Part 1: Define an RL Environment

1. **Describe the environment that you defined. Provide a set of states, actions, rewards, main objective, etc**

The environment defined in the provided code is a SearchRescueEnv class, which inherits from the gym.Env class of the OpenAI Gym library. This class simulates a search and rescue operation in a grid-based environment.

Grid Size: The state-space environment is based on a 4x4 grid.

States: The state is shown as a zero-filled 4x4 matrix, with a 1 designating the agent's position and a 2 designating the objective position.

Actions: Up (0), down (1), right (2), and left (3) are the four distinct actions that are accessible. The agent moves in the matching direction on the grid as a result of these activities.

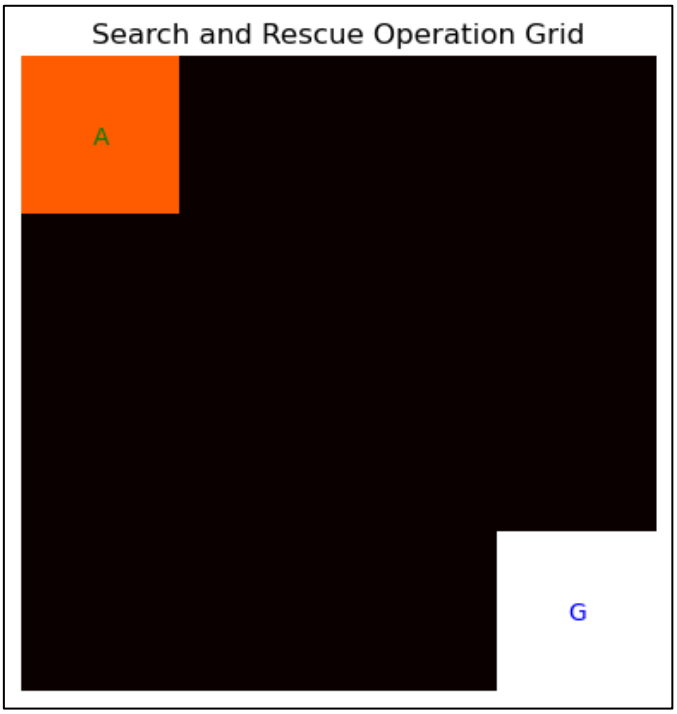
Rewards: The reward system is designed as follows:

- **A step penalty of -1** is given for each move to discourage aimless wandering.
- **Moving closer to the goal** yields a reward of **+1**.
- **Visiting a new cell** provides a reward of **+1**.
- Successfully **reaching the goal** position grants a large reward of **+10**.

Main Objective: The agent's main objective is to navigate the grid and reach the goal position (3,3), starting from the basecamp position (0,0).

Episode Termination: An episode ends if the agent reaches the goal or if 10 timesteps have passed.

2. Visualization of the Environment



3. Safety in AI

In the SearchRescueEnv, several measures are in place to ensure the safety and integrity of the environment:

Constrained Actions: The step method updates the agent's position based on the action taken, ensuring that the agent does not move outside the grid. This is done using min and max functions to restrict the agent's movement within the grid boundaries.

Defined State-Space: The state-space is clearly defined as a 4x4 grid. The state representation is updated in each timestep to reflect the current position of the agent and the goal.

Valid Action Selection: The environment's action space is a discrete space with four actions (up, down, left, right). This ensures that the agent chooses only from these allowed actions.

Tracking Visited States: The environment maintains a 'visited' matrix to track which cells have been visited, contributing to a more efficient exploration.

Termination Criteria: The environment specifies a termination condition, either when the agent reaches the goal or after 10 timesteps, preventing endless loops or aimless wandering.

These safety measures ensure that the agent operates within the defined rules and constraints of the environment, promoting a structured and safe exploration in the grid.

References:

Intro to reinforcement learning: temporal difference learning, SARSA vs. Q-learning | by Viet Hoang Tran Duong | Towards Data Science

Reinforcement Learning - Algorithms (unsw.edu.au)

1) **SARSA (State-Action-Reward-State-Action)** is a reinforcement learning algorithm that operates on the policy being used to make decisions. The agent modifies its Q-values by taking into account the current state, action, reward, and the subsequent state, action pair. The system employs an exploration-exploitation technique to effectively manage the trade-off between exploring new actions and using the information acquired so far. It is used in reinforcement learning to obtain a policy for a Markov Decision Process (MDP). As an on-policy Temporal Difference (TD) control mechanism, it iteratively improves the decision-making policy by integrating fresh knowledge obtained from recent actions and observations.

   The following rule is used to update the Q-values, which show how much the expected benefits are for each state-action pair, every time the system moves from state s to state s':

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right]$$

S is the present state

A is the present action

R is the reward that you get for doing action an in state s

S' is the new state after action an is taken

A' is the next action that is done in the new state s'.

