Report – Assignment 1
Assignment 1 - Defining & Solving RL Environments
Anip Kumar Paul, September 12, 2024

Assignment: In this assignment, I am defining and solving a reinforcement learning task for set up the environments for the "Warehouse Robot" problem by following the Gymnasium standards. In the first part of this assignment, I have defined the deterministic and stochastic environment based on the Markov Decision Process. I have described in detail below:

Part 1: Defining RL Environments

In this Part, I have defined the environment for the Warehouse Robot. The goal of this task is to enable the robot to move around the warehouse grid by avoiding the obstacle shelves pick up an item from the pre-defined location and deliver to the pre-designated drop-off location. The dimension of the warehouse is 6x6 grid space. The robot will follow the six possible actions: Up, Down, Left, Right, Pick-up, Drop-off and there is static obstacle (shelves) that should be avoided.

1. **Environment Type:**

I worked on two types of environments, Deterministic and Stochastic environment.

For deterministic environment, all action by the robot lead to predictable outcomes where $P(s', r \mid s, a) = \{0, 1\}$. For example, if the robot takes any action, it always follows that action to the end up in the adjacent grid cell. The pick-up and drop-off actions are deterministic, resulting in either success or failure based on the robot's location and item possession state.

For Stochastic environment, the robot follows the probabilistic outcomes, which mean if the agent takes the action to go right, the agent ends up in the grid block on the right with a probability of 0.9 (90%), but stays in the same grid block with a probability of 0.1 (10%), where $\sum_{s',r} P(s', r|s, a) = 1$. The pick-up and drop-off actions remain deterministic, but the movement actions are affected by probability.

Environment Setup:

Grid position: I have setup my environment as stated in the assignment. The grid size of the environment is 6x6. I have set up the initial position of the agent is [0, 0] in the grid space. I have kept the item in location [2, 2] from where the agent will pick-up the item, and the goal position is [5, 5] where the agent should drop-off the item. In my environment, I have put several obstacles in the path of the robot and location of the obstacles are: (1, 4), (1, 2), (2, 1), (2, 5), (3, 3), (4, 2), (4, 4), (5, 0).

Actions: In my environment robot will follow six actions which are Move Down (action = 0), Move Up (action = 1), Move Right (action = 2), Move Left (action = 3), Pick Up (action = 4), and Drop Off (action = 5.

States: As the grid size of my task environment is 6x6 so there is 36 possible positions with two other states as the agent has picked up the item or not. So total state space is 72.

Terminal state: In this environment there is only one terminal state which is the goal position and the robot will drop off the item to the goal position and will end the episode.

Rewards: In my environment robot will get reward with 20 points for successfully delivering the item to the goal position, -1 point penalty for each step taken to encourage efficiency, -20 points penalty for hitting an obstacle and I have added another extra penalty for -10 points f or failed pick-up or drop-off attempts. The robot may take up to 15 timesteps to pick up the item and drop off the item to the goal position.

Objective: the main objective of this task is the robot must have to learn an optimal policy that allows it to pick up the item and deliver it to the goal position in the minimum number of steps by avoiding obstacles. This is achieved by maximizing cumulative rewards using a Q-learning algorithm.

2. The visualizations of the environments

I have displayed here my environment setup with the initial position is in yellow color block, terminal position with the blue color block and the obstacle shelves is in the environment with the deep purple color block.

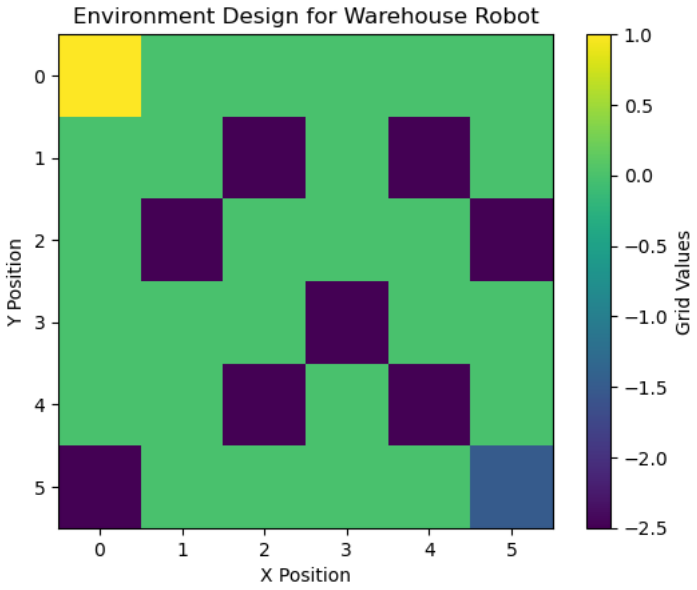


Figure 1: Environment set up for Warehouse Robot.

After set up all of the function and position with the item position, the environment looks like:

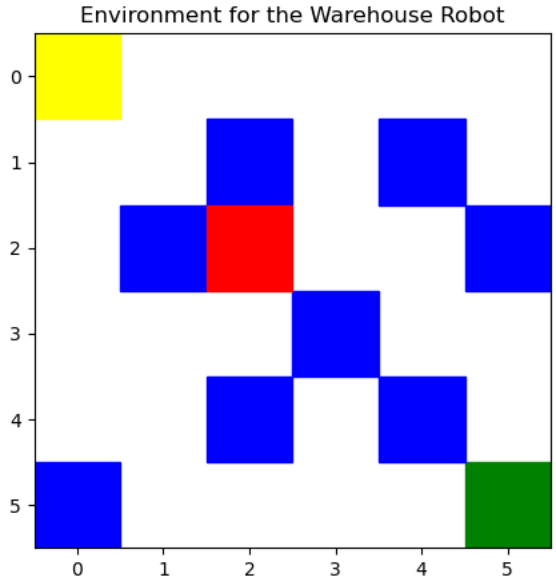


Figure 2: Environment with item (red block) for Warehouse Robot.

Robot movement position: I have demonstrated the environment which shows the successful movement of the robot.

```
action = 0
observation, reward, done, truncated, info = env.step(action)
env.render()

action = 2
observation, reward, done, truncated, info = env.step(action)
env.render()
```

The action = 0 and 2 mean that the robot will go down and right, respectively, shown in Figure 3 and Figure 4.

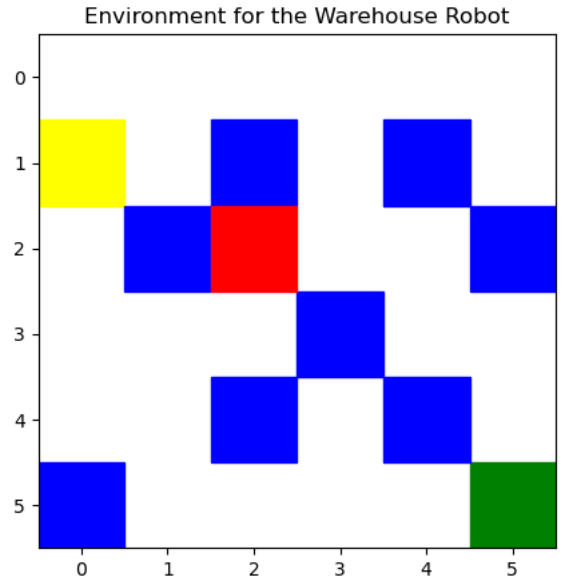


Figure 3: Robot movement (action = down).

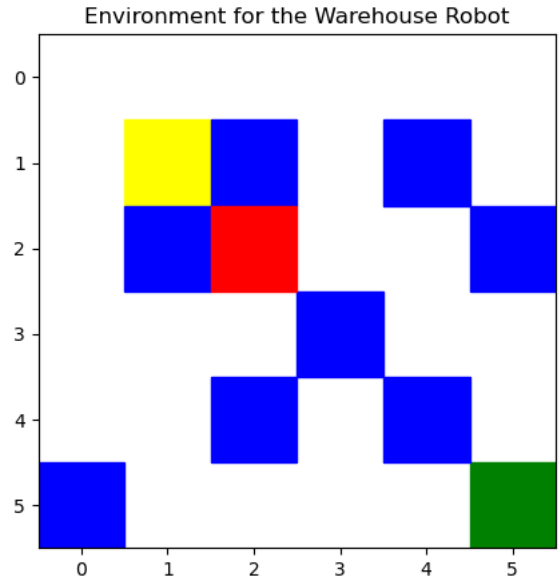
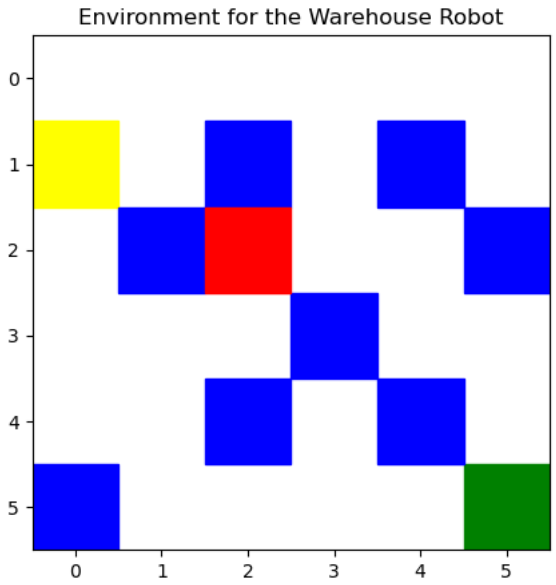


Figure 4: Robot movement (action = right).

```
action = 3
observation, reward, done, truncated, info = env.step(action)
env.render()

action = 1
observation, reward, done, truncated, info = env.step(action)
env.render()
```

The action = 3 and 1 mean that the robot will go left and up, respectively, shown in Figure 5 and Figure 6.



