CSE 4/546 Reinforcement Learning
Spring 2025

Topic - Warehouse Robot

Instructor: Prof Alina Vereshchaka

Name : Ashutosh Sharan **UBID:** asharan2
UB Person Number : 50610518

Part I:

1. Describe the deterministic and stochastic environments:

- **States:** The environment is a 6x6 grid where the agent's state is defined by its current position (agent_pos) and the number of packages it has collected (num_packages). The grid includes package positions, obstacles, and a dropoff point.
- **Actions:** The agent can perform 6 discrete actions:
  - **0:** Move up
  - **1:** Move down
  - **2:** Move left
  - **3:** Move right
  - **4:** Pickup package
  - **5:** Dropoff package

- **Main Objective:** The goal is to navigate the grid, collect up to 3 packages(can be increased or decreased), and drop them off at the designated dropoff point while avoiding obstacles and unnecessary penalties.
- **Deterministic Environment :** Agent always moves as intended (100% accuracy).

- **Stochastic Environment :** Agent has a 90% chance of moving as intended and a 10% chance of staying in place.

2. Environment Visualization:

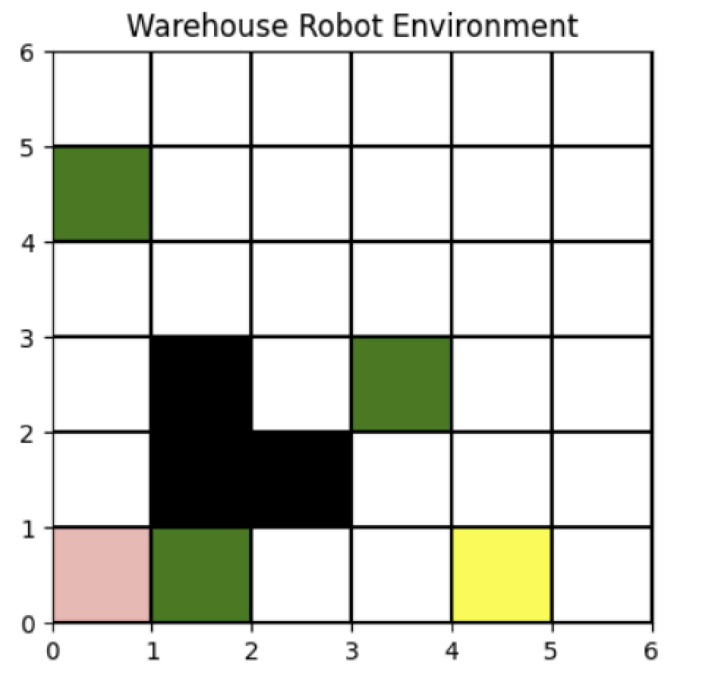
Initially:

[0,0] ->Agent (Pink block)

[[0, 1], [2, 3], [4, 0]] -> Initial package positions (Green block)

[0, 4] -> Fixed dropoff point (Yellow block)

[[1, 1], [1, 2], [2, 1]] -> Obstacles/Shelves (Black block)



Scenario : Agent tries to move.

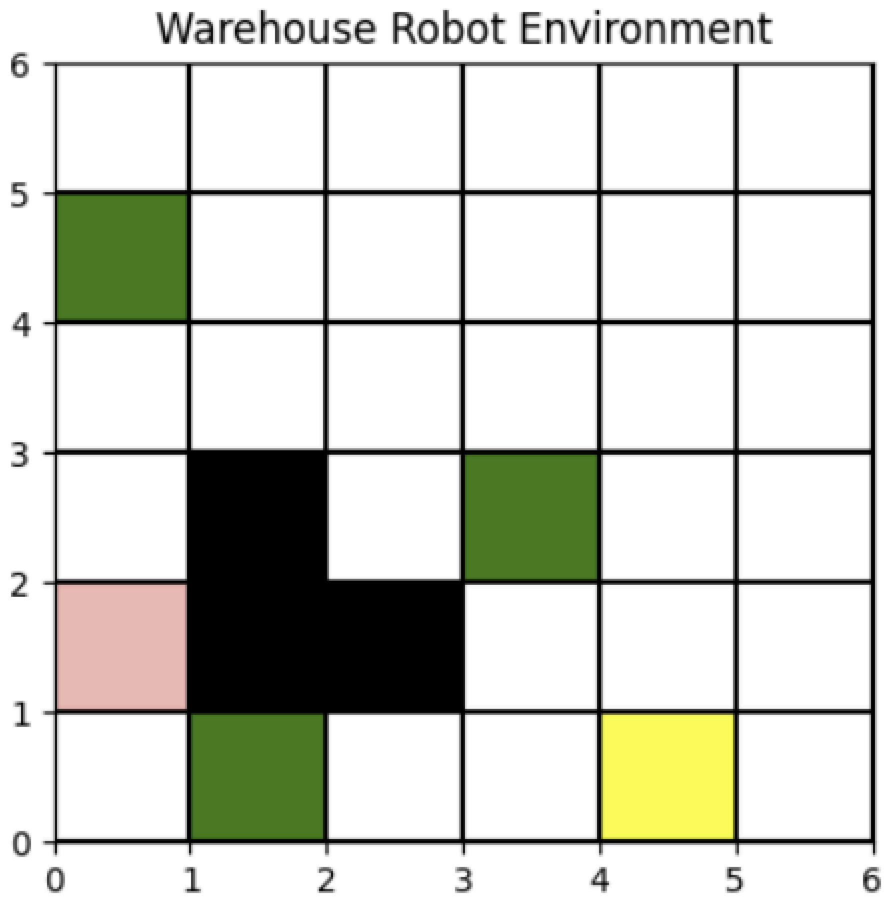
Action: 0 (Up)

Reward: -1

Cumulative Reward: -6

Agent Position: (1, 0)

Has Picked Up: False



Agent moves up to block [1,0]

Scenario : Agent and object location are the same

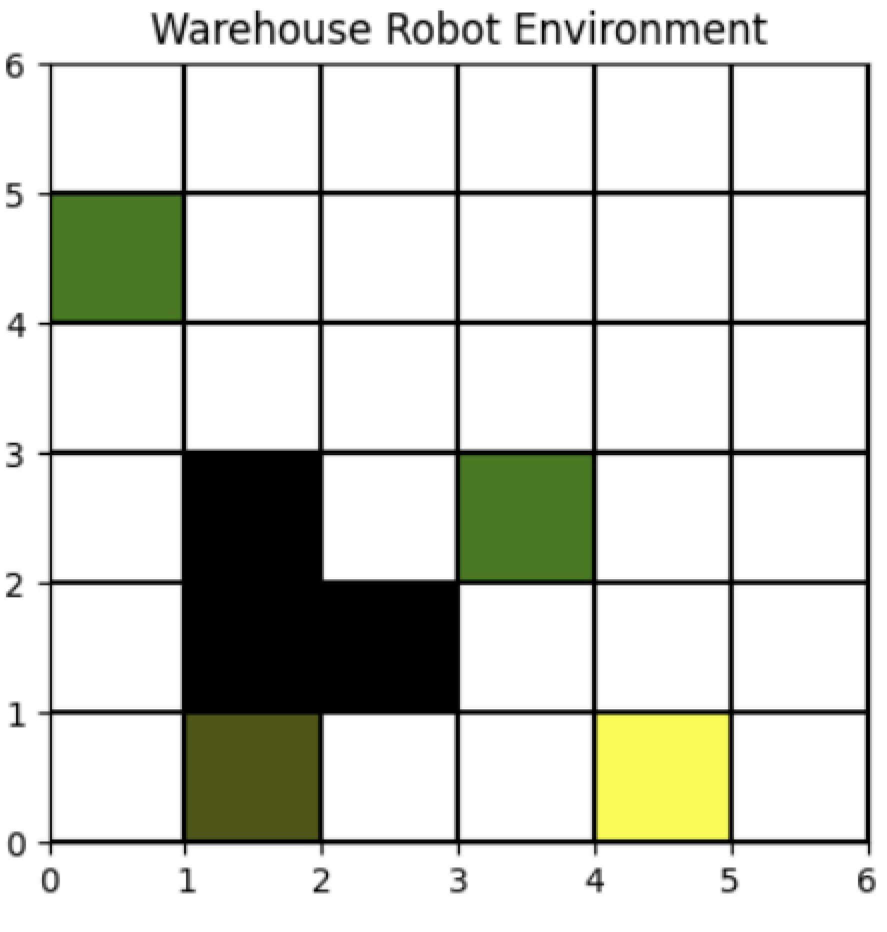
Action: 3 (Right)

Reward: -1

Cumulative Reward: -99

Agent Position: (0, 1)

Has Picked Up: False



Agent has become translucent because the object has not been picked up yet.

Scenario : Agent and object location are the same and agent picks up the object

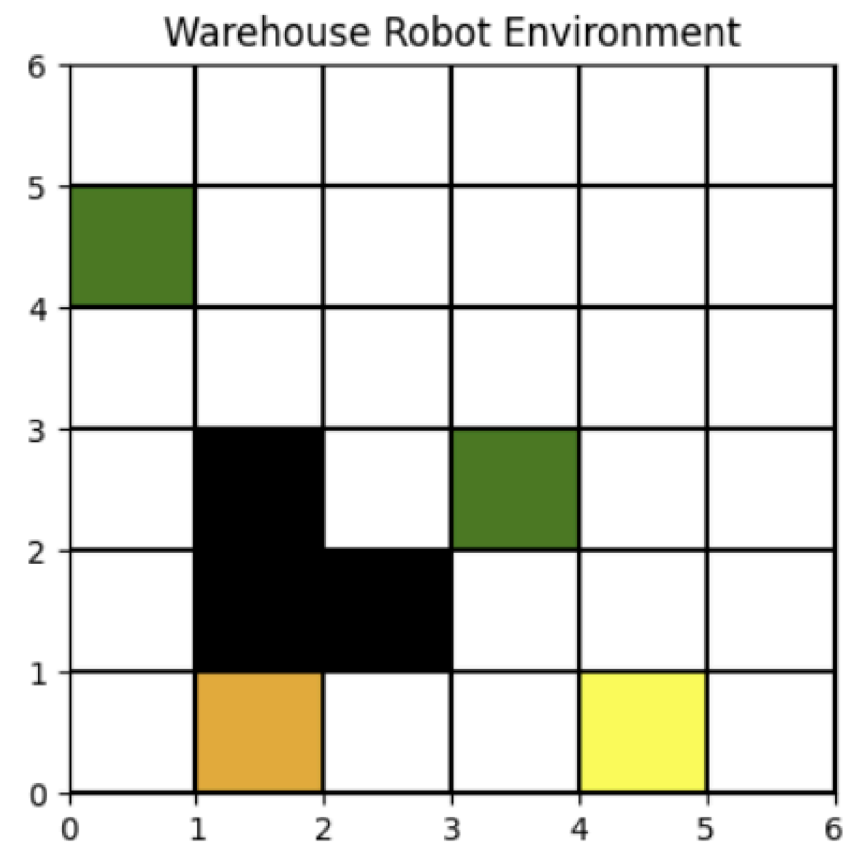
Action: 4 (Pickup)

Reward: 25

Cumulative Reward: -119

Agent Position: (0, 1)

Has Picked Up: True



Agent color becomes orange and the object is picked up.