α is the learning rate, which tells us how much new information takes precedence over old information.

γ is the discount factor and it tells us how important future rewards will be.

Key features:

SARSA is an on-policy algorithm, which means that it looks at the policy that is used to make choices and finds ways to make it better.

The action-value function Q is used to evaluate the policy.

It changes its Q-values based on the rewards that are predicted under the present policy.

Advantages

When compared to other algorithms like Q-learning, SARSA is easier to understand and use.As an on-policy method, it can learn more conservative policies because it changes based on how the policy actually works, taking into account the exploration-exploitation trade-off.

Disadvantages

SARSA's tendency to incorporate exploration into its updates can lead it to converge on suboptimal policies. This means that if the exploration strategy is weak, the resulting policy may perform poorly. Compared to off-policy methods like Q-learning, SARSA's reliance on its own actions for updates can lead to slower learning, as it doesn't consider potentially better options. This can be a disadvantage in situations where taking risks is crucial for achieving optimal performance.

Double Q-learning

A reinforcement learning method that works outside of policies is called double Q-learning. It is similar to the SARSA algorithm in a few ways. This method refines its learning by using two different Q-tables. The procedure involves doing an action in the current state, recording the resulting reward and the future state, and then using the highest expected Q-value for the next state as listed in Table-B to modify the estimated Q-value for the previous state-activity combination in Table-A. It switches between these two tables throughout its learning period.

$$Q_A(s,a) \leftarrow Q_A(s,a) + \alpha\left[r + \gamma Q_B(s', \arg\max_{a'} Q_A(s',a')) - Q_A(s,a)\right]$$

$$Q_B(s,a) \leftarrow Q_B(s,a) + \alpha\left[r + \gamma Q_A(s', \arg\max_{a'} Q_B(s',a')) - Q_B(s,a)\right]$$

S is the present state

A is the action

R is the reward that you get for doing action an in state s

S’ is the new state after action an is taken

α is the learning rate

γ is the discount factor

The action-value functions Q A and Q B are updated according to somewhat different criteria, the overestimation bias is lessened.

Double Q-learning uses two different value predictors to keep the steps of picking actions and judging those actions separate. This stops overestimated value predictions from happening.

Each time through the method, the strategy based on the replacement value function is used to change two value functions, Q A and Q B.

Advantages:

Traditional Q-learning tends to overvalue predictions, but this method makes that a lot less likely to happen.

When two estimators are used together, they produce more accurate value predictions, which makes policy learning work better.

In uncertain situations, where standard Q-learning's tendency to overvalue might get in the way of success, double Q-learning usually does better.

Disadvantages:

Two Q-tables require more memory to be kept up to date.

Because the changes are spread out over two different tables, the rate of learning may slow down.

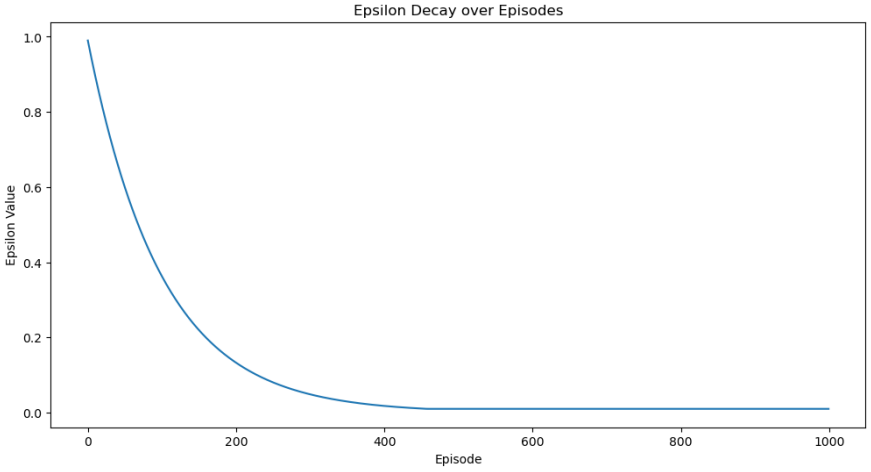
Because changes must be managed and synchronized across two tables, execution is slightly more difficult than with regular Q-learning.