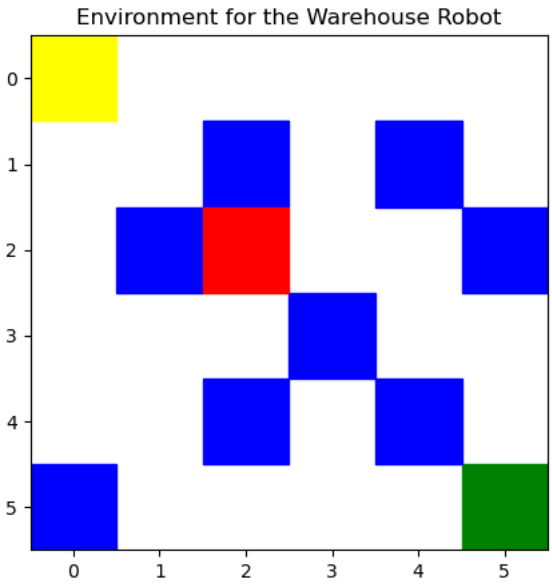
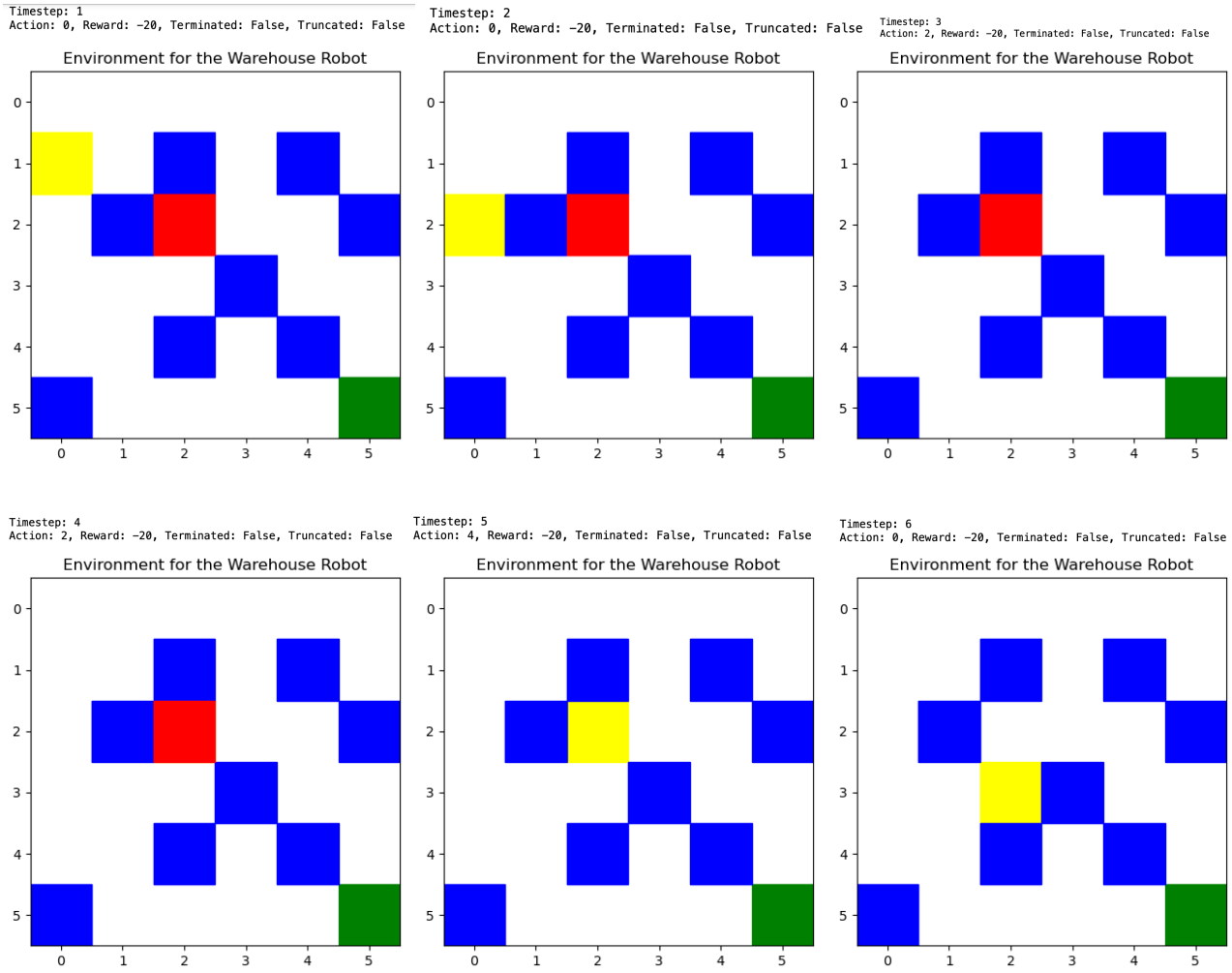


Figure 3: Robot movement (action = left). Figure 4: Robot movement (action = up).

Design the Random Agent for Deterministic Environment



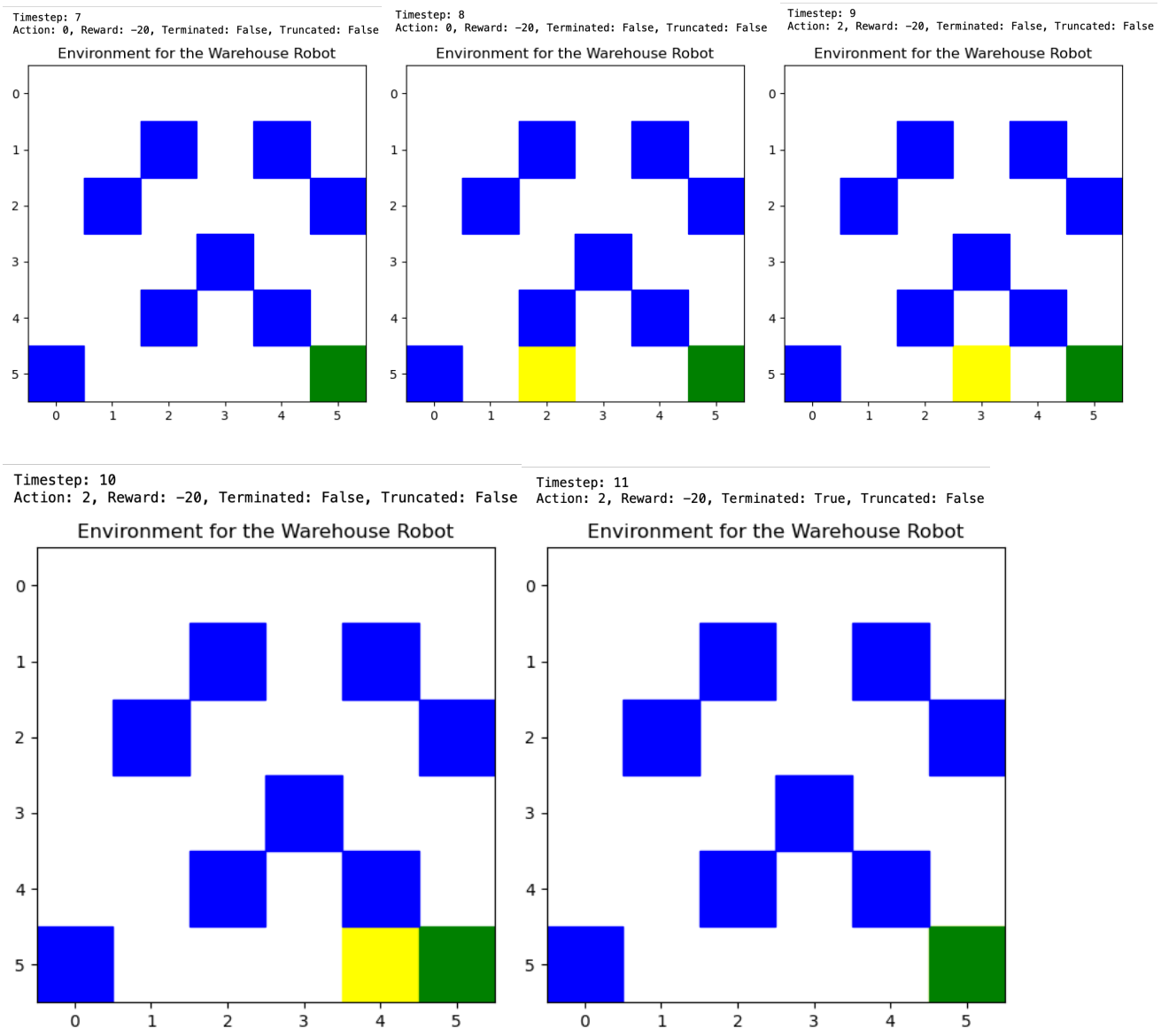
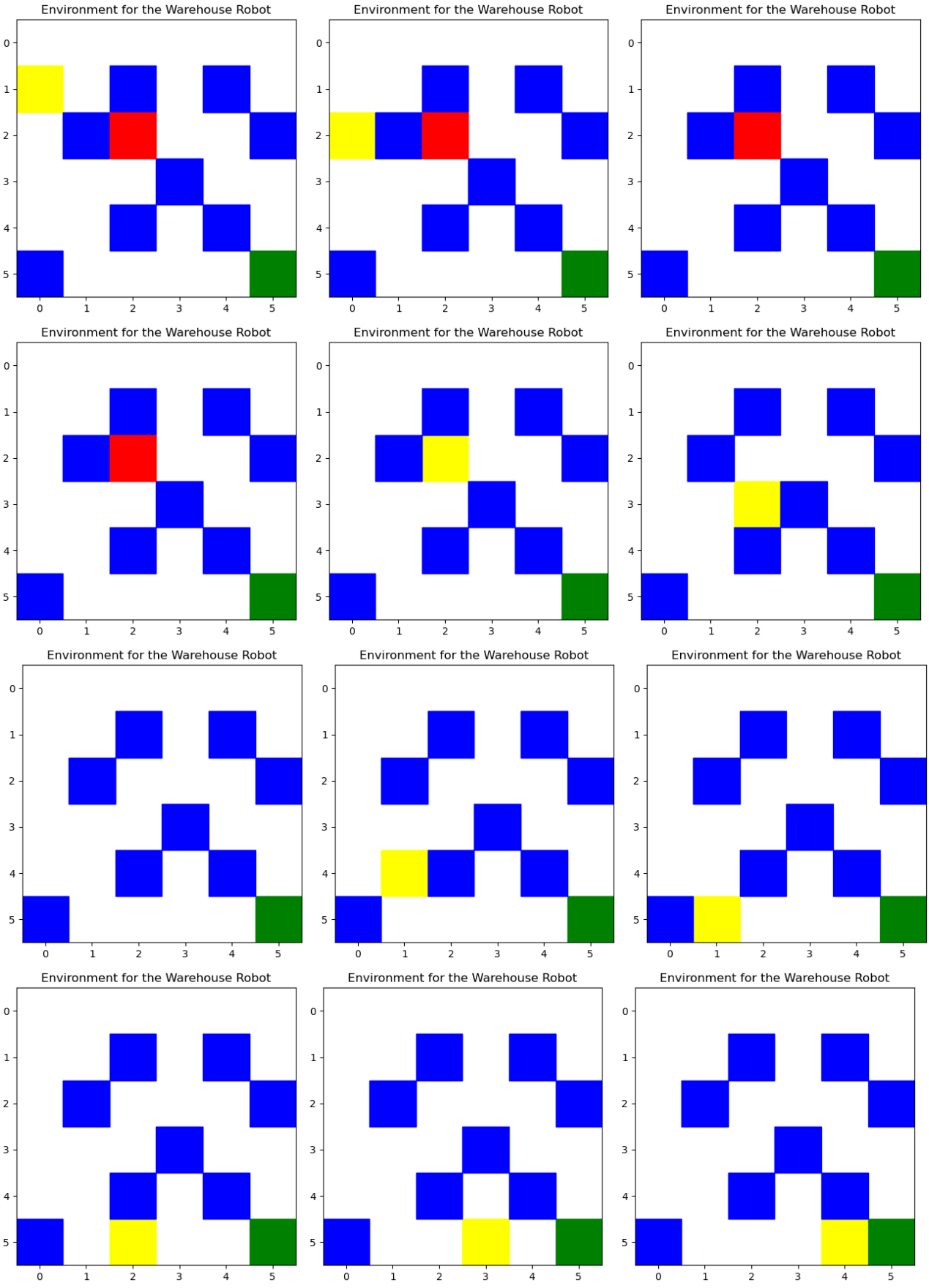