3. How did you define the stochastic environment?

- The stochastic behavior is introduced during movement actions (0, 1, 2, 3) in the _move_agent method:

Python code :

if self.deterministic or random.random() < 0.9:

It means that there is a 90% chance to execute the intended move and a 10% chance to stay in the current position

In a **stochastic environment**, there is a small chance that the agent stays in its current position instead of moving in the intended direction, simulating real-world uncertainty like slippage or delays.

---

4. What is the difference between the deterministic and stochastic environments?

Aspect	**Deterministic Environment**	**Stochastic Environment**
Movement	Agent always moves as intended (100% accuracy).	Agent has a 90% chance of moving as intended and a 10% chance of staying in place.
Predictability	Outcomes are predictable based on the agent's actions.	Outcomes have an element of randomness due to stochastic behavior.
Complexity	Simpler, easier for the agent to plan optimal paths.	More challenging due to randomness, requiring robust exploration strategies.
Action Effect	Direct effect on the next state.	Indirect effect due to possible unexpected outcomes (agent staying in place).

---

5. Safety in AI:

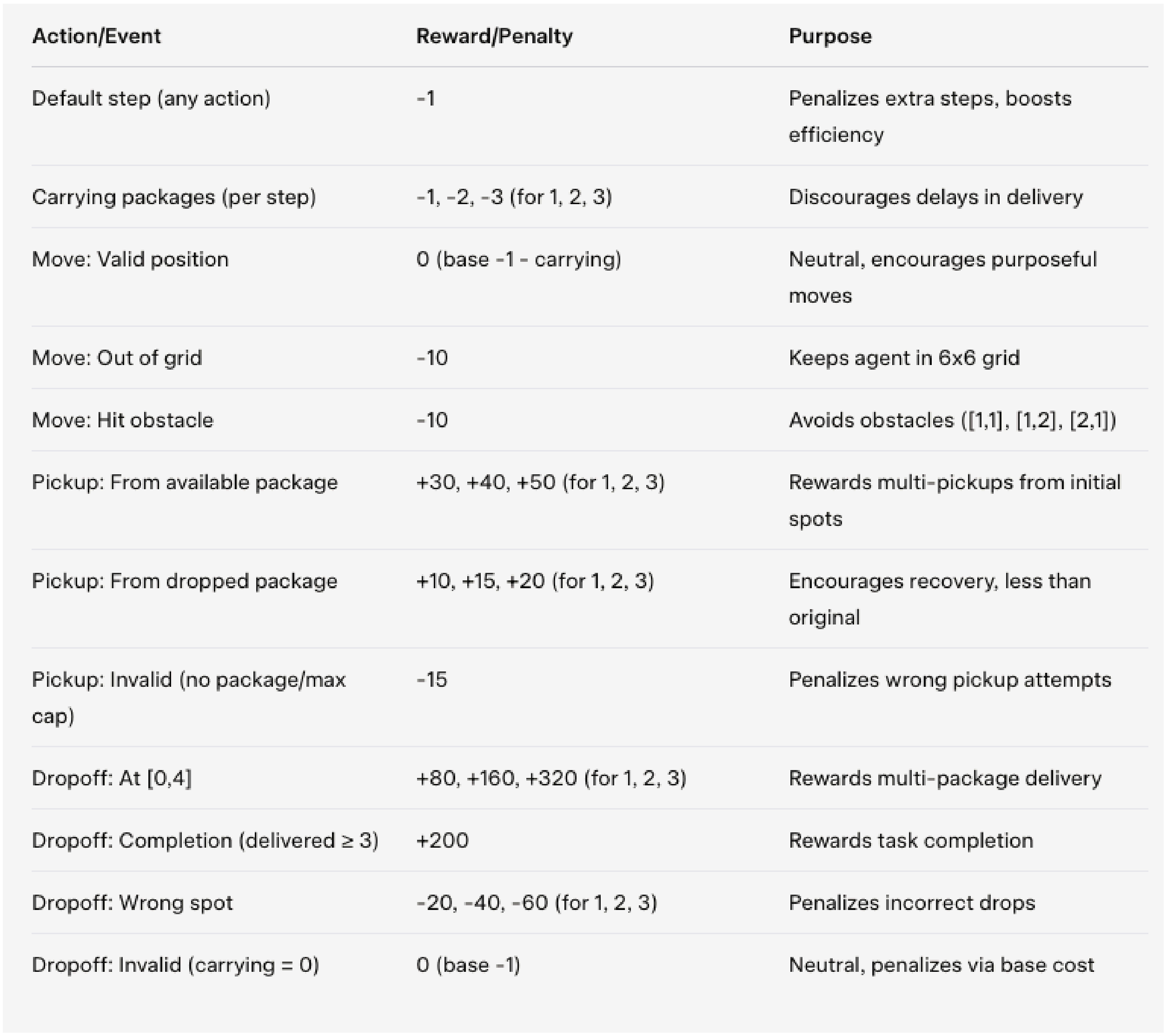
To ensure the safety of the environment:

- **Valid Movement:** The _is_valid method checks if the agent's next position is within grid boundaries and not blocked by obstacles:

- **Collision Handling:** If the agent attempts to move into an obstacle, it receives a penalty (-20), encouraging safer behavior.
- **Action Restrictions:** The pickup and dropoff actions are constrained based on conditions, such as ensuring the agent can only pick up a package when one is available and not exceeding the capacity of 3 packages.
- **Defined State Space:** The grid and states are clearly defined, preventing undefined behavior.
- **Reward Mechanism:** Negative rewards discourage invalid moves and dangerous actions, encouraging the agent to learn optimal and safe navigation.

---

6. Complex Environment Scenario: 3 Packages



Action/Event	Reward/Penalty	Purpose
Default step (any action)	-1	Penalizes extra steps, boosts efficiency
Carrying packages (per step)	-1, -2, -3 (for 1, 2, 3)	Discourages delays in delivery
Move: Valid position	0 (base -1 - carrying)	Neutral, encourages purposeful moves
Move: Out of grid	-10	Keeps agent in 6x6 grid
Move: Hit obstacle	-10	Avoids obstacles ([1,1], [1,2], [2,1])
Pickup: From available package	+30, +40, +50 (for 1, 2, 3)	Rewards multi-pickups from initial spots
Pickup: From dropped package	+10, +15, +20 (for 1, 2, 3)	Encourages recovery, less than original
Pickup: Invalid (no package/max cap)	-15	Penalizes wrong pickup attempts
Dropoff: At [0,4]	+80, +160, +320 (for 1, 2, 3)	Rewards multi-package delivery
Dropoff: Completion (delivered ≥ 3)	+200	Rewards task completion
Dropoff: Wrong spot	-20, -40, -60 (for 1, 2, 3)	Penalizes incorrect drops
Dropoff: Invalid (carrying = 0)	0 (base -1)	Neutral, penalizes via base cost

---

6. MECE Table:

Category	Test Case	Expected Outcome
Movement	Valid move	Agent updates its position successfully.
	Obstacle collision	Agent’s position doesn’t change, and a penalty is applied.
	Out of grid	Agent’s position doesn’t change, and a penalty is applied.
Pickup	Successful pickup	Agent picks up a package, and it’s removed from the environment.
	Invalid pickup (no package)	No change, and penalty is applied.
	Invalid pickup (3 packages)	No change, and penalty is applied.
Dropoff	Successful dropoff	Packages dropped, and reward is applied.
	Invalid dropoff (occupied cell)	No packages dropped, and a penalty is applied.
	Invalid dropoff (no packages held)	No action, and penalty is applied.
Special dropoff point	Dropoff at fixed point	Bonus reward applied, and the environment terminates.
Stochastic behavior	Randomized outcome	10% chance to stay in place, 90% chance to move correctly.
Cumulative reward	Reward accumulation	Correct cumulative reward based on actions.
Termination condition	Task completion	Environment terminates correctly at the dropoff point.
Edge cases	Empty package list	Handle gracefully without errors.
	Full package capacity	Cannot pick more than 3 packages.
	Stuck due to obstacles	Handle gracefully or end the episode.

Part II:

1. Results and Discussion

1.1 Q-learning in Deterministic Environment

I applied Q-learning to the deterministic warehouse environment, where the agent starts at [0,0], picks up packages from [4,5], [3,3], and [5,5], delivers them to [0,4], and avoids obstacles at [1,1], [1,2], and [2,1]. I trained the agent using optimized parameters obtained via Optuna.