2) **Applying SARSA to solve the environment defined in Part 1. Include Qtable. Plots should include epsilon decay and total reward per episode.**

Applying SARSA

Qtable:

```
Final Q-Table:
State: (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 4.94 | Action 1: 2.79 | Action 2: 10.31 | Action 3: 3.69
State: (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 4.77 | Action 1: 10.39 | Action 2: 2.19 | Action 3: 4.10
State: (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.37 | Action 1: 0.65 | Action 2: 7.36 | Action 3: -0.71
State: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.82 | Action 1: 0.65 | Action 2: 2.91 | Action 3: -1.16
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2) Action 0: -0.91 | Action 1: -0.61 | Action 2: 2.10 | Action 3: -0.75
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2) Action 0: -0.57 | Action 1: -0.67 | Action 2: 6.40 | Action 3: -0.15
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2) Action 0: 0.63 | Action 1: 2.36 | Action 2: 9.47 | Action 3: 0.58
State: (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 2.59 | Action 1: 5.88 | Action 2: 10.16 | Action 3: 3.41
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2) Action 0: 4.65 | Action 1: 8.60 | Action 2: 10.90 | Action 3: 5.01
State: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 1.62 | Action 1: 10.79 | Action 2: 5.85 | Action 3: 5.79
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2) Action 0: 0.77 | Action 1: 0.99 | Action 2: 10.88 | Action 3: -0.17
State: (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.83 | Action 1: 5.65 | Action 2: 0.51 | Action 3: -0.56
State: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.10 | Action 1: 9.40 | Action 2: -0.10 | Action 3: -0.07
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2) Action 0: 5.60 | Action 1: 11.00 | Action 2: 6.59 | Action 3: 6.30
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.00 | Action 1: 0.00 | Action 2: 0.00 | Action 3: 0.00
State: (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -1.09 | Action 1: 2.17 | Action 2: -0.67 | Action 3: -0.63
```



The provided visuals offer insights into the learning process of a SARSA-based learning agent within a simulated setting.

Trend of Epsilon Decay Across Episodes:

The first graph offers a clear depiction of the gradual decrease in the exploration parameter, epsilon, across a sequence of 1000 training episodes.

Initially, epsilon is set to a maximum value of 1.0, which indicates that the agent is programmed to explore the environment fully, without any bias towards previously learned actions or strategies.

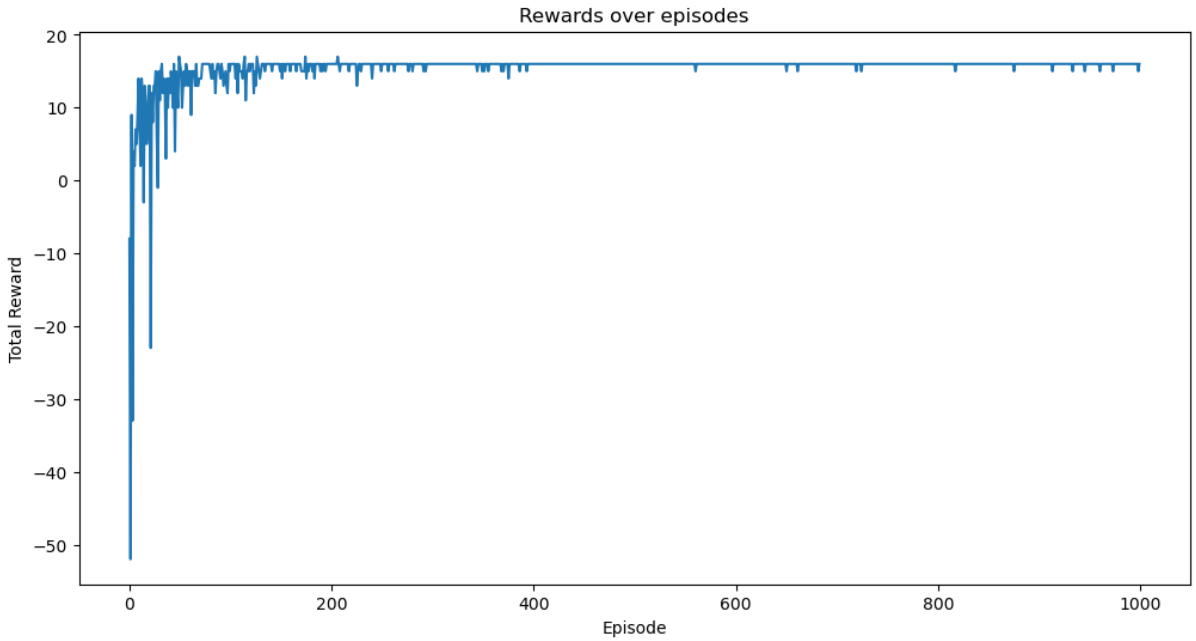
As training progresses, the epsilon value is systematically reduced by a factor of 0.99 with the conclusion of each episode. This steady decline is maintained until epsilon approaches a minimum threshold of 0.01, beyond which it does not reduce any further. This plot visually encapsulates the agent's shift from a phase of initial exploration to a more focused exploitation of its accumulated knowledge.