Figure 5: Robot movement for the deterministic environment.

3. Design the Random Agent for stochastic Environment

```
if (self.env_type == 'Stochastic'):
    action_prob = np.random.random()
    if action in [0, 1, 2, 3]:
        if action_prob <= 0.9:  #  I consider the 90% chance to take the chosen actio
            chosen_action = action
        else:
        # consider the 10% chance to choose a random action
            possible_actions = [0, 1, 2, 3]  # This is my all possible actions
            possible_actions.remove(action)
            chosen_action = np.random.choice(possible_actions)
    else:
    # Deterministic behavior
        chosen_action = action
```

I consider the 90% chance to take the chosen action. Consider the 10% chance to choose a random action.


Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot
Environment for the Warehouse Robot

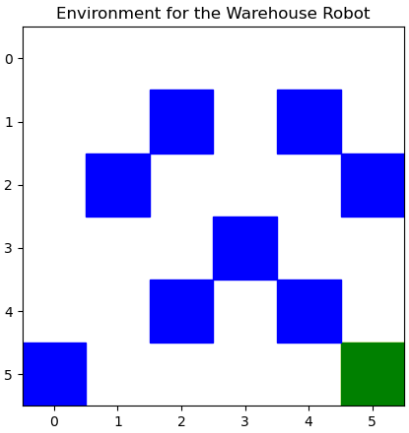


Figure 6: Robot movement for the Stochastic environment.

4. Difference between the deterministic and stochastic environments:

In a deterministic environment, the outcome from the environment for any action is always predictable and consistent which represents that when an agent takes an action, the result will always be the same. For example, if a robot moves "up" on a grid, it will move exactly one cell up every time without any variation. However, a stochastic environment represents the uncertainty, where the outcome of an action can vary even if the same action is repeated. For example, a robot attempting to move "up" might succeed most of the time, but occasionally it could slip and move in a different direction due to randomness. In a stochastic environment, outcomes are not entirely predictable, making it more challenging for the agent to learn and adapt. While deterministic environments are easier for agents to navigate and learn from, stochastic environments more accurately reflect real-world situations, such as stock trading, where randomness and unpredictability are common factors.

5. Write a brief review (∼ 5 sentences) explaining how you ensure the safety of your environments.

To keep my environment safe, I make sure the agent stays within the allowed space and only takes valid actions. If the agent tries to move outside the boundaries or do something that isn't stated, the system stops it and set back to the initial position. For example, the agent can't move through obstacles or pick up or drop items in the wrong place. If the agent tries to do something wrong, it gets a penalty, which helps it learn to make better decisions and stay safe. This way, the agent follows the rules and learns safely.

Part 2

Q-Learning

In this assignment, I implemented the Q-learning algorithm to solve deterministic and stochastic environments. In the deterministic environment, every action taken by the agent had a predictable outcome, allowing the agent to learn a direct and clear path to maximize rewards. In the stochastic environment, actions had some randomness, so the agent had to adapt to uncertain outcomes.
Here, I have shown the Q-Table for deterministic environment Q-Table after training 1000 episodes)

Environment Type: Deterministic

Q-table:

	DOWN	UP	RIGHT	LEFT	PICK	DROP
S0	-139.17	-140.02	-139.12	-143.08	-141.17	-143.05
S1	-121.30	-122.86	-121.33	-134.53	-122.23	-123.49
S2	-103.93	-104.45	-104.24	-110.18	-105.12	-104.71
S3	-88.75	-89.04	-88.86	-91.26	-89.64	-88.46
S4	-75.38	-75.79	-75.57	-76.20	-76.02	-74.75
S5	-65.28	-64.94	-64.25	-66.13	-65.54	-64.27
S6	-121.52	-135.39	-121.40	-121.66	-122.53	-121.78
S7	-104.86	-111.09	-104.67	-116.03	-107.36	-108.06
S8	-87.67	-97.84	-88.41	-98.42	-87.82	-88.56
S9	-76.61	-79.12	-76.37	-77.53	-76.67	-78.19
S10	-65.20	-65.54	-65.82	-66.71	-66.61	-65.48
S11	-56.73	-56.21	-56.84	-57.79	-57.09	-57.76
S12	-104.51	-114.64	-104.39	-106.92	-105.39	-104.35
S13	-88.13	-97.62	-88.12	-94.41	-88.67	-90.37

Figure 7: Q – table for deterministic env.

After training the model for 5000 episodes, I have got new trained Q-Table and I have presented below that table for both deterministic and stochastic environment. I have also attached the .h5 file for both Q table with the assignment package.

Environment Type: Deterministic

Q-table:

	DOWN	UP	RIGHT	LEFT	PICK	DROP
S0	-189.80	-204.11	-189.80	-204.11	-204.11	-204.11
S1	-175.05	-189.80	-175.05	-204.11	-189.80	-189.80
S2	-159.85	-175.05	-189.80	-189.80	-175.05	-175.05
S3	-175.05	-189.80	-204.11	-175.05	-189.80	-189.80
S4	-189.80	-204.11	-217.98	-189.80	-204.11	-204.11
S5	-204.11	-217.98	-217.98	-204.11	-217.98	-217.98
S6	-175.05	-204.11	-175.05	-189.80	-189.80	-189.80
S7	-159.85	-189.80	-159.85	-189.80	-175.05	-175.05
S8	-144.17	-175.05	-175.05	-175.05	-159.85	-159.85
S9	-159.85	-189.80	-189.80	-159.85	-175.05	-175.05
S10	-175.05	-204.11	-204.11	-175.05	-189.80	-189.80
S11	-189.80	-217.98	-204.11	-189.80	-204.11	-204.11
S12	-189.80	-189.80	-159.85	-175.05	-175.05	-175.05
S13	-175.05	-175.05	-144.17	-175.05	-159.85	-159.85

(a)

Environment Type: Stochastic

Q-table:

	DOWN	UP	RIGHT	LEFT	PICK	DROP
S0	-182.81	-197.98	-186.95	-211.77	-210.64	-202.77
S1	-198.07	-207.45	-171.82	-201.89	-201.46	-193.26
S2	-162.69	-189.23	-207.74	-188.93	-188.64	-182.13
S3	-176.61	-202.90	-214.92	-207.53	-201.17	-207.45
S4	-211.74	-215.18	-219.46	-210.52	-216.19	-218.24
S5	-218.43	-219.18	-216.24	-216.33	-215.34	-215.71
S6	-207.16	-210.64	-193.16	-181.41	-205.31	-205.46
S7	-159.51	-198.24	-204.20	-203.80	-202.14	-204.66
S8	-155.34	-201.74	-201.43	-168.02	-193.17	-168.13
S9	-201.20	-188.19	-210.31	-153.66	-204.37	-199.33
S10	-206.89	-212.20	-216.46	-199.04	-206.15	-206.22
S11	-216.63	-213.71	-217.73	-204.55	-218.73	-219.25
S12	-211.03	-211.25	-205.20	-208.56	-204.83	-211.09
S13	-208.27	-190.68	-157.32	-197.58	-202.29	-195.29

(b)

Figure 8: (a) After training the Q – table for deterministic env and (b) for Stochastic env.

Here, I have demonstrated the Rewards per episode graph for Deterministic Environment for the Q learning Algorithm in figure 9.