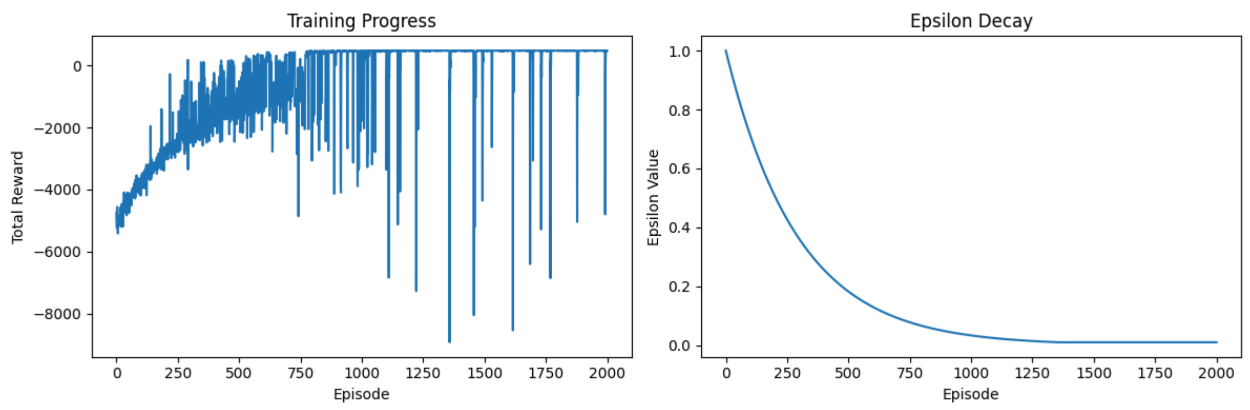
Best Hyperparameters Found:

Alpha (Learning Rate): 0.0176

Gamma (Discount Factor): 0.9492

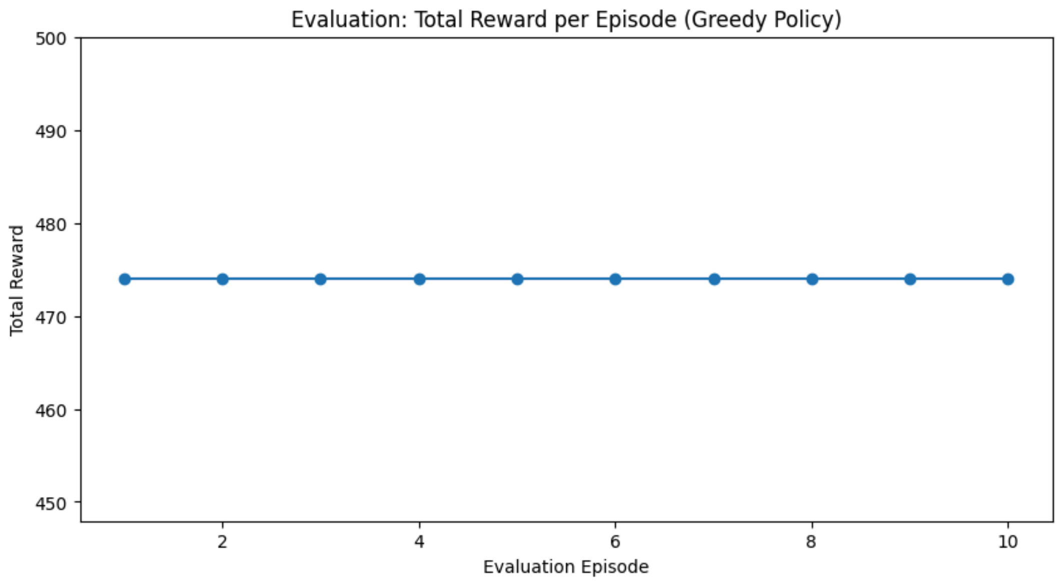
Epsilon Decay: 0.9966

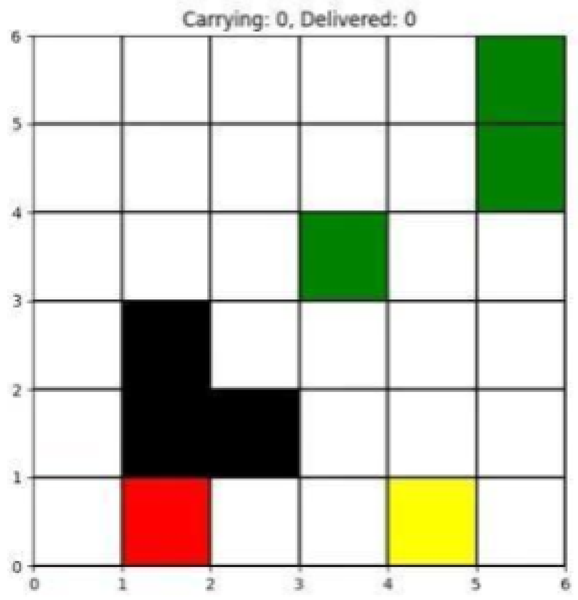
Plots and Results:



- **Total Reward per Episode Plot :** The "Training Progress" plot illustrates the agent's total reward over 2000 episodes of training. Initially, the rewards are highly negative and fluctuate significantly, reflecting the agent's exploratory actions and limited knowledge of the environment. As training progresses, the total reward improves, showing that the agent is learning a better policy. However, there are occasional sharp drops in reward after approximately 1000 episodes, which could indicate instability or challenges in policy optimization. Despite these fluctuations, the overall trend suggests that the agent is gradually converging toward a more effective strategy.

- **Epsilon Decay Plot :** The "Epsilon Decay" plot shows the gradual reduction of the epsilon value over 2000 episodes, following an exponential decay schedule. Starting at 1.0, epsilon decreases rapidly during the initial episodes, encouraging extensive exploration of the environment. As training progresses, epsilon approaches a near-zero value, reducing exploration and prioritizing exploitation of the learned policy. This controlled reduction helps balance exploration and exploitation, allowing the agent to refine its strategy while minimizing random actions in later stages of training.



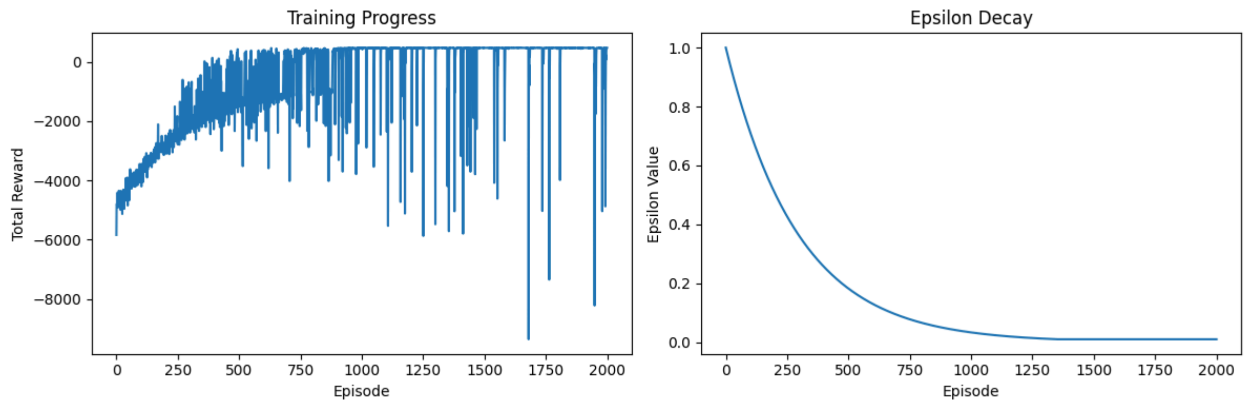


Double click on this image to play or click this link : download1.mp4

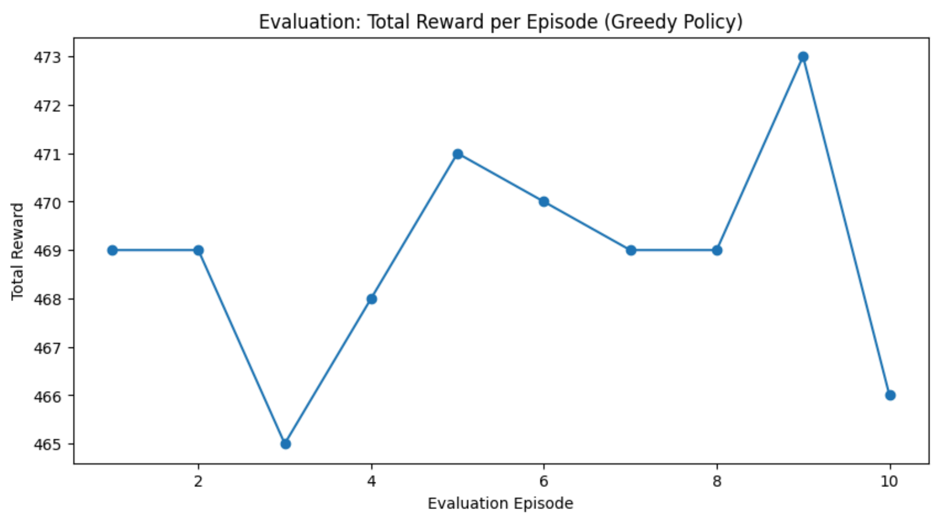
1.2 Q-learning in Stochastic Environment

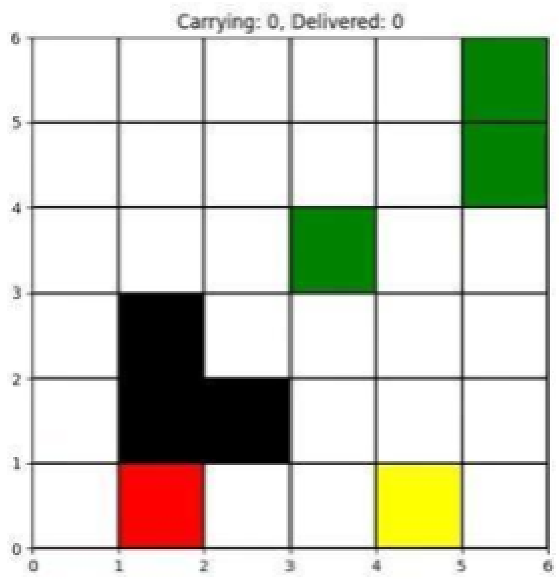
I applied Q-learning to the stochastic environment, where movement actions have a 10% chance of slipping (remaining in place). I used the same training parameters as the deterministic case.

Plots and Results:



- **Epsilon Decay Plot**: The right graph depicts the epsilon decay over 2000 episodes, which controls the exploration-exploitation balance in an epsilon-greedy policy. Epsilon starts at 1 (pure exploration) and decreases exponentially, encouraging more exploitation of learned strategies as training progresses.
- **Total Reward per Episode Plot**: The left graph shows the agent's total reward per episode during training in a stochastic environment. Initially, the rewards are significantly negative, indicating poor performance as the agent explores and learns the environment. Over time, the rewards improve, reflecting the agent's ability to develop a better policy. However, due to the stochastic nature of the environment, there are noticeable fluctuations and occasional sharp drops in rewards even in later episodes. Despite these variations, the overall trend demonstrates that the agent is progressively adapting and achieving more consistent performance.