The curve's flattening at the lower bound of epsilon signifies a point where the agent has potentially established a policy that balances the benefits of exploration with the gains from exploiting its learned knowledge. This plateau indicates that further exploration is deemed less necessary as the agent has presumably learned enough about the environment to navigate it effectively.

The graph does not show any erratic changes or spikes in epsilon, implying that the agent's learning process is not disrupted by significant anomalies or sudden changes in the environment that might otherwise necessitate a return to higher exploration rates.

Despite the decay, the agent still retains a small degree of exploration (as epsilon never reaches zero). This ensures that the agent retains some capacity for discovery, preventing the policy from becoming overly rigid and allowing the agent to adapt to any potential changes or nuances in the environment that were not encountered during the initial episodes.

The smooth descent of the epsilon curve underscores a consistent and well-regulated transition in the agent's behavior from random exploration to informed decision-making based on its experience.



1. **Balanced Exploration and Exploitation:** The SARSA agent adeptly balances exploration and exploitation, crucial for efficient reinforcement learning.
2. **Stable Rewards:** The rewards achieved by the agent have stabilized over time, indicating genuine mastery of an effective strategy.
3. **Parameter Selection:** The choice of learning rate (alpha), discount factor (gamma), epsilon decay schedule, and minimum epsilon value demonstrates a deep understanding of these parameters' impact on the agent's learning curve.

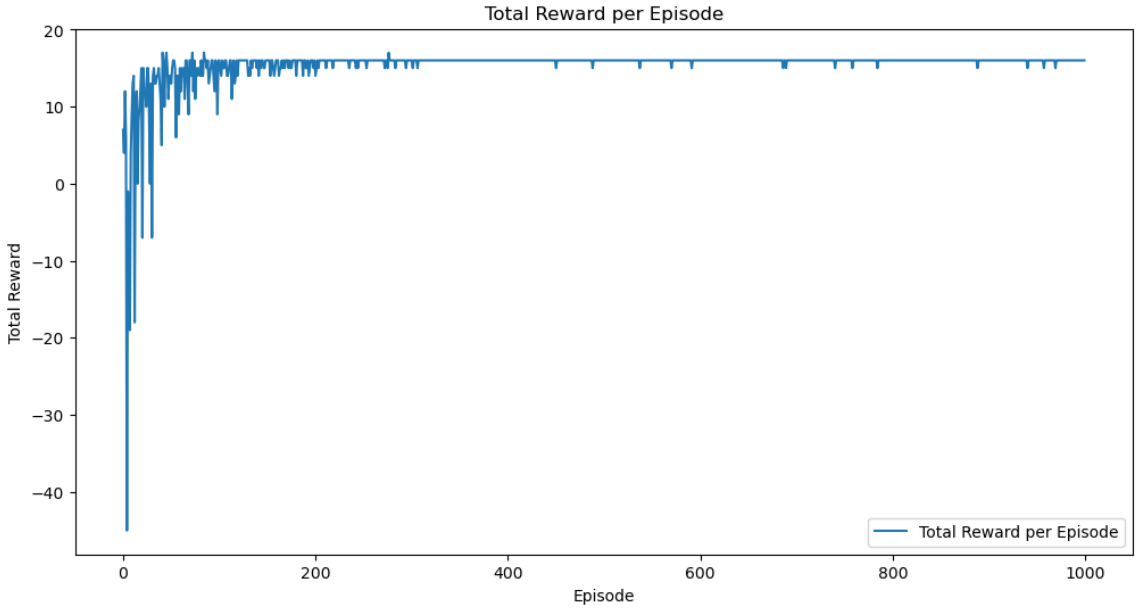
4. **Adaptability:** The agent's proficiency in the current environment should be complemented by monitoring its performance under diverse conditions to ensure adaptability and generalization capabilities.
5. **Ethical Considerations:** Deployment of the agent in real-world scenarios necessitates ongoing ethical considerations to align its learned behaviors with ethical guidelines, potentially requiring intervention or fine-tuning mechanisms.

In summary, the SARSA agent's commendable learning progress includes a balanced exploration-exploitation approach, stable rewards, well-tuned parameters, adaptability testing, and ethical awareness as key factors contributing to its success. Ongoing monitoring and ethical considerations remain essential in its development and deployment.