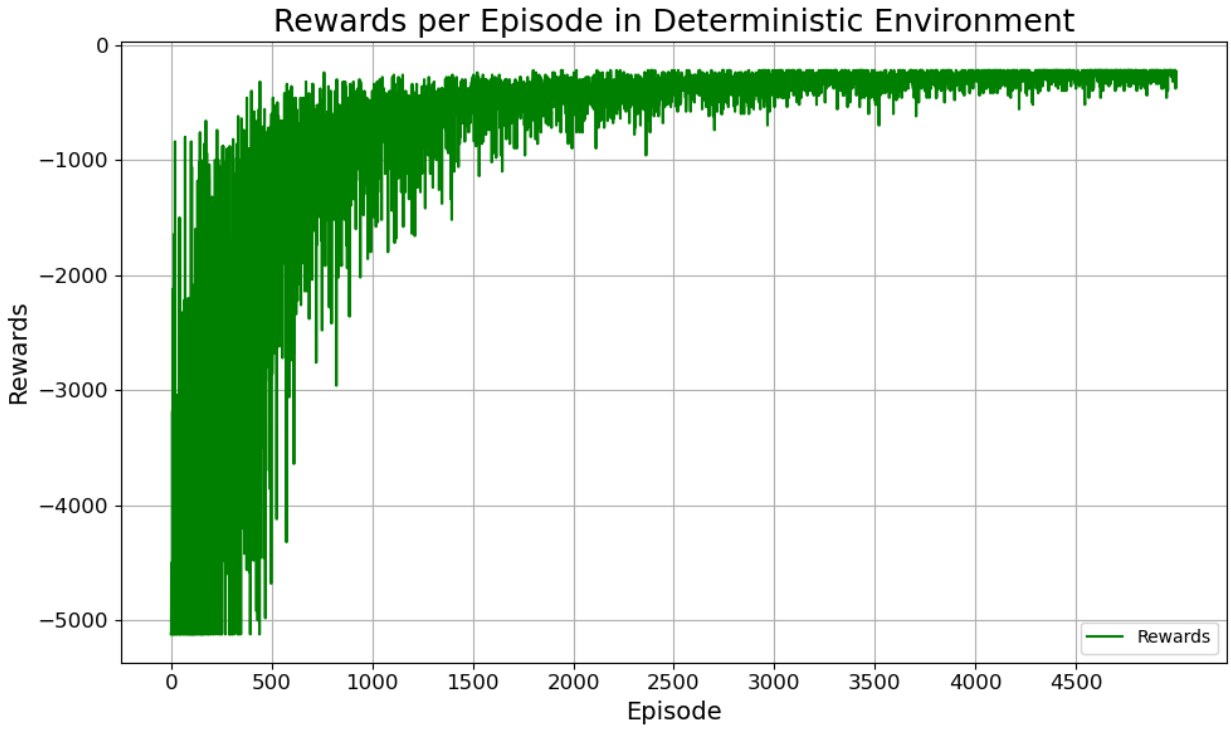


Figure 9: Rewards per episode graph for Deterministic Environment.

I have presented the Epsilon decay curve per episode for Deterministic Environment for the Q learning Algorithm in figure 10.

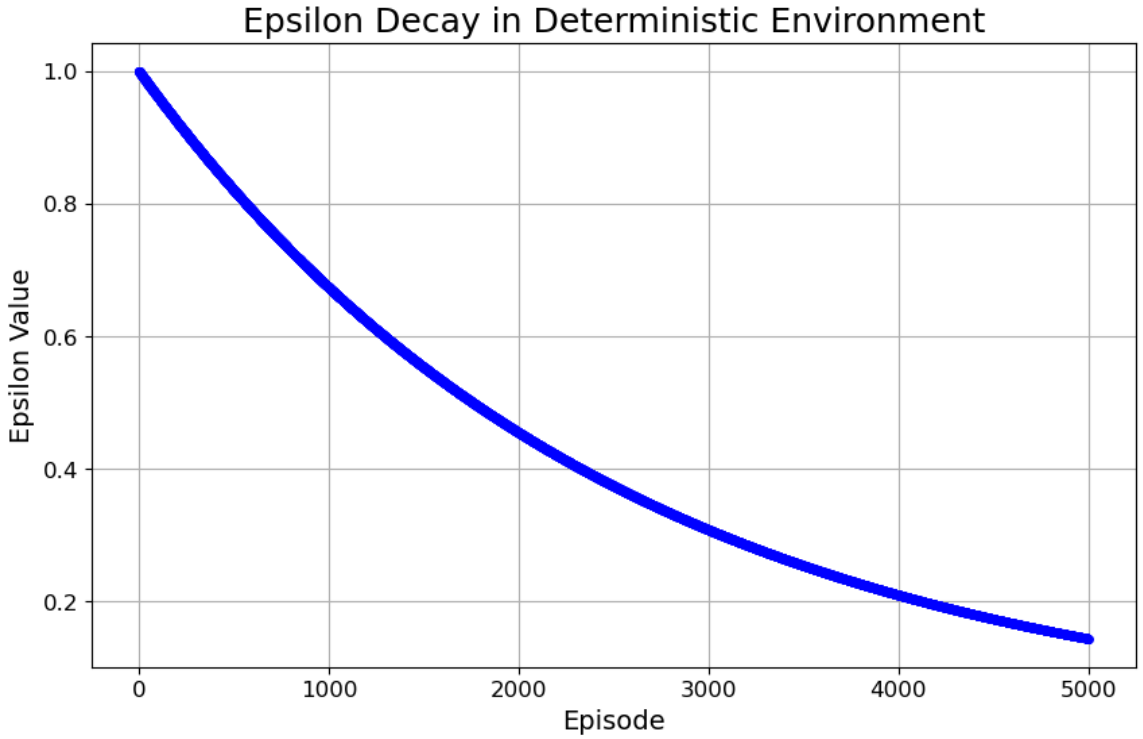


Figure 10: Epsilon decay graph per episode in Deterministic Environment

Here, I have demonstrated the Rewards per episode graph for Stochastic Environment for the Q learning Algorithm in figure 11.

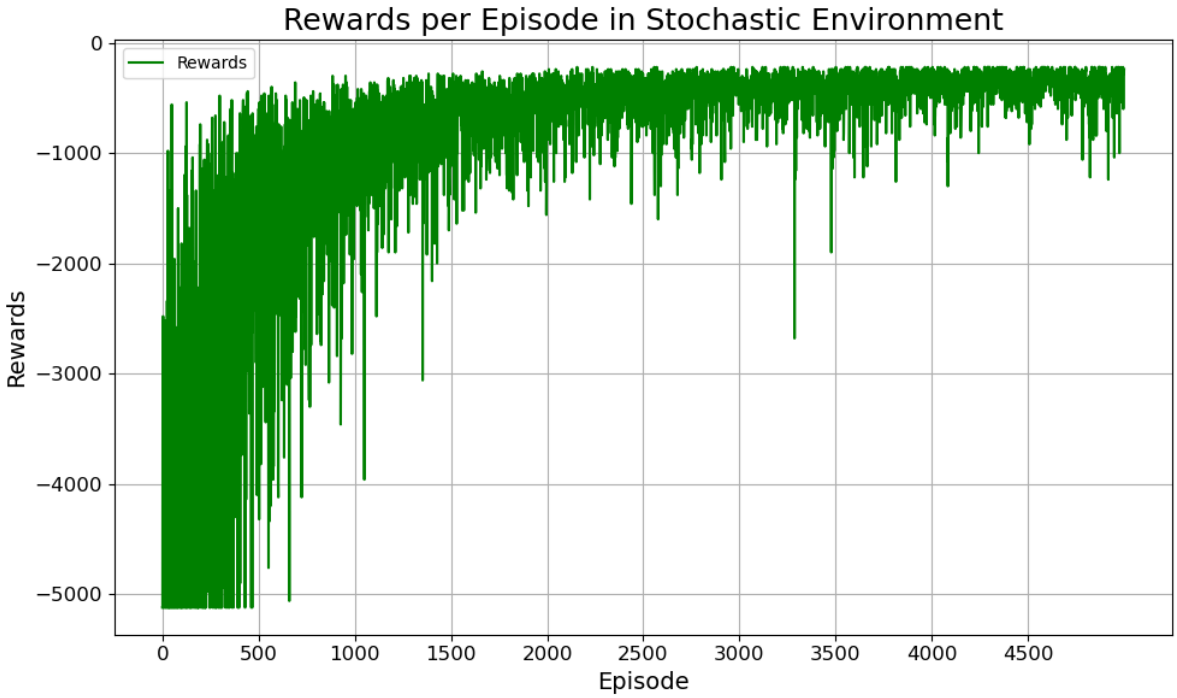


Figure 11: Rewards per episode in Stochastic Environment

I have presented the Epsilon decay curve per episode for Stochastic Environment for the Q learning Algorithm in figure 12.

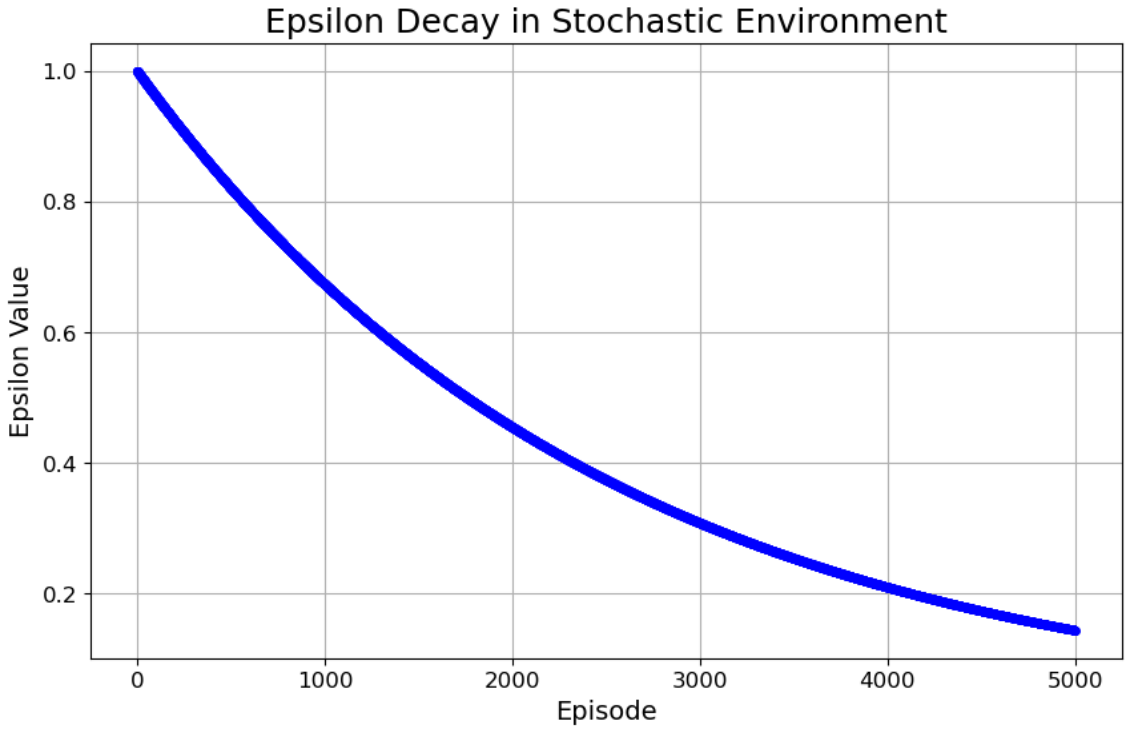


Figure 11: Epsilon decay graph per episode in Stochastic Environment

In Q learning algorithm, I have performed the Greedy Action technique for deterministic and stochastic algorithm, and the corresponding figures are in the python file (I have not put here as it is difficult to maintain the sequence of the graph).