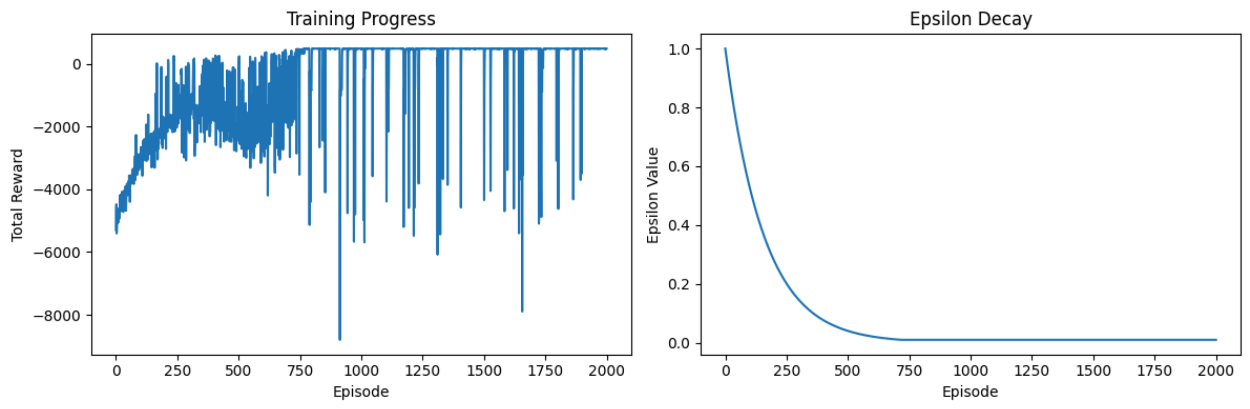


Double click on this image to play or click this link : download2.mp4

1.3 Double Q-learning (DQL) in Deterministic Environment

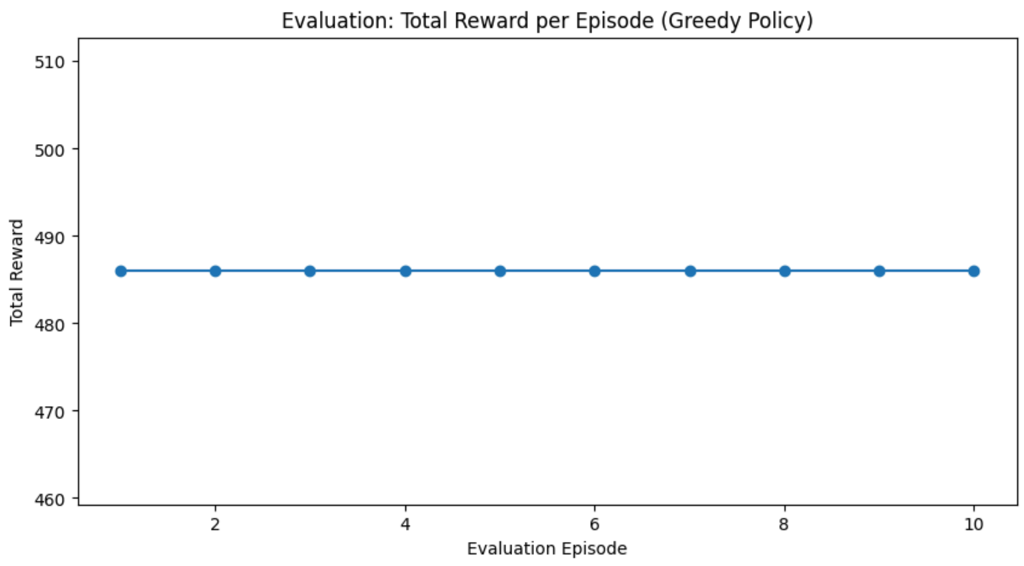
I applied Double Q-learning, using two Q-tables to reduce overestimation bias, to the deterministic environment with optimized hyperparameters using Optuna (alpha=0.1137, gamma=0.9924, epsilon_decay=0.9936, as specified in the notebook). The goal is to maximize multiple pickups and thereby the rewards.

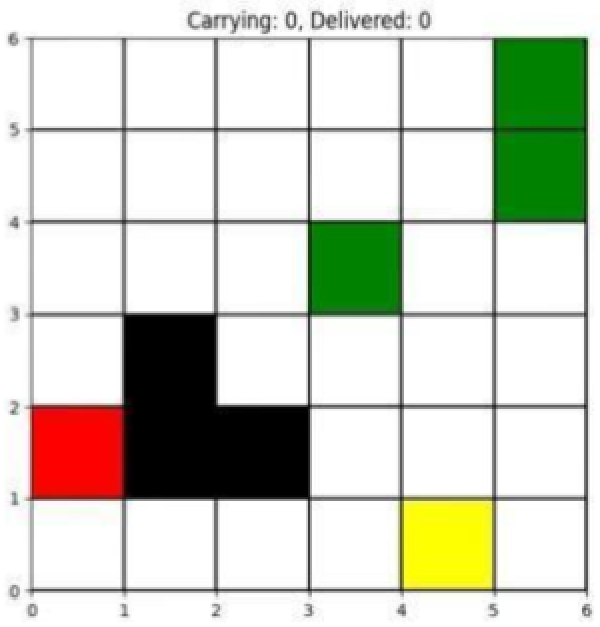
Plots and Results:



- **Total Reward per Episode Plot**: In Double Q-Learning, the training progress shows fewer extreme drops in rewards compared to Q-Learning. This suggests that Double Q-Learning mitigates the overestimation bias inherent in standard Q-Learning.The rewards in Double Q-Learning stabilize more consistently over time, indicating better convergence behavior. In contrast, Q-Learning exhibits more erratic fluctuations and instability, especially in later episodes.

- **Epsilon Decay Plot :** The epsilon decay curves are similar in both cases, as they depend on the exploration schedule rather than the algorithm itself. As epsilon decreases, both algorithms shift from exploration to exploitation. However, Double Q-Learning benefits from its reduced bias during exploitation.





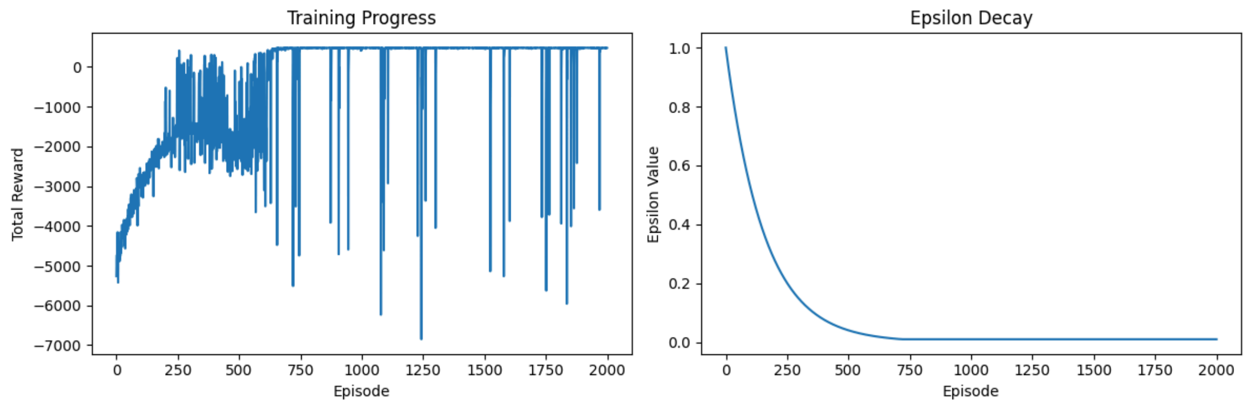
Double click on this image to play or click this link : download3.mp4

DQL outperforms Q-learning, achieving higher rewards (486 in evaluations compared to 474) due to its reduced overestimation bias. The optimized hyperparameters enhance convergence, and the robot effectively prioritizes multiple pickups, aligning with the reward structure.

1.4 Double Q-learning (DQL) in Stochastic Environment

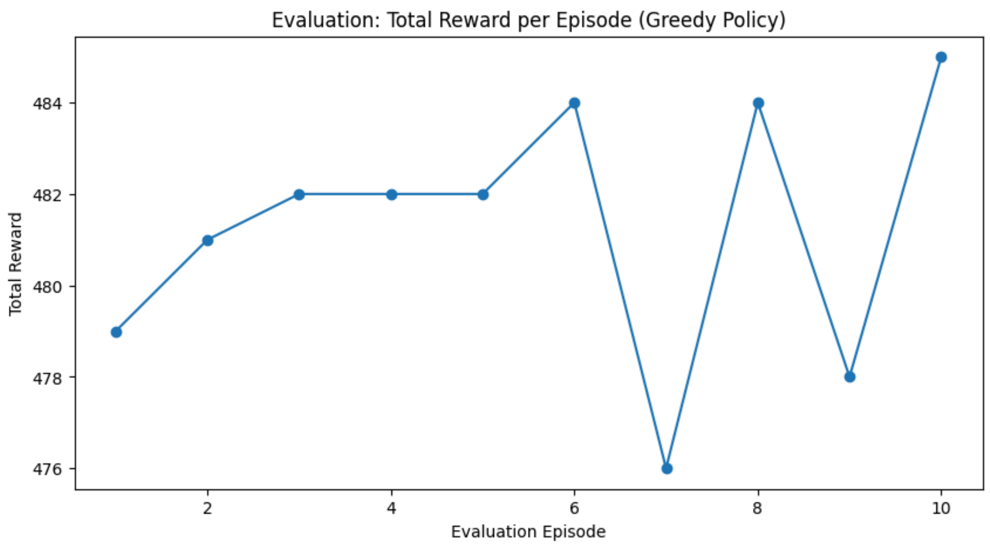
I applied Double Q-learning, using two Q-tables to the stochastic environment with the same optimized hyperparameters. The stochastic slip (10%) challenges the agent, but DQL mitigates overestimation bias.