Applying Double Q-learning to solve the environment defined in Part 1. Include Q-tables. Plots should include epsilon decay and total reward per episode. Include the details of the setup that returns the best results

Q-tables:

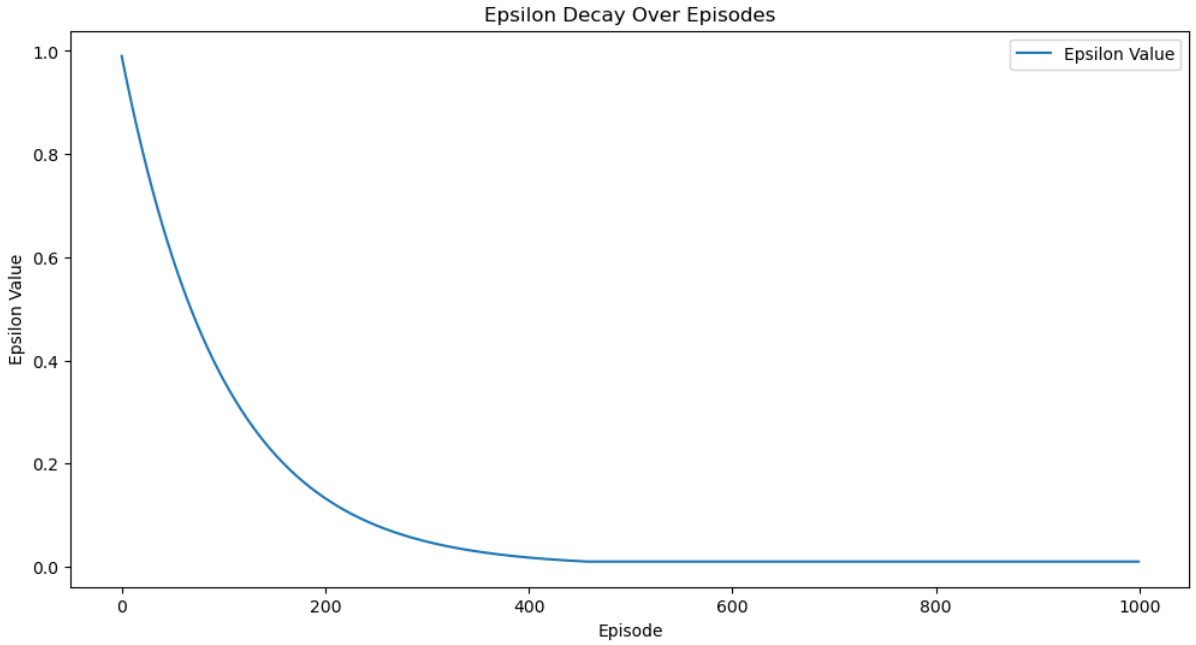
```
Trained Q1-Table:
State: (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 4.25 | Action 1: 10.57 | Action 2: 1.93 | Action 3: 4.42
State: (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.28 | Action 1: 0.99 | Action 2: 2.71 | Action 3: 0.70
State: (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.43 | Action 1: 0.29 | Action 2: 2.98 | Action 3: -0.46
State: (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.11 | Action 1: 6.48 | Action 2: 1.62 | Action 3: 0.20
State: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.10 | Action 1: 7.84 | Action 2: 2.15 | Action 3: 0.84
State: (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.18 | Action 1: 7.01 | Action 2: -0.23 | Action 3: -0.03
State: (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.08 | Action 1: 8.15 | Action 2: 0.35 | Action 3: 0.46
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2) Action 0: 3.29 | Action 1: 3.74 | Action 2: 10.81 | Action 3: 2.92
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2) Action 0: 0.80 | Action 1: 1.03 | Action 2: 8.22 | Action 3: 0.09
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2) Action 0: 1.47 | Action 1: 2.60 | Action 2: 10.42 | Action 3: 0.97
State: (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 4.55 | Action 1: 10.66 | Action 2: 2.75 | Action 3: 3.14
State: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2) Action 0: 3.21 | Action 1: 1.58 | Action 2: 10.71 | Action 3: 4.14
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2) Action 0: 0.95 | Action 1: -0.05 | Action 2: 2.97 | Action 3: -0.33
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2) Action 0: 2.47 | Action 1: 6.22 | Action 2: 10.90 | Action 3: 4.67
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2) Action 0: 2.91 | Action 1: 11.00 | Action 2: 5.99 | Action 3: 4.66
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.00 | Action 1: 0.00 | Action 2: 0.00 | Action 3: 0.00

Trained Q2-Table:
State: (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 4.50 | Action 1: 10.59 | Action 2: 1.69 | Action 3: 3.01
State: (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.40 | Action 1: 1.14 | Action 2: 1.80 | Action 3: 0.08
State: (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.32 | Action 1: 2.05 | Action 2: 4.19 | Action 3: -0.16
State: (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.05 | Action 1: 7.19 | Action 2: 1.92 | Action 3: 0.52
State: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.66 | Action 1: 9.76 | Action 2: 1.68 | Action 3: 0.83
State: (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: -0.01 | Action 1: 4.68 | Action 2: -0.48 | Action 3: -0.19
State: (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.05 | Action 1: 6.80 | Action 2: 0.78 | Action 3: 0.54
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2) Action 0: 2.81 | Action 1: 4.81 | Action 2: 10.80 | Action 3: 3.49
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2) Action 0: 1.57 | Action 1: -0.40 | Action 2: 8.08 | Action 3: 0.05
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2) Action 0: 1.11 | Action 1: 0.09 | Action 2: 10.53 | Action 3: 1.31
State: (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 4.04 | Action 1: 10.63 | Action 2: 3.33 | Action 3: 2.37
State: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2) Action 0: 1.78 | Action 1: 1.68 | Action 2: 10.73 | Action 3: 3.64
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2) Action 0: -0.10 | Action 1: -0.31 | Action 2: 4.48 | Action 3: -0.12
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2) Action 0: 2.33 | Action 1: 6.47 | Action 2: 10.90 | Action 3: 4.16
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2) Action 0: 4.07 | Action 1: 11.00 | Action 2: 5.75 | Action 3: 5.38
State: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2) Action 0: 0.00 | Action 1: 0.00 | Action 2: 0.00 | Action 3: 0.00
```