I have conducted the hyperparameter tuning for two different hyperparameter one is decay rate, and another is discount rate. I have presented here the nine separate combination of those two parameters and presented the graph below, Figure 12.

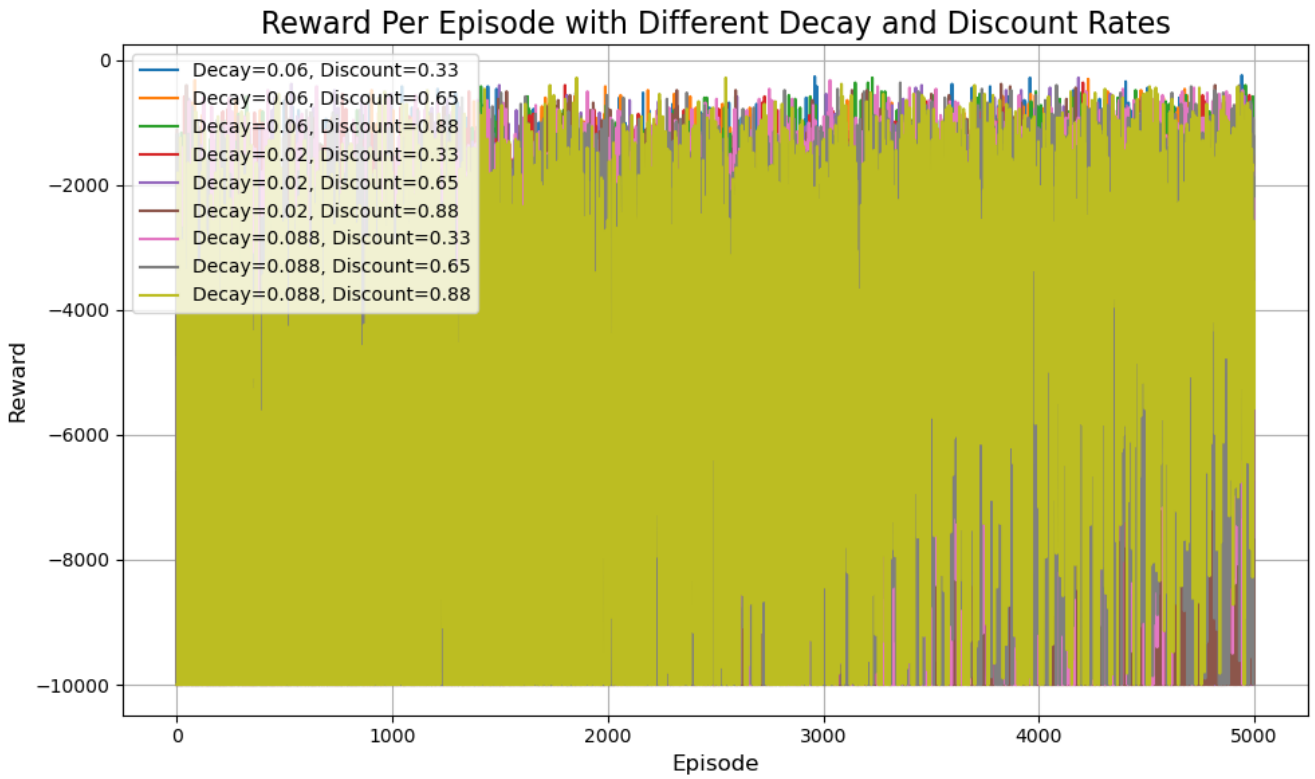


Figure 12: Reward Per Episode with two different hyperparameter Decay and Discount Rates

2.2 Apply the SARSA method

From the other TD method, I choose SARSA method and perform serval determination for my environment. I have shown the Q-Table for deterministic environment for the SARSA method. Q-Table after training 1000 episodes)

Environment Type: Deterministic

Q-table:

	DOWN	UP	RIGHT	LEFT	PICK	DROP
S0	-145.05	-152.64	-143.64	-154.28	-151.54	-154.66
S1	-125.08	-132.07	-125.04	-146.89	-129.08	-128.29
S2	-106.83	-109.30	-107.08	-122.95	-111.81	-109.30
S3	-91.02	-91.46	-91.13	-93.51	-92.90	-91.30
S4	-76.24	-79.27	-77.00	-79.66	-79.28	-78.07
S5	-65.83	-66.72	-65.15	-66.61	-67.39	-67.92
S6	-124.33	-148.81	-124.27	-126.62	-126.95	-127.08
S7	-107.20	-112.70	-107.34	-126.49	-107.20	-111.09
S8	-89.87	-94.51	-89.97	-100.80	-90.09	-90.72
S9	-77.18	-80.31	-77.55	-85.05	-78.91	-78.54
S10	-65.81	-65.85	-65.27	-66.60	-66.79	-67.59
S11	-56.87	-57.50	-59.97	-56.94	-59.27	-59.80
S12	-106.69	-122.02	-106.60	-110.75	-109.13	-109.12
S13	-90.31	-102.30	-90.06	-96.33	-92.37	-91.93

Figure 13: Q-Table for the SARSA method for the Deterministic Environment

After training the model for 5000 episodes for SARSA method, I have got new trained Q-Table and I have presented below that table for both deterministic and stochastic environment. I have also attached the .h5 file for both Q table with the assignment package.

Q-Table for the SARSA method for the Deterministic and Stochastic Environment after trained on 5000 episodes

Environment Type: Deterministic

Q-table (SARSA):

	DOWN	UP	RIGHT	LEFT	PICK	DROP
S0	-300.01	-295.19	-249.15	-287.62	-292.36	-283.90
S1	-288.92	-286.65	-248.21	-296.25	-291.73	-273.33
S2	-248.33	-272.96	-321.11	-282.70	-244.84	-263.92
S3	-300.29	-258.52	-364.95	-246.45	-338.12	-303.23
S4	-363.64	-511.06	-543.87	-371.39	-400.62	-484.94
S5	-557.04	-539.50	-495.07	-388.05	-550.17	-567.13
S6	-297.39	-268.63	-331.40	-288.88	-321.12	-309.09
S7	-204.85	-290.17	-274.59	-279.49	-278.57	-287.58
S8	-251.61	-243.65	-263.71	-247.65	-229.32	-239.22
S9	-355.02	-283.37	-346.68	-263.60	-305.23	-341.28
S10	-263.90	-411.03	-501.67	-420.09	-308.42	-397.06
S11	-544.41	-547.03	-432.96	-370.06	-466.94	-552.66
S12	-355.17	-322.26	-191.33	-315.42	-323.24	-323.00
S13	-250.82	-273.03	-167.53	-238.23	-275.58	-272.12

(a)

Environment Type: Stochastic

Q-table (SARSA):

	DOWN	UP	RIGHT	LEFT	PICK	DROP
S0	-239.87	-302.04	-320.04	-312.30	-308.12	-305.61
S1	-253.20	-307.08	-309.04	-306.68	-309.83	-234.00
S2	-310.44	-320.83	-193.95	-307.74	-313.15	-318.92
S3	-286.93	-255.99	-316.33	-317.98	-343.55	-313.49
S4	-330.42	-322.46	-348.95	-316.73	-320.00	-337.47
S5	-356.84	-362.96	-367.12	-366.14	-364.63	-363.51
S6	-270.77	-311.58	-238.67	-293.08	-297.43	-302.16
S7	-222.80	-299.73	-282.61	-240.61	-298.34	-275.40
S8	-227.84	-307.69	-286.11	-281.96	-280.79	-298.29
S9	-299.73	-325.53	-318.41	-229.13	-312.57	-317.17
S10	-339.26	-339.10	-351.35	-312.95	-323.19	-351.21
S11	-366.70	-361.43	-356.09	-310.75	-344.84	-362.66
S12	-292.25	-314.77	-193.29	-306.63	-304.18	-297.56
S13	-244.10	-269.50	-189.87	-310.44	-218.44	-265.92

(b)

Figure 14: (a) After training the Q – table for deterministic env and (b) for Stochastic env.

Here, I have demonstrated the Rewards per episode graph for Deterministic Environment for the SARSA Algorithm in figure 15.

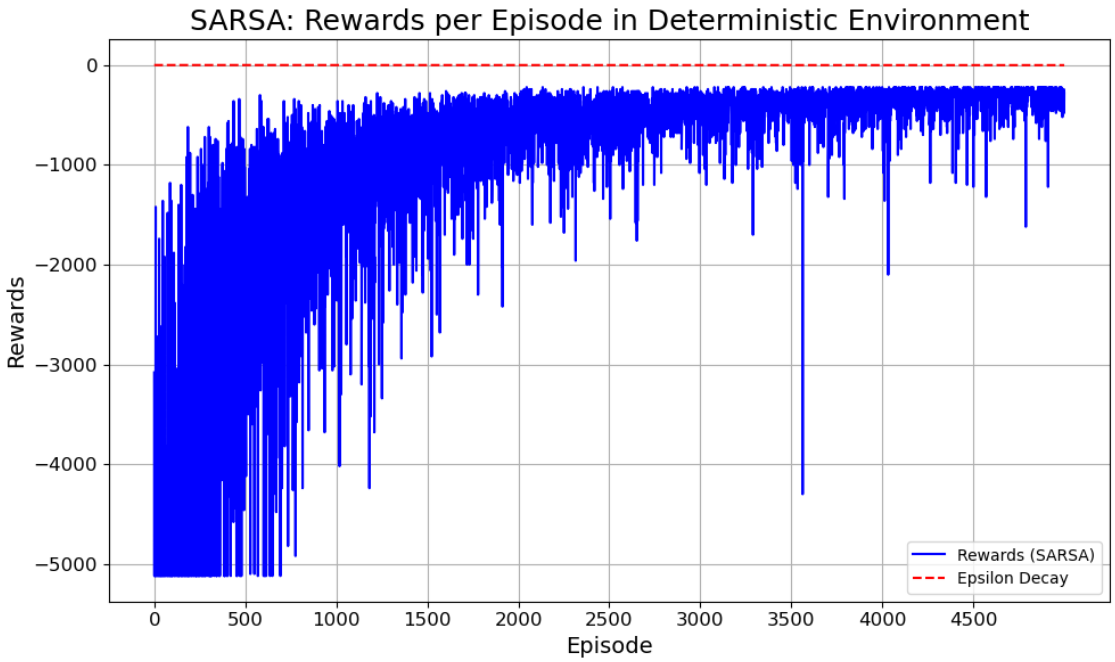


Figure 15: Rewards per episode in Deterministic Environment for the SARSA method

I have presented the Epsilon decay curve per episode for Deterministic Environment for the SARSA Algorithm in figure 16.