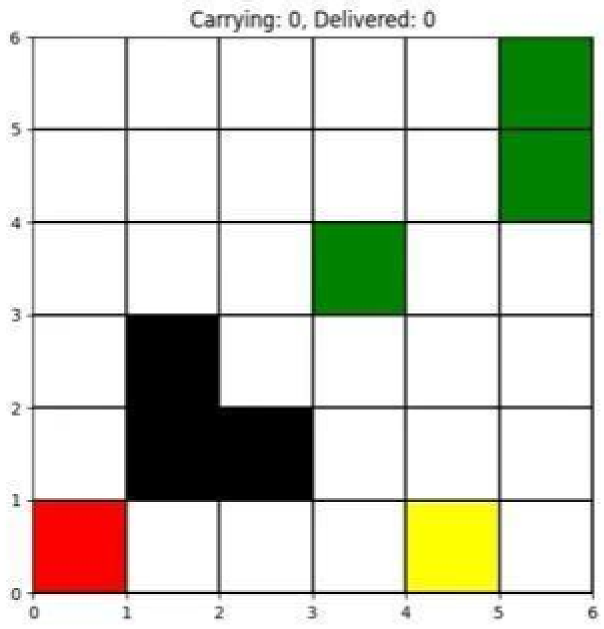
Plots and Results:



- **Total Reward per Episode Plot**: In Double Q-Learning, the total reward curve shows less variability and fewer extreme drops in rewards compared to the first image (Q-Learning). This indicates that Double Q-Learning is more stable during training.

- **Epsilon Decay Plot :** Both methods use an epsilon-greedy strategy for exploration, as shown by the identical epsilon decay curves in both images. However, Double Q-Learning's improved stability ensures that it exploits learned policies more effectively during later stages of training when epsilon is low.



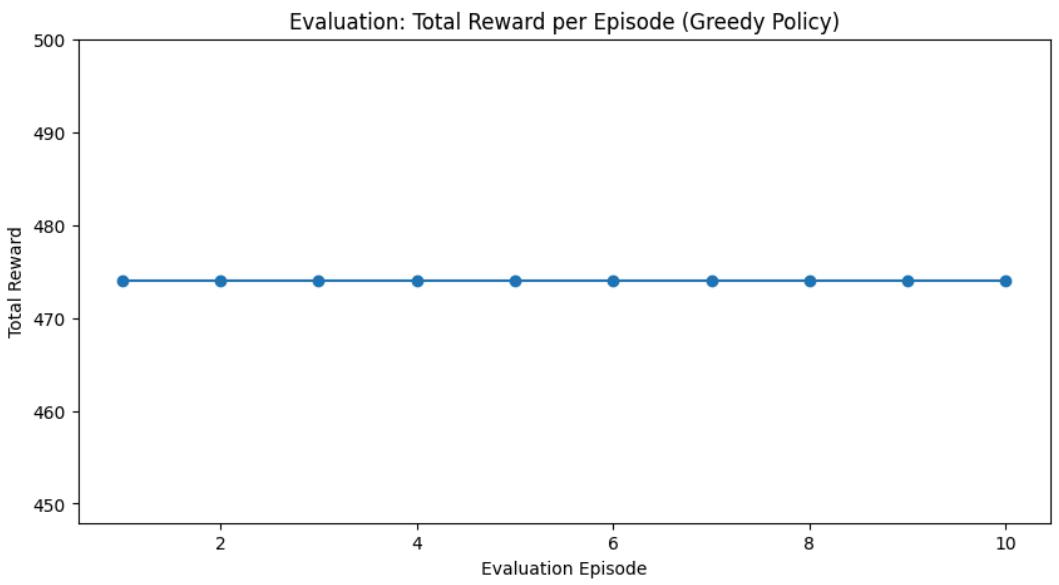


Double click on this image to play or click this link : download4.mp4

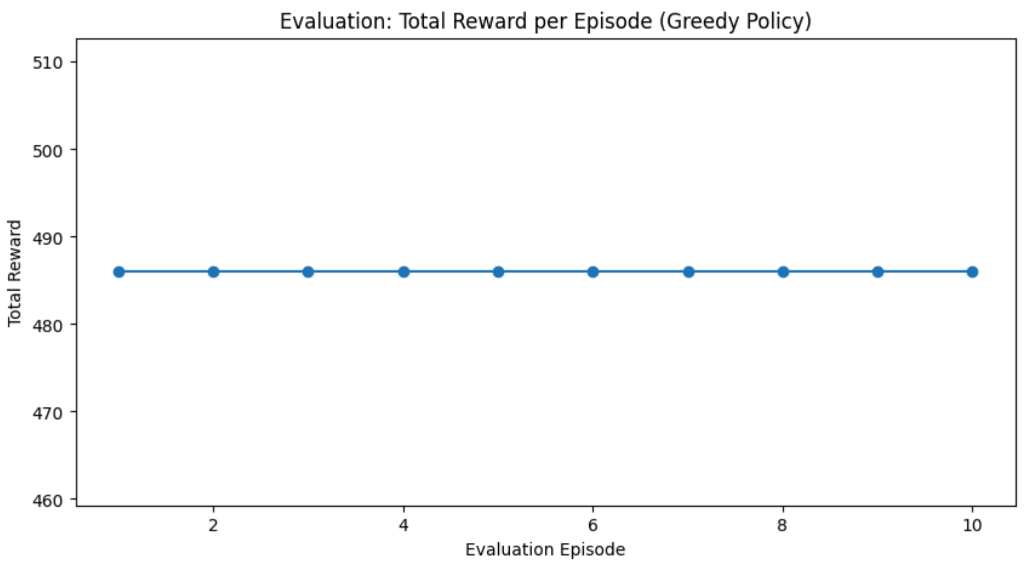
DQL outperforms Q-learning, achieving higher rewards (485 in evaluations compared to 473) due to its reduced overestimation bias. The optimized hyperparameters enhance convergence, and the robot effectively prioritizes multiple pickups, aligning with the reward structure.

1.5 Evaluation Results (Greedy Policy)

Deterministic Environment:

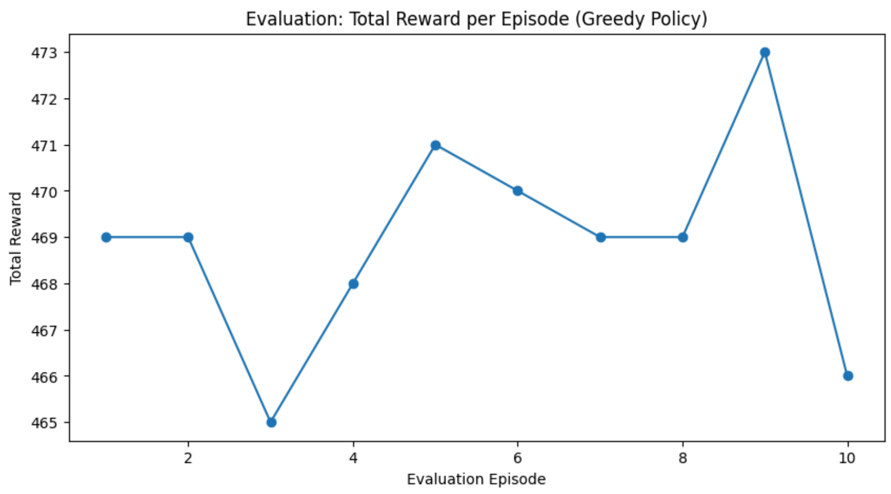


- **Q-learning**: In Q-learning evaluation data over 10 episodes using a greedy strategy (epsilon=0), rewards are consistently 474 across 10 episode

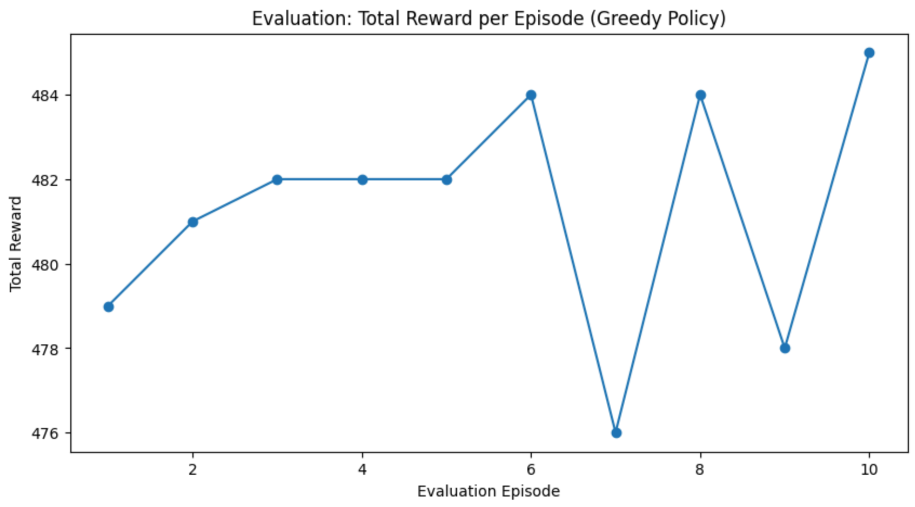


- **DQL**: In Double Q-learning evaluation data over 10 episodes using a greedy strategy (epsilon=0), rewards are consistently 486 across 10 episode

Stochastic Environment:



- **Q-learning**: In Q-learning evaluation data over 10 episodes using a greedy strategy (epsilon=0), rewards reach a max of 473 across 10 episode

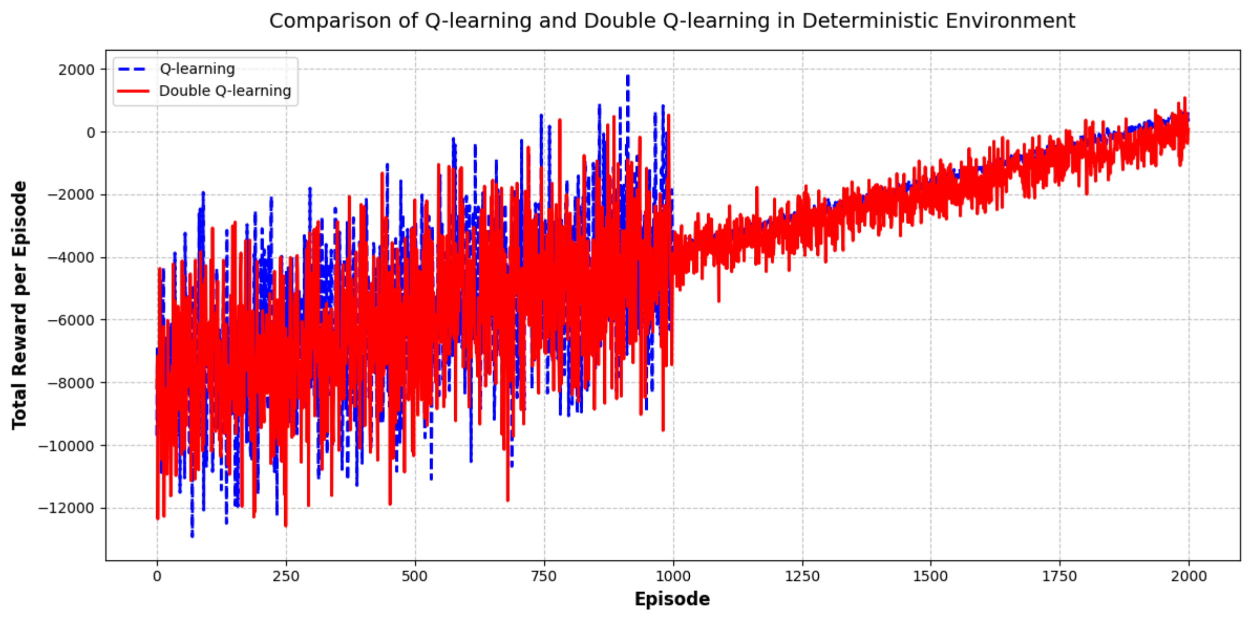


- **DQL**: In Q-learning evaluation data over 10 episodes using a greedy strategy (epsilon=0), rewards reach a max of 485 across 10 episode

DQL consistently outperforms Q-learning in both environments, achieving higher, stable rewards by optimizing for multiple pickups.

