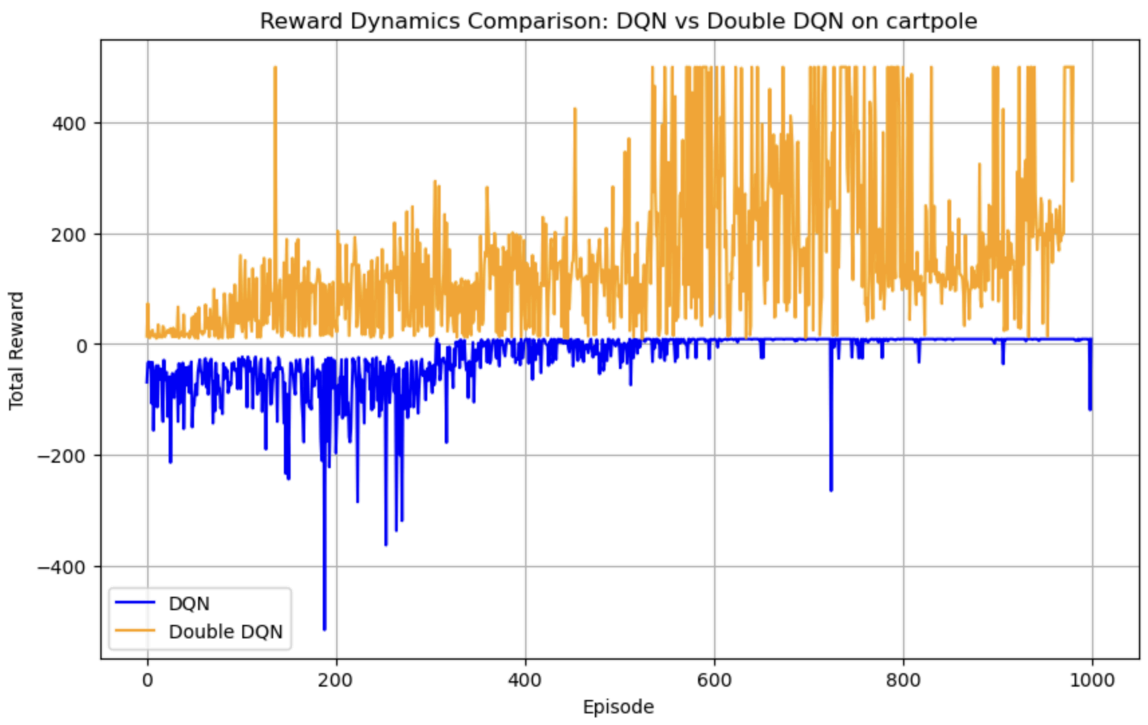
Warehouse robot environment

total rewards per episode for DQN (blue) and Double DQN (orange) in a deterministic environment. Both algorithms show a trend of increased stability and higher rewards over time, though Double DQN demonstrates more fluctuation early on. By around episode 600, both algorithms achieve relatively stable rewards close to zero.