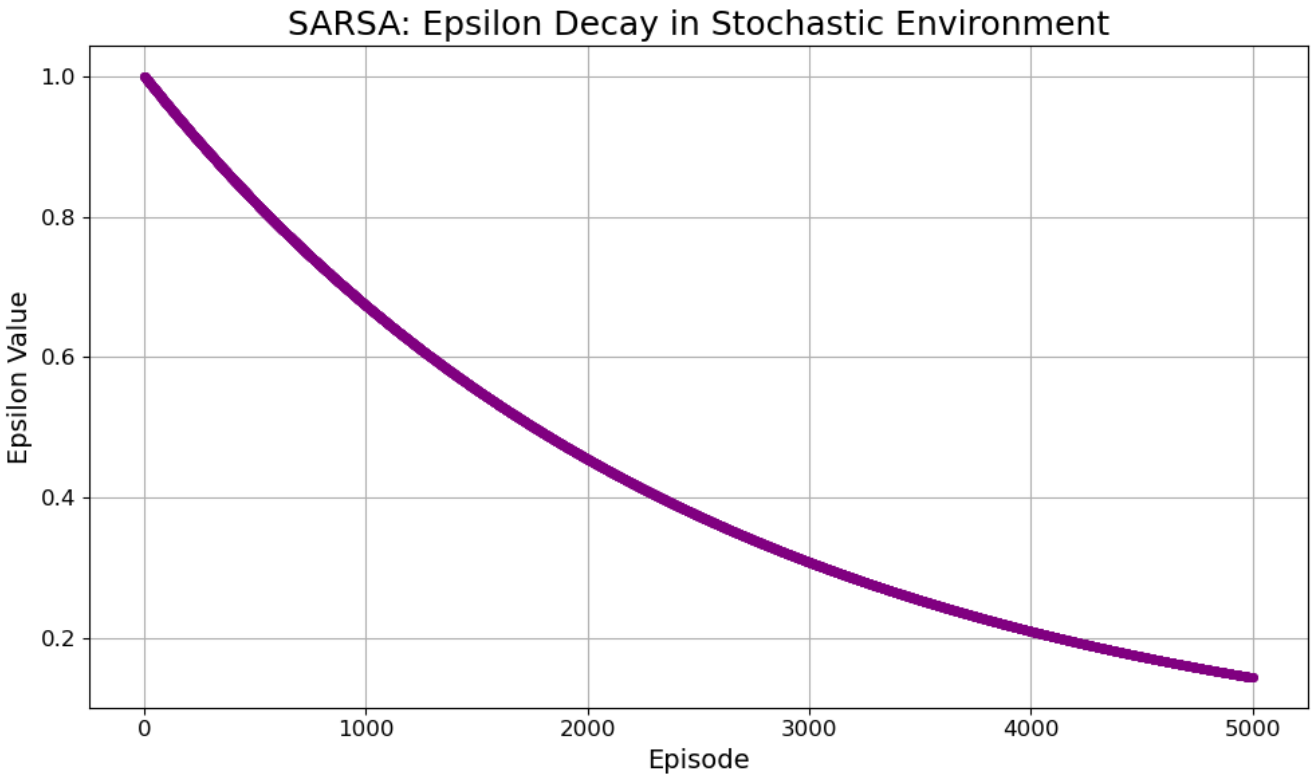


Figure 16: Epsilon decay per episode in Deterministic Environment for the SARSA method
Here, I have demonstrated the Rewards per episode graph for Stochastic Environment for the SARSA Algorithm in figure 17.

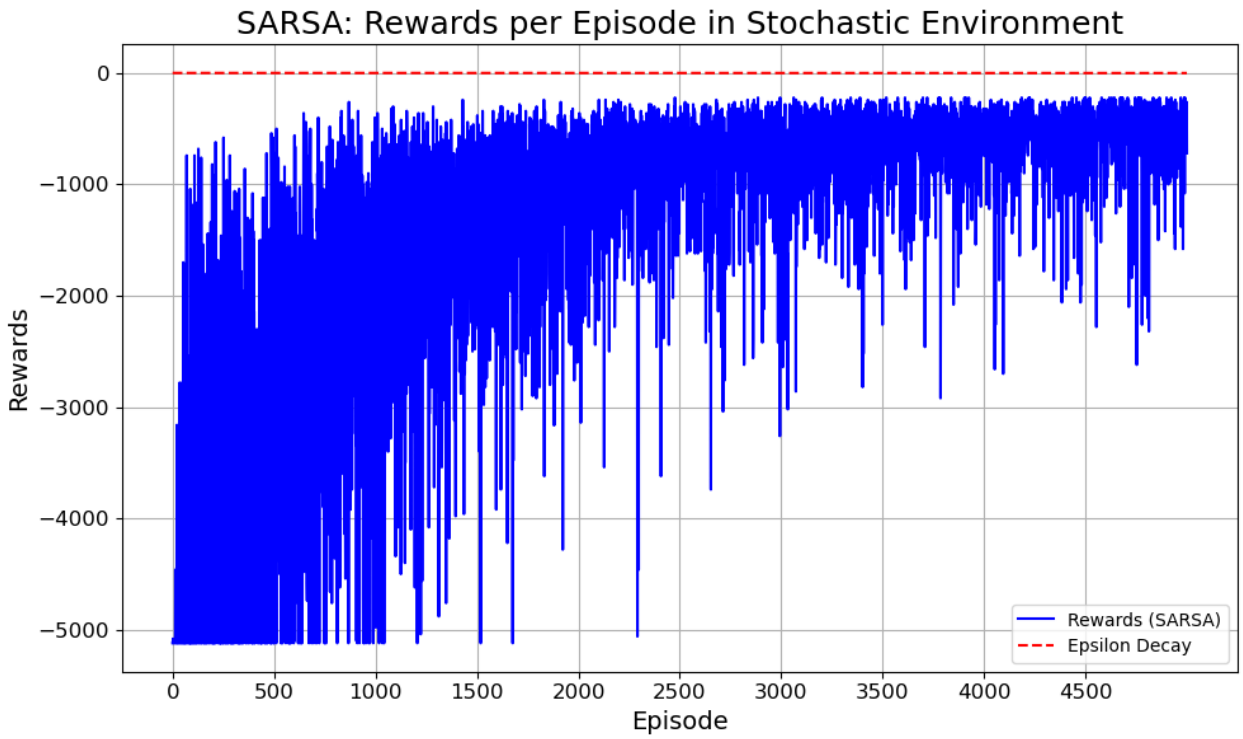


Figure 17: Rewards per episode in Stochastic Environment for the SARSA method

I have presented the Epsilon decay curve per episode for Stochastic Environment for the SARSA Algorithm in figure 18.

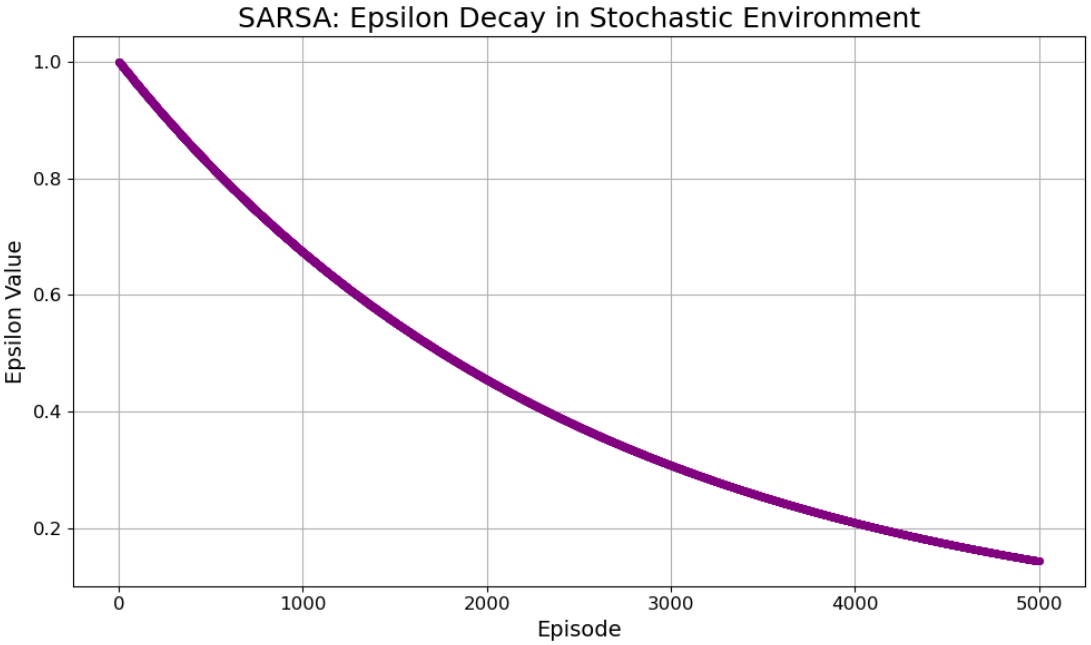


figure 18: Epsilon decay per episode in Stochastic Environment for the SARSA method

In SARSA algorithm, I have performed the Greedy Action technique for deterministic and stochastic algorithm, and the corresponding figures are in the python file (I have not put here as it is difficult to maintain the sequence of the graph).

I have conducted the hyperparameter tuning for two different hyperparameter one is decay rate, and another is discount rate. I have presented here the nine separate combination of those two parameters and presented the graph below, Figure 19.