2. Comparison of Algorithms in Deterministic Environment

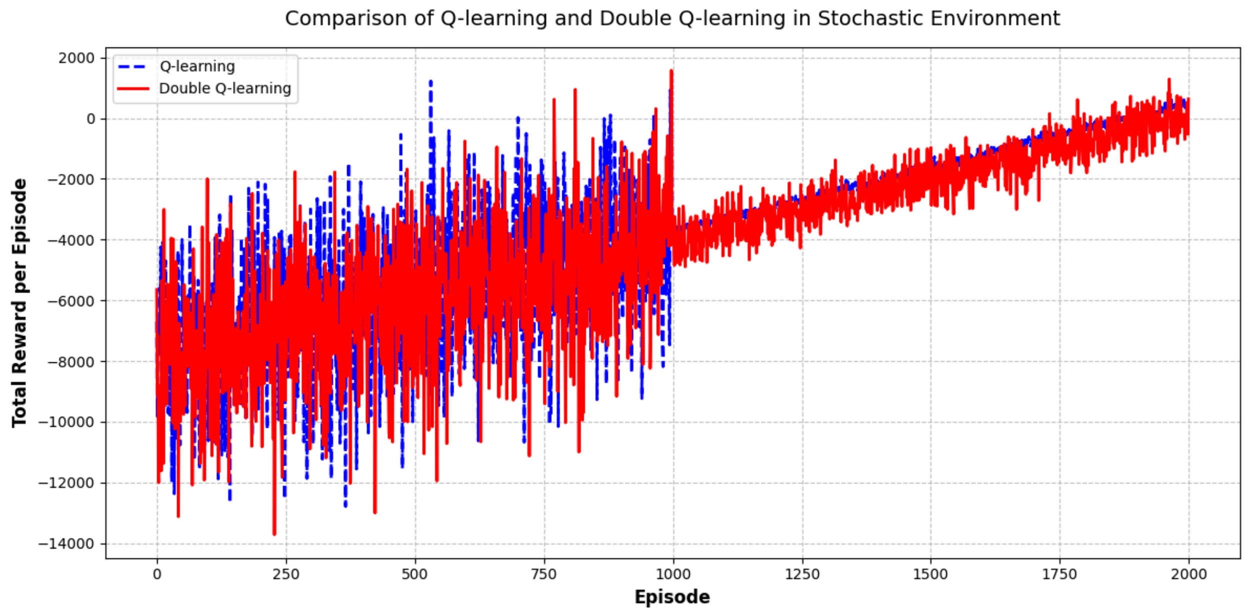
Combined Reward Plot:



Interpretation: Both algorithms show significant variability in rewards during early episodes, reflecting instability as they learn to optimize their policies. Over time, however, both methods demonstrate improvement, with rewards trending upward as episodes progress. Double Q-learning (red line) appears to stabilize slightly earlier and exhibits less extreme fluctuations compared to Q-learning (blue dashed line), suggesting it mitigates overestimation bias more effectively. This aligns with the theoretical advantage of Double Q-learning in reducing over-optimistic value estimates, leading to more consistent learning in deterministic settings. Overall, both methods converge toward higher rewards, but Double Q-learning shows a smoother trajectory.

3. Comparison of Algorithms in Stochastic Environment

Combined Reward Plot:



Interpretation: Initially, both algorithms exhibit high variability in rewards, with large fluctuations in performance. As training progresses, the rewards for both methods stabilize and improve significantly, indicating learning and adaptation to the environment. Double Q-learning appears to converge slightly faster and demonstrates less extreme negative fluctuations compared to Q-learning, suggesting its robustness in mitigating overestimation bias in stochastic settings. This comparison highlights the effectiveness of Double Q-learning in achieving more stable learning outcomes in complex environments.

4. Explanation of Tabular Methods

Q-learning: An off-policy TD method that updates Q-values using the maximum future reward:

- **Update Rule**:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

- **Key Features**: Model-free, converges to the optimal policy, uses epsilon-greedy exploration, but overestimates in stochastic settings due to max operator bias.

Double Q-learning (DQL): An off-policy TD method using two Q-tables to reduce overestimation bias:

- **Update Rule**:

  - **Estimator Q1**: Obtain best actions
  - **Estimator Q2**: Evaluate Q for the above action

$$Q_1(s, a) \leftarrow Q_1(s, a) + \alpha(\text{Target} - Q_1(s, a))$$

Q Target: $r(s, a) + \gamma \max_{a'} Q_1(s', a')$

Double Q Target: $r(s, a) + \gamma\, Q_2(s', \operatorname{argmax}_{a'} Q_1(s', a'))$

- **Key Features**: Model-free, mitigates overestimation, robust to stochasticity, uses epsilon-greedy exploration, suitable for both deterministic and stochastic environments.

Both methods use Q-tables for discrete state-action pairs, fitting your 6x6 grid and 6 actions.

5. Criteria for a Good Reward Function

A well-designed reward function in reinforcement learning should meet the following criteria:

1. **Guides Toward the Goal**: Rewards should align with the ultimate objective (here, delivering all packages to [0,4]).

2. **Encourages Efficient Behavior**: Positive rewards should promote minimal steps and optimal strategies (e.g., delivering multiple packages at once).

3. **Discourages Undesirable Actions**: Negative rewards should deter mistakes (e.g., hitting obstacles, dropping packages incorrectly).

4. **Appropriately Shaped**: Intermediate rewards should help the agent learn step-by-step, especially in complex tasks.

5. **Balances Exploration and Exploitation**: Rewards should neither be too sparse (hindering learning) nor too dense (over-rewarding trivial actions).

In this environment, the reward function encourages picking up multiple packages (progressive pickup rewards), delivering them together (exponential drop-off rewards), and avoiding errors (significant penalties), aligning well with these criteria.

I tried multiple reward functions and this is the interpretation of the results :

1. **Higher Carrying Penalty (e.g., -1 - 2 * self.carrying)**:
   - **Effect**: Carrying 1 package would cost -3 per step, 2 packages -5, etc., compared to -2 and -3 now.
   - **Result**: The agent preferred delivering packages one by one to avoid high penalties, increasing total steps and reducing efficiency. Total reward could drop due to fewer multi-package deliveries (e.g., +80 each vs. +320 for 3).

2. **Flat Pickup Reward (e.g., +20 regardless of carrying)**:
   - **Effect**: No extra incentive for sequential pickups.
   - **Result**: The agent picked up packages sporadically rather than clustering pickups, potentially leading to more trips and lower rewards from drop-offs (e.g., more +80s instead of +160 or +320).

3. **Linear Drop-off Reward (e.g., +40 * self.carrying)**:
   - **Effect**: Delivering 1 = +40, 2 = +80, 3 = +120 vs. current +80, +160, +320.
   - **Result**: Less advantage in delivering multiple packages at once, encouraging more frequent smaller deliveries, reducing total reward (e.g., +120 + 200 vs. +320 + 200).

4. **No Distinction Between Package Types**:
   - **Effect**: Picking up from dropped packages yields the same reward as available packages.
   - **Result**: The agent was not deterred from dropping packages incorrectly, leading to suboptimal cycles of dropping and re-picking, lowering efficiency and total reward.

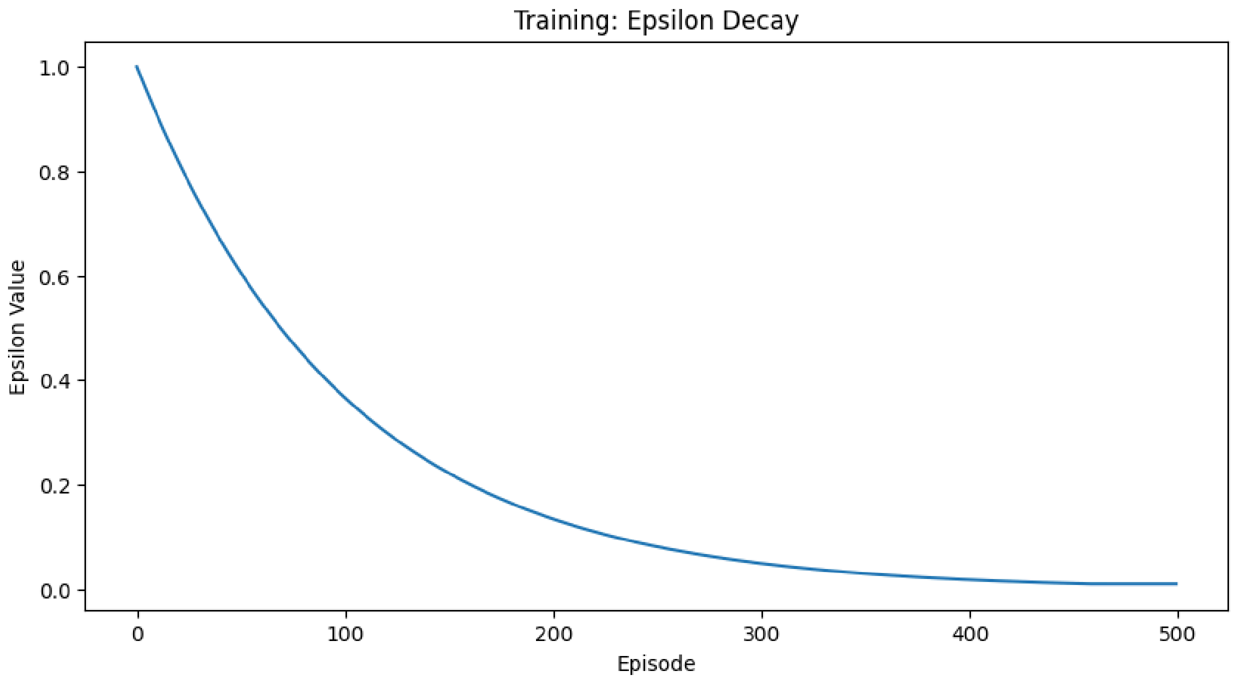
The current reward function, with its progressive pickup rewards (+30, +40, +50), exponential drop-off rewards (+80, +160, +320), and moderate carrying penalty, outperforms these alternatives by strongly incentivizing an efficient strategy: trying to pick up all 3 packages and delivering them together. This is evident from the high reward for delivering 3 packages (+320 + 200 = +520), which can offset the modest per-step costs (-1 to -4) even over a longer path.