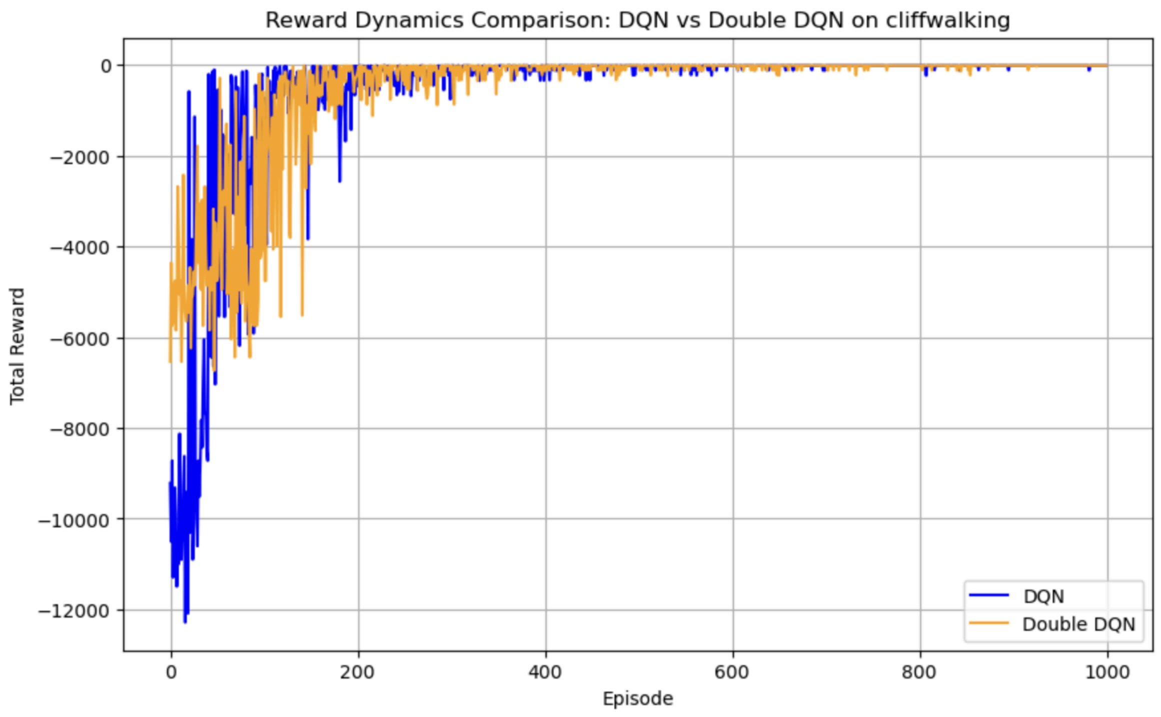
Cartpole environment

The Double DQN (orange) significantly outperforms DQN (blue), showing higher and more stable rewards across episodes. DQN struggles to increase rewards, often oscillating around negative values. This suggests Double DQN is better at learning and stabilizing the policy in the Cartpole environment.



Cliffwalking environment

Both DQN and Double DQN initially experience large negative rewards but gradually improve. Double DQN (orange) converges faster and achieves slightly higher performance, while DQN (blue) has more variability, indicating Double DQN's advantage in handling the challenging Cliffwalking environment.



6) Provide your interpretation of the results. E.g., how the same algorithm behaves on different environments, or how various algorithms behave on the same environment.

In the deterministic Warehouse environment, both DQN and Double DQN gradually improve their performance, with rewards stabilizing close to zero by episode 600. While both algorithms initially experience high variability in rewards, Double DQN displays more frequent fluctuations early on but eventually converges more smoothly.

In the CartPole environment, Double DQN consistently achieves higher rewards than DQN and converges more quickly, highlighting its advantage in environments where continuous positive rewards can be obtained by maintaining balance. DQN, while improving over time, shows a slower rate of convergence and struggles to match Double DQN's performance, indicating sensitivity to overestimations in the value function.

In the Cliffwalking environment, both algorithms face significant initial penalties due to frequent negative rewards, but Double DQN improves more steadily and achieves better overall performance. DQN exhibits greater variability and slower convergence, especially in handling the hazards associated with the environment. This comparison underscores Double DQN's ability to handle both continuous reward environments like CartPole and risk-intensive environments like Cliffwalking more effectively than DQN.

---

7)Include all the references that have been used to complete this part

1)https://medium.com/@ameetsd97/deep-double-q-learning-why-you-should-use-it-bedf660d5295
2)https://pylessons.com/CartPole-DDQN
3)https://medium.com/@leosimmons/double-dqn-implementation-to-solve-openai-gyms-cartpole-v-0-df554cd0614d
4)https://medium.com/@lgvaz/understanding-q-learning-the-cliff-walking-problem-80198921abbc
5)https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html