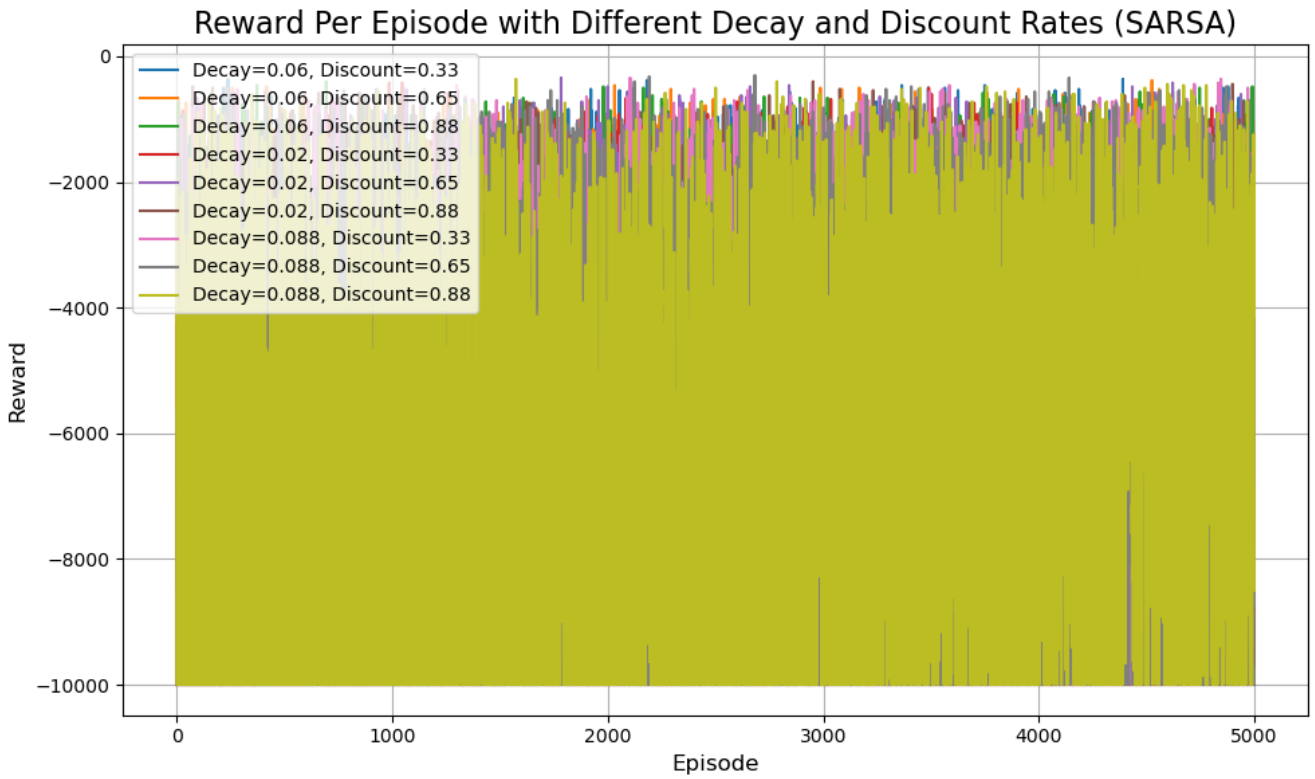


Figure19: Reward Per Episode with two different hyperparameter Decay and Discount Rates for the SARSA method

Part-3

In this Part, I have implemented the Q – learning algorithm to learn the trend in stock price environment which is given and perform a series of trades over a period of time. to end up with a profit. I have updated my Q-learning code to work with the Stock Trading Environment. In this environment, I have 3 possible action buy/sell/hold.

I have generated the Q table after training the dataset.

After implementing the Q-learning algorithm to the stock trading problem, I have demonstrated the “epsilon decay” and “total reward per episode”. The epsilon decay plot shows how the agent started by exploring different actions (buy, sell, or hold) randomly, with epsilon beginning at 1.0. I have plotted the decay curve for 100 episodes. After training, epsilon decreased to 0.01, meaning the agent started to rely more on the trading knowledge it gained. In the total reward per episode plot, I have shown the agent how well performed during training. In the beginning, the rewards were low and unpredictable as

the agent was still exploring. However, training the model for large number of episodes, the rewards generally increased, which represents that the agent learned to make better and more profitable trades. The increase in rewards indicates that the Q-learning algorithm helped the agent find a successful trading strategy, while the epsilon decay confirms the shift from exploration to relying on its learned knowledge.

I have attached here a Part of a training session of the Stock Trading Environment setup for the give data file:

```
Training progress:  11%|▉             | 112/1000 [00:00<00:04, 217.99Episode/s]

Episode 1/1 - Total Reward: -245.90507584969197, Final Portfolio Value: 107600.55541799996
Episode 1/1 - Total Reward: -225.11915941374704, Final Portfolio Value: 102460.01467200002
Episode 1/1 - Total Reward: -171.63193897051306, Final Portfolio Value: 118768.64180499992
Episode 1/1 - Total Reward: -237.1291085773898, Final Portfolio Value: 101325.414675
Episode 1/1 - Total Reward: -133.2396033191006, Final Portfolio Value: 125948.31631000002
Episode 1/1 - Total Reward: -79.65475154998764, Final Portfolio Value: 128203.44460500003
Episode 1/1 - Total Reward: -54.5267751357323, Final Portfolio Value: 124273.97461599996
Episode 1/1 - Total Reward: -207.14571062865195, Final Portfolio Value: 147606.57871099995
```

I have present here the Q-Table for the Stock Trading Environment with the Q-learning algorithm.

Q- Table

```
Saving Q-table for the stock trading environment.


Q-table:
            BUY   SELL   HOLD
State_0 526.08 507.47 507.68
State_1 510.84 509.43 544.50
State_2 507.62 498.41 498.40
State_3 492.90 506.85 494.45
Saved Q-table into file:  /Users/anip/Desktop/anipkuma_assignment1_checkpoint/Trained_Q_Val
ue_stock_trading_q_table_Nvidia.h5
Q-table shape: (4, 3)
```

Here, I have demonstrated the Rewards per episode graph for the given Stock Trading Environment with the Q-learning Algorithm in figure 20.

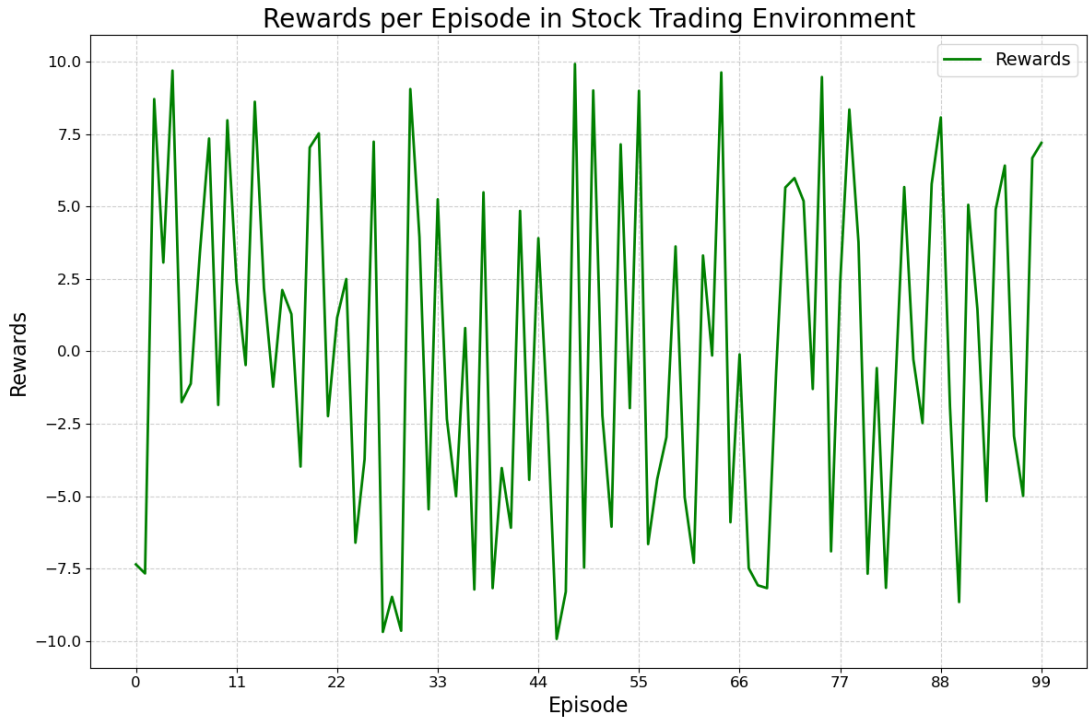


Figure 20: Rewards per episode in Stock Trading Environment for the Q-Learning Algorithm.

I have presented the Epsilon decay curve per episode for Stock Trading Environment for the Q-Learning Algorithm in figure 21. I have shown for 100 episode that is why the curve looks like straight decay line.