PART III

Results of Applying Q-Learning to the Stock Trading Problem

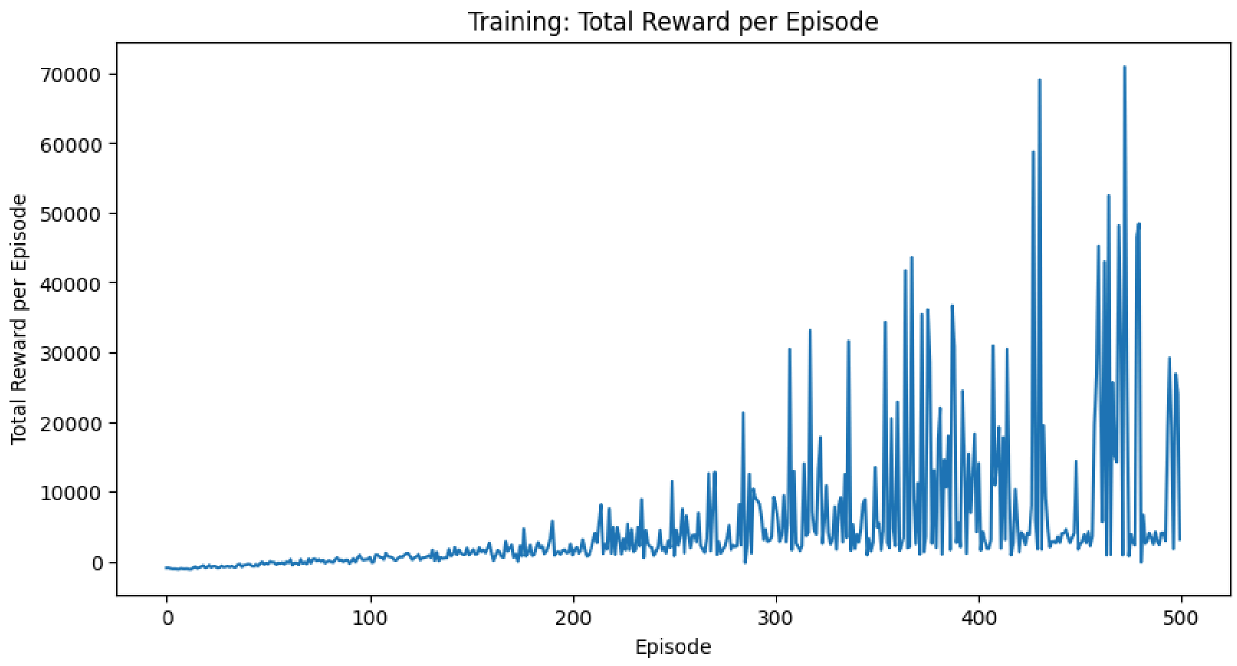
Training Phase

Epsilon Decay:



The epsilon decay plot shows the exploration rate (ϵ) decreasing exponentially over 500 episodes. Initially, the agent explores the environment extensively with a high epsilon value (close to 1). Over time, as learning progresses, epsilon decreases and the agent increasingly exploits the learned policy by taking more greedy actions.

Total Reward per Episode:



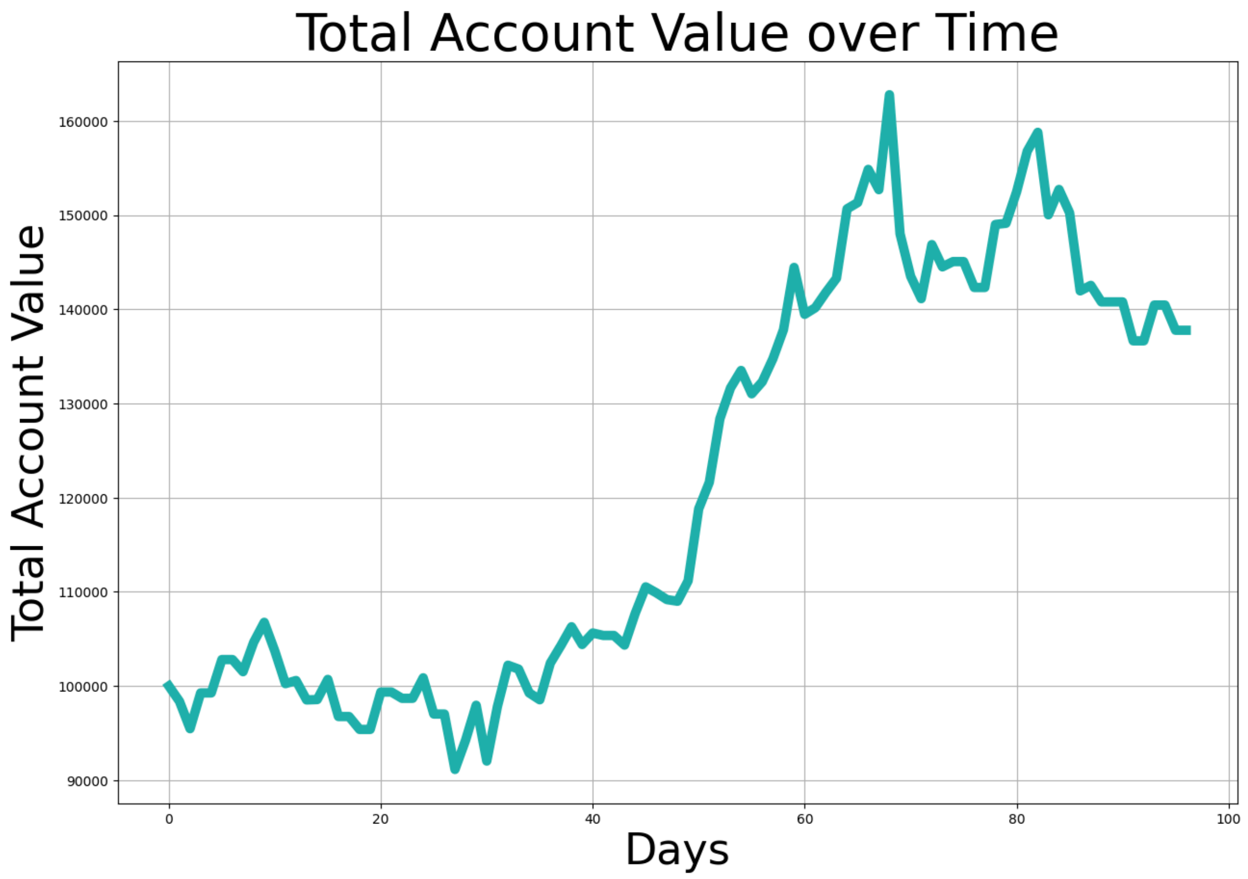
The total reward per episode increases significantly as training progresses. Early episodes show low rewards due to random exploration and lack of knowledge about the environment. However, as the agent learns an optimal policy, rewards improve, with noticeable spikes in later episodes. This indicates that the agent has successfully learned a strategy to maximize returns.

(The **number_of_days_to_consider** hyperparameter is set to **8** for both training and testing the agent, as this value produced the best results.)

Evaluation Phase

Account Value Over Time:

The evaluation plot demonstrates the agent's performance when trained and tested using a greedy policy (i.e., selecting actions with maximum expected rewards). The total account value starts at around $100,000 and shows consistent growth over time, peaking above $160,000 before experiencing some fluctuations.



Despite some volatility, the overall trend is positive, indicating that the Q-learning algorithm has enabled the agent to make profitable trading decisions. The evaluation results highlight that the trained agent can perform well in a simulated trading environment, achieving substantial growth in account value while adapting to market dynamics.