Training Performance (Total Reward per Episode Graph):

1. **Initial Underperformance:** The graph exhibits an initial phase of low performance, where the agent accumulates negative total rewards.
2. **Rapid Improvement:** Within the first few dozen episodes, the agent experiences a remarkable turnaround, consistently achieving positive rewards for the remainder of the training.
3. **Stabilization of Rewards:** The significant point of interest is the stabilization of rewards at a positive value, indicating the agent's proficiency in not only learning but also effectively applying the learned policy.
4. **Absence of Fluctuations:** In the later episodes, there is a conspicuous absence of significant fluctuations in reward values, suggesting that the agent's policy has converged, and it consistently applies its learned knowledge.
5. **Adaptation and Learning:** The agent's quick adaptation and learning capabilities are evident in its rapid turnaround from initial struggles.
6. **Decision-Making Stability:** Policy convergence highlights the agent's decision-making stability, a valuable trait for real-world applications where consistency is crucial.
7. **Generalization Testing:** The importance of testing the agent's acquired policy in diverse scenarios to assess its ability to generalize beyond the training environment is emphasized.
8. **Iterative Learning:** The graph underscores the iterative nature of reinforcement learning, where initial challenges lead to improvement,

highlighting the agent's resilience and adaptability throughout its learning journey.



Epsilon Decay (Epsilon Decay Over Episodes Graph):

1. **Gradual Exploration Shift:** Epsilon decay involves a gradual transition from exploration to exploitation as the agent becomes more proficient. It starts with comprehensive exploration, enabling the agent to understand the environment in its early stages.
2. **Controlled Exploration Reduction:** Over time, epsilon is systematically reduced after each episode, following an exponential pattern. This controlled reduction balances exploration and exploitation, encouraging the agent to favor actions it deems optimal.
3. **Steady Exploitation Phase:** When epsilon reaches a minimum threshold of 0.01, it remains constant. This signifies a shift towards a more exploitation-centric strategy, with the agent relying on its acquired knowledge for consistent decision-making.
4. **Smooth Decay Profile:** The epsilon decay graph displays a smooth and continuous decay curve, devoid of sudden irregularities. This controlled pattern ensures a gradual shift from exploration to exploitation during training, promoting learning stability.
5. **Balanced Exploration-Exploitation:** Epsilon decay strikes a balance between exploration and exploitation, addressing one of the central challenges in

reinforcement learning. It encourages extensive initial exploration and a subsequent focus on exploiting learned knowledge.

6. **Learning Consistency:** The consistent decay mechanism enhances learning stability by avoiding abrupt changes in exploration behavior, which could disrupt the agent's convergence towards an optimal policy.
7. **Adaptive Strategy:** While exploration decreases, it's essential to assess the agent's adaptability to unforeseen challenges or environmental changes, ensuring it can adjust its strategy as needed.