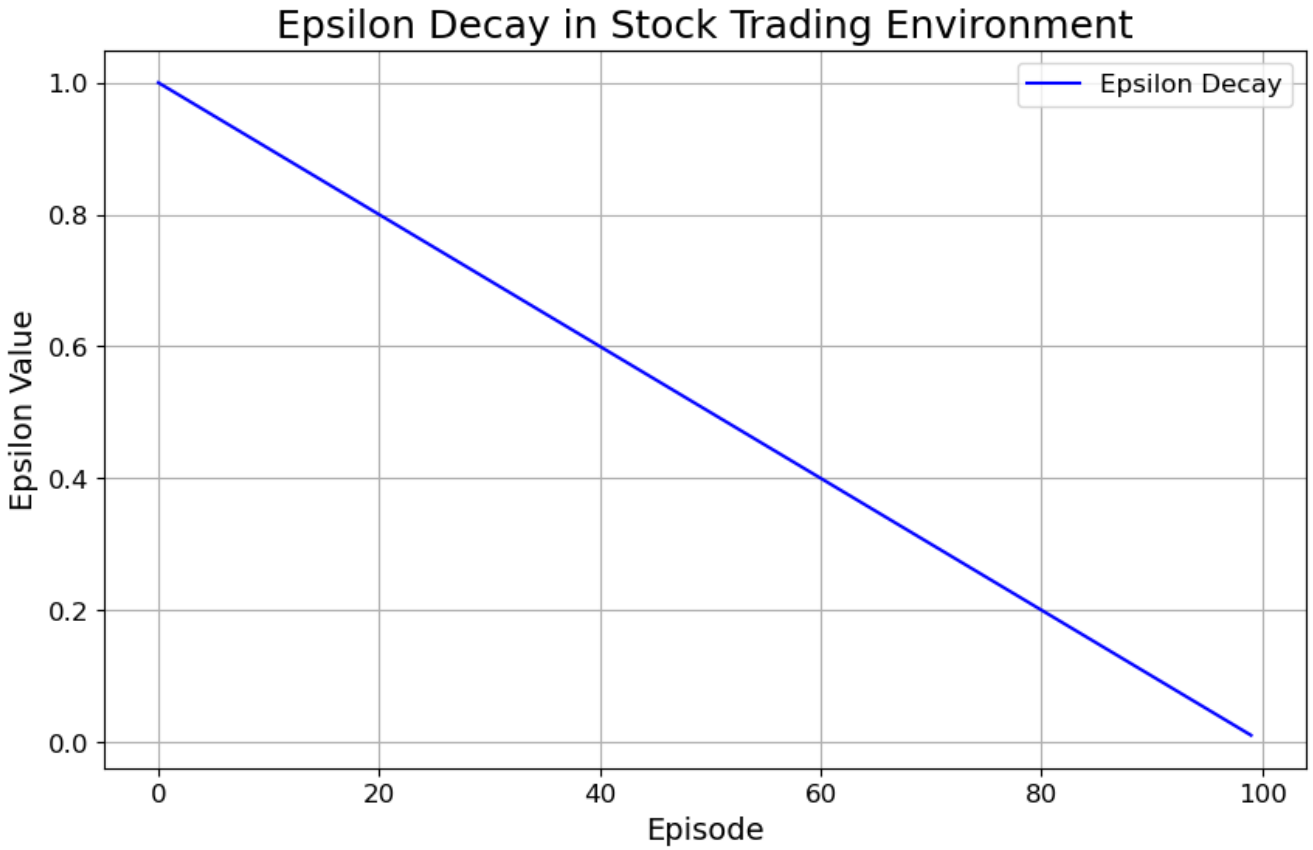
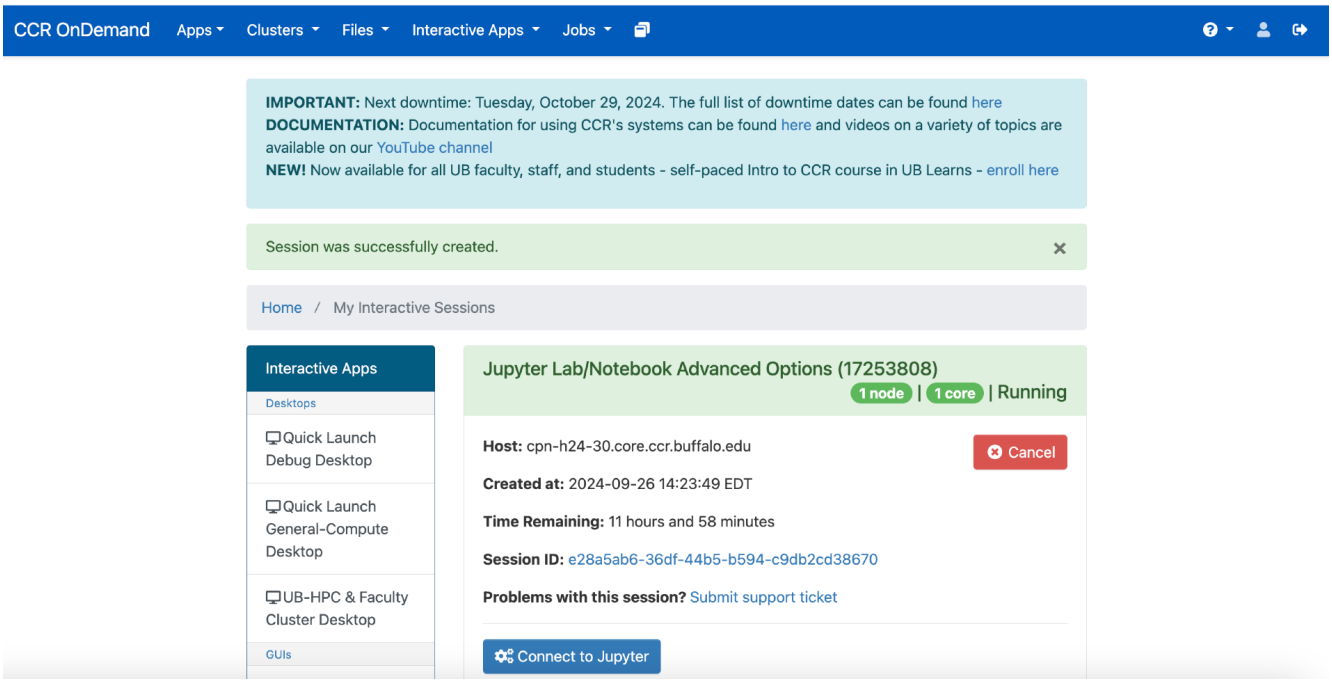


Figure 21: Epsilon decay per episode in Stock Trading Environment for the Q-Learning Algorithm.

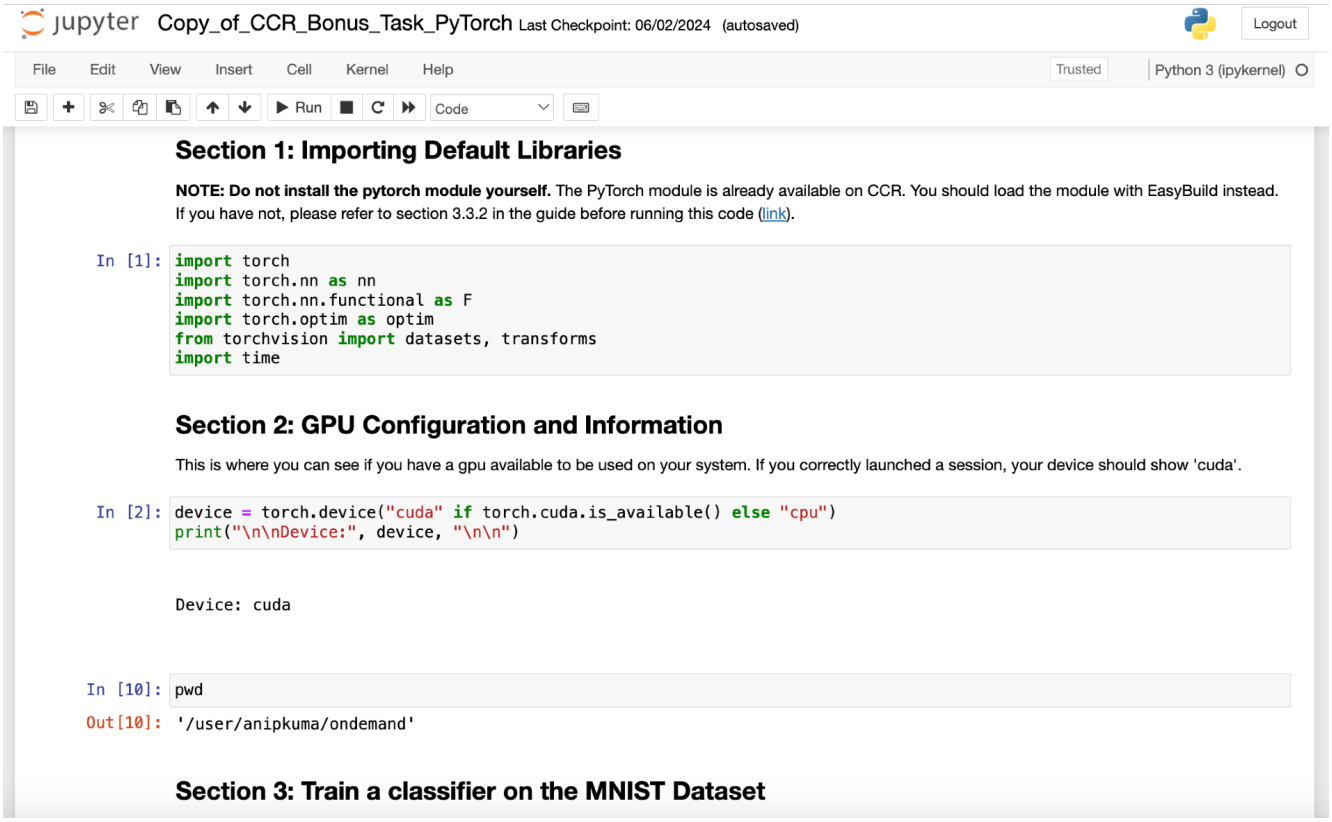
Bonus

CCR

I have successfully login and set up the CCR (On demand). I have created a Jupyter Session using the slurm scheduler and successfully launch the session. I have attached here the screenshot of the set up page and job details.



I have a previous file for checking the CCR setup. I have run this file which shows that the device is "cuda" which represents that I have successfully connect to the GPU support. Here is my screenshot for the set up.



Here, I have run a model with MNIST dataset and attached here screenshot.

```
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw/t10k-images
-idx3-ubyte.gz

  0%|          | 0/1648877 [00:00<?, ?it/s]

Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw/t10k-labels
-idx1-ubyte.gz

  0%|          | 0/4542 [00:00<?, ?it/s]

Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw

Test set: Average loss: 0.0033, Accuracy: 9683/10000 (97%)
```

References:

a) Demo lecture by Dr. Alina (https://ubuffalo-my.sharepoint.com/personal/avereshc_buffalo_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Favereshc%5Fbuffalo%5Fedu%2FDocuments%2F2024%5FFall%5FRL%2F%5Fpublic%2FCourse%20Materials%2FRL%20Environment%20Demo&ga=1)

b)Demo lecture by TA (https://ubuffalo-my.sharepoint.com/personal/avereshc_buffalo_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Favereshc%5Fbuffalo%5Fedu%2FDocuments%2F2024%5FFall%5FRL%2F%5Fpublic%2FCourse%20Materials%2FRL%20Environment%20Visualization&ga=1)

c) Demo lecture by Dr. Alina (Python demo tutorial for Multi-armed Bandit (MAB), https://piazza.com/class/m071wk03ok62ep/post/44)

d) https://taraqur.medium.com/reinforcement-learning-2-agent-in-the-warehouse-ea07a605d3ff

e) https://medium.com/@alwinraju/in-depth-guide-to-implementing-q-learning-in-python-with-openai-gyms-taxi-environment-cd356cc6a288¶

f) Robotic Warehouse: https://github.com/semitable/robotic-warehouse

g) Reinforcement Learning: Part 6: Temporal differencing (SARSA, Q learning, Expected SARSA) https://medium.com/@j13mehul/reinforcement-learning-part-6-temporal-differencing-sarsa-q-learning-expected-sarsa-b7725c755410

h) Class Lectures