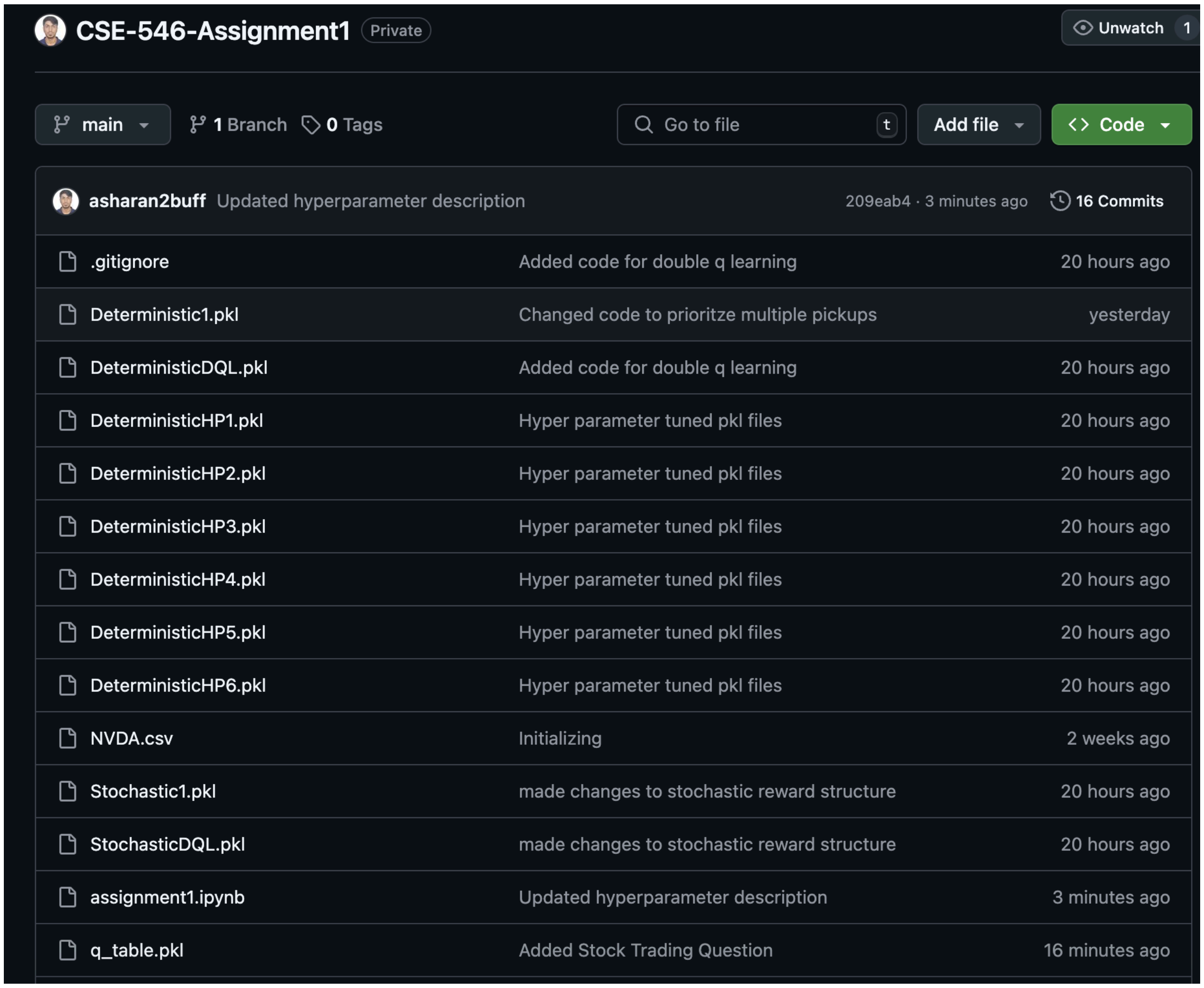
Conclusion

The Q-learning algorithm demonstrates its effectiveness in solving the stock trading problem by enabling an agent to learn optimal trading strategies through trial and error. The results show promising potential for applying reinforcement learning techniques in financial decision-making tasks.

Github Repository Link :

https://github.com/asharan2buff/CSE-546-Assignment1

Folder Structure :

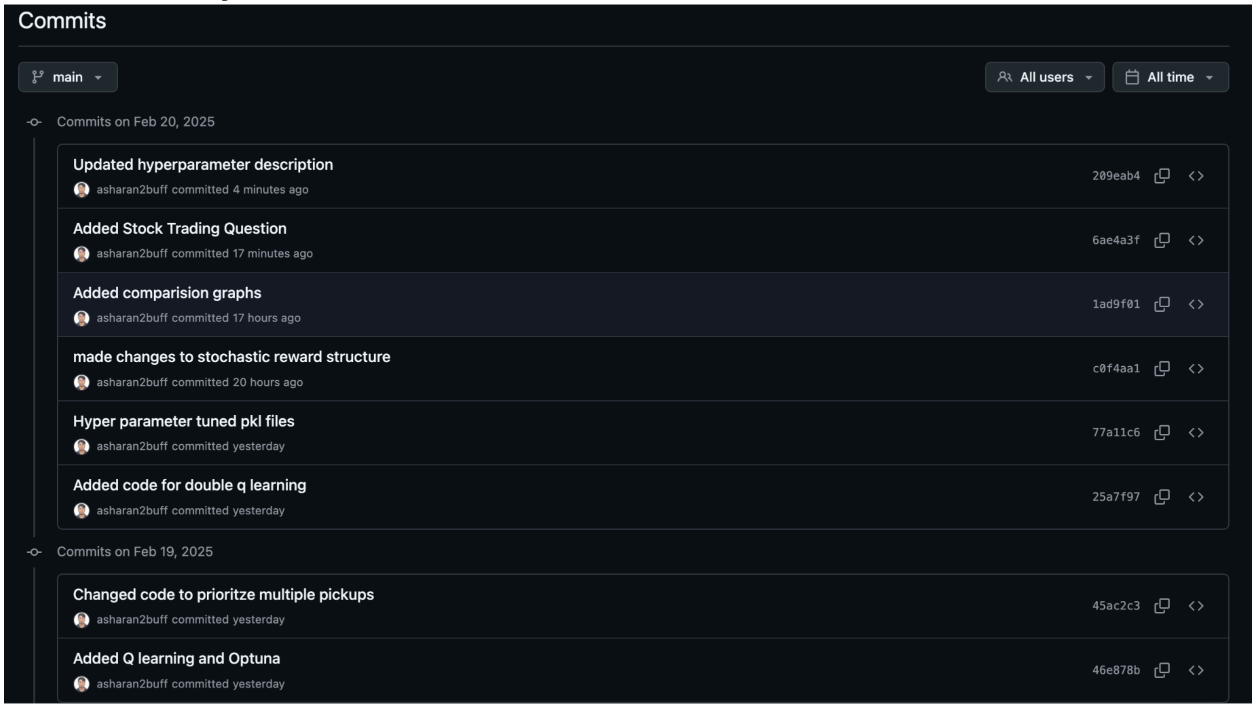


CSE-546-Assignment1 Private

asharan2buff Updated hyperparameter description 209eab4 · 3 minutes ago 16 Commits

.gitignore	Added code for double q learning	20 hours ago
Deterministic1.pkl	Changed code to prioritze multiple pickups	yesterday
DeterministicDQL.pkl	Added code for double q learning	20 hours ago
DeterministicHP1.pkl	Hyper parameter tuned pkl files	20 hours ago
DeterministicHP2.pkl	Hyper parameter tuned pkl files	20 hours ago
DeterministicHP3.pkl	Hyper parameter tuned pkl files	20 hours ago
DeterministicHP4.pkl	Hyper parameter tuned pkl files	20 hours ago
DeterministicHP5.pkl	Hyper parameter tuned pkl files	20 hours ago
DeterministicHP6.pkl	Hyper parameter tuned pkl files	20 hours ago
NVDA.csv	Initializing	2 weeks ago
Stochastic1.pkl	made changes to stochastic reward structure	20 hours ago
StochasticDQL.pkl	made changes to stochastic reward structure	20 hours ago
assignment1.ipynb	Updated hyperparameter description	3 minutes ago
q_table.pkl	Added Stock Trading Question	16 minutes ago

Commit History:



Commits

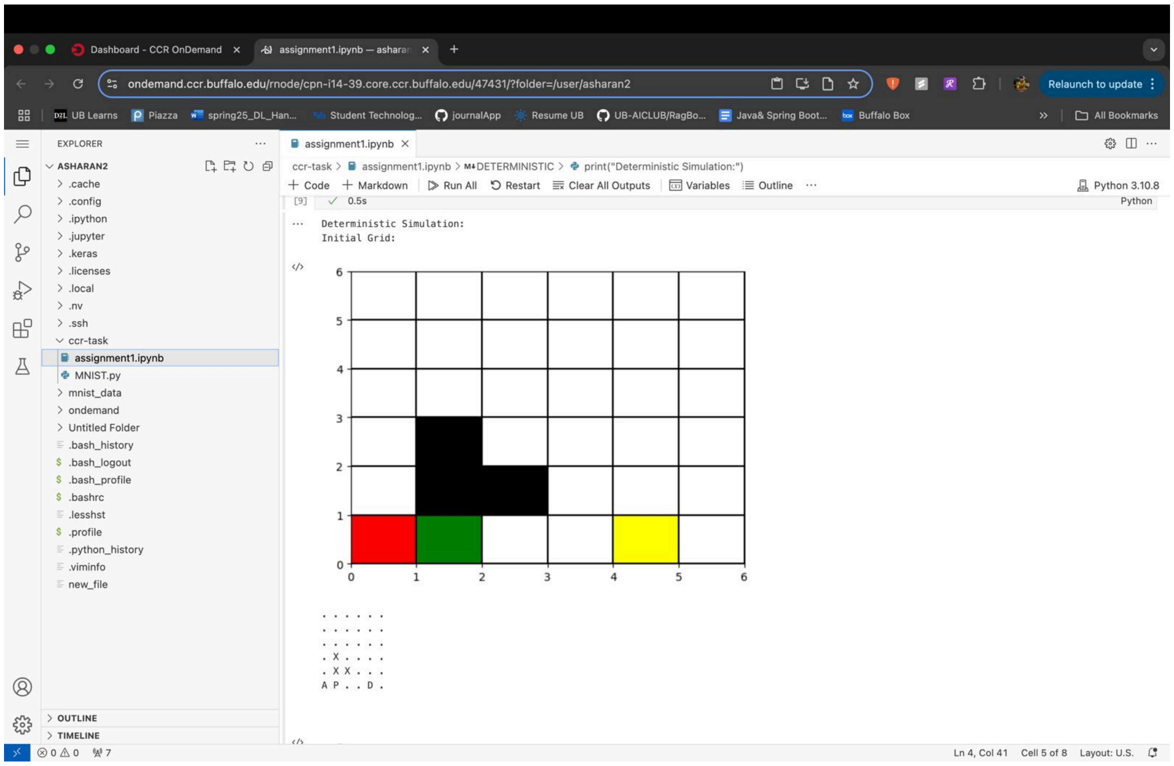
Commits on Feb 20, 2025

Updated hyperparameter description asharan2buff committed 4 minutes ago	209eab4
Added Stock Trading Question asharan2buff committed 17 minutes ago	6ae4a3f
Added comparision graphs asharan2buff committed 17 hours ago	1ad9f01
made changes to stochastic reward structure asharan2buff committed 20 hours ago	c0f4aa1
Hyper parameter tuned pkl files asharan2buff committed yesterday	77a11c6
Added code for double q learning asharan2buff committed yesterday	25a7f97

Commits on Feb 19, 2025

Changed code to prioritze multiple pickups asharan2buff committed yesterday	45ac2c3
Added Q learning and Optuna asharan2buff committed yesterday	46e878b

CCR Submission :



```
Deterministic Simulation:
Initial Grid:
```

```
. . . . . .
. . . . . .
. . . . . .
. X . . . .
. X X . . .
A P . . D .
```