Insights into Double Q-Learning for Rescue Missions

1. **Initialization and Progression of Strategy Tables**: Initially, the Q1 and Q2 tables contain no data, reflecting a starting point without biases. As training progresses, these tables fill with values reflecting the rewards of different actions, guiding the agent's choices.
2. **Fine-tuning of Learning Dynamics**: Achieving the best outcomes relies on carefully adjusting the learning rate, the reward discounting, and the rate at which the agent reduces its random exploration. This fine-tuning helps the agent learn efficiently, balancing the initial exploration with later exploitation of known strategies.
3. **Sustaining an Adaptive Approach**: Keeping the exploration rate from reaching zero is essential for maintaining the agent's adaptability. This ensures that the agent continues to explore and adjust to new aspects of the environment even as it becomes more experienced.
4. **Analysis of Agent Development**: The graphs reflect a successful learning trajectory, showing that the agent has settled into a stable strategy that consistently earns rewards. The smooth reduction in exploration rate indicates a well-managed transition from learning to applying known strategies.
5. **Future Validation of Strategies**: To validate the effectiveness of the agent's strategies, a thorough examination of the final Q-tables and stress-testing the agent in diverse scenarios would be necessary. This would ascertain the agent's strategic depth and its potential for further refinement.

Provide the evaluation results for both SARSA and Double Q-learning. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learned policy. Plot should include the total reward per episode.

SARSA:

```
Episode: 1, Total Reward: 16
Episode: 2, Total Reward: 16
Episode: 3, Total Reward: 16
Episode: 4, Total Reward: 16
Episode: 5, Total Reward: 16
Episode: 6, Total Reward: 16
Episode: 7, Total Reward: 16
Episode: 8, Total Reward: 16
Episode: 9, Total Reward: 16
Episode: 10, Total Reward: 16
```

The consistent findings, which provide the same outcome every time, imply that the agent has established a stable policy. By adhering to a greedy strategy, the agent continuously chooses, without considering other possibilities, what it has come to understand as the best courses of action based on its value estimations.

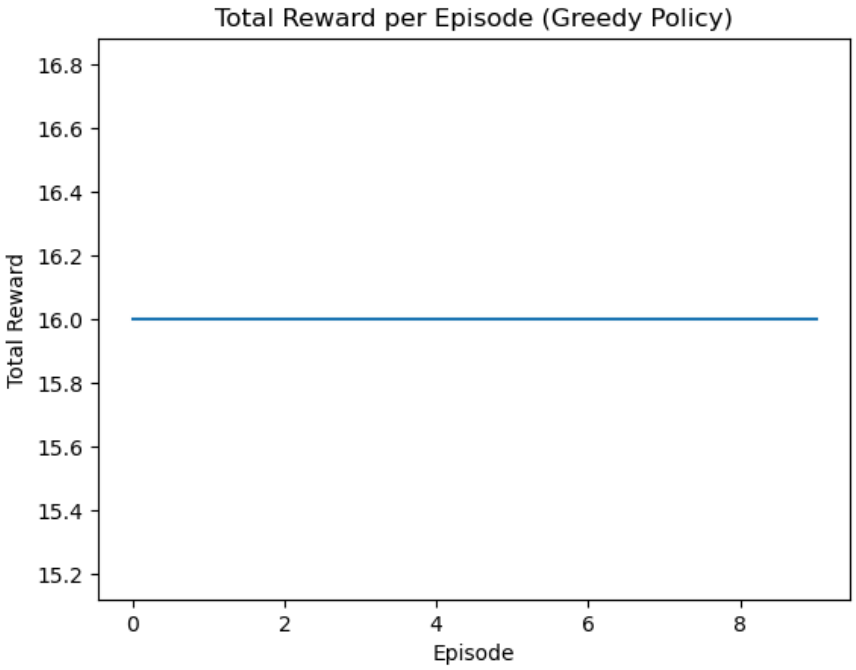
The consistency in the rewards might indicate several scenarios:

It's feasible that the agent has discovered an optimum or near optimal strategy that maximizes reward in a deterministic environment.

The environment may not vary in its rewards in response to the agent's behaviors, leading to a repeatable result.

The agent may have learned an efficient strategy that doesn't need further research if it has already done enough exploring in earlier episodes (which aren't included in the shared data).

In reinforcement learning, striking a balance between exploration and exploitation is essential, especially when using on-policy techniques like SARSA. Here, there is no exploration—which is acceptable for policy review but not for continuous learning—because the agent is just acting in a greedy manner.



The graph demonstrates that an agent with a greedy policy consistently performs well, with a flat line indicating the same total reward for each episode across all ten episodes. This consistency suggests the agent has learned a good rule for the task, especially if the task's world doesn't change. However, if the work setting changes or the learning phase isn't over, this uniformity may indicate insufficient exploration, which is crucial in reinforcement learning.

Double Q-learning:

```
Greedy Episode 1: Total Reward 16
Greedy Episode 2: Total Reward 16
Greedy Episode 3: Total Reward 16
Greedy Episode 4: Total Reward 16
Greedy Episode 5: Total Reward 16
Greedy Episode 6: Total Reward 16
Greedy Episode 7: Total Reward 16
Greedy Episode 8: Total Reward 16
Greedy Episode 9: Total Reward 16
Greedy Episode 10: Total Reward 16
```

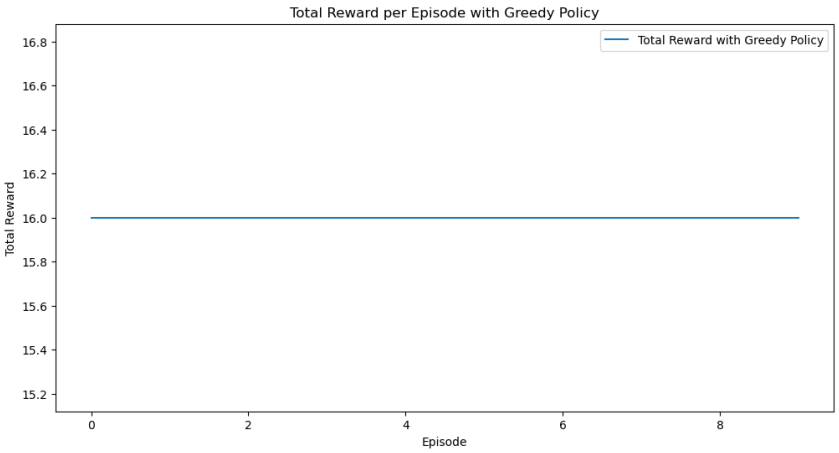
The assessment of the Double Q-learning algorithm demonstrates a continuously steady performance throughout ten episodes of a greedy policy execution. From the first to the tenth episode, the agent earned a total reward of sixteen. The consistency of the incentives given out for each episode indicates that the agent's policy has

probably converged and that it is consistently implementing the training phase's most lucrative activities.

This outcome—that is, the agent constantly receives the same reward in each episode—may suggest that the agent is making the most use of the learnt policy in a potentially deterministic setting where rewards and state transitions are constant. The absence of reward variance may also indicate that the agent has already adequately explored during the training phase before implementing the greedy policy, or that the environment does not need exploration after an acceptable policy is learnt.

This steady performance while using Double Q-learning may also demonstrate the algorithm's advantage in lessening the overestimation of Q-values, which may result in more dependable learning and judgment. But it's crucial to remember that, even though a greedy policy can assess how well a learned policy is working, it doesn't investigate other options that might yield larger rewards in the event that the environment changed or if the environment's complexity prevented the current policy from fully capturing the best strategies.

In conclusion, when evaluated using a greedy method, your Double Q-learning algorithm seems to have successfully learnt a stable policy that performs consistently over numerous episodes, suggesting a well-trained model in the specified environment.



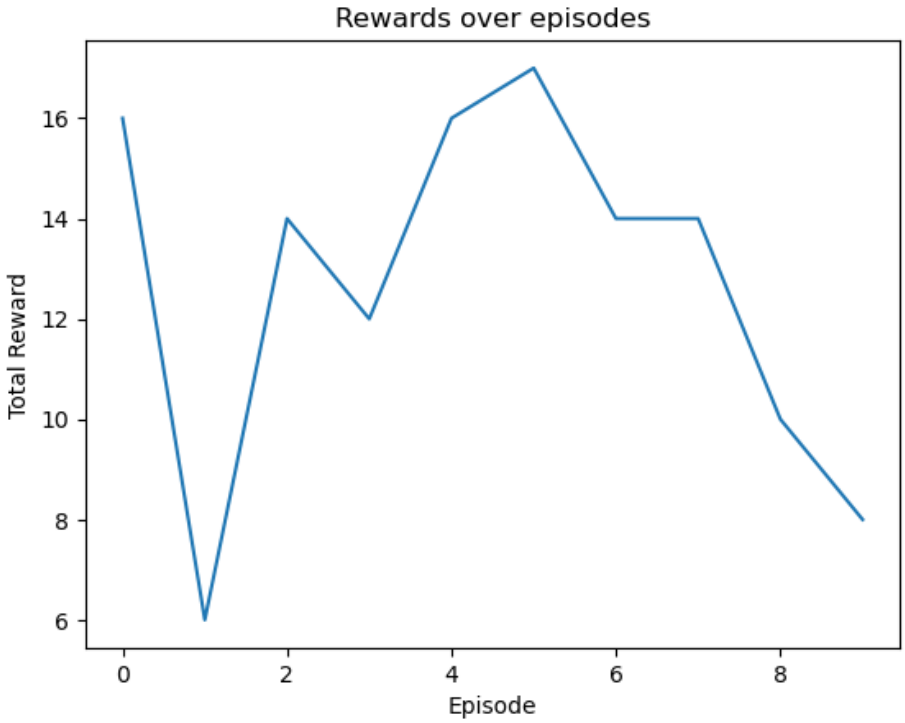
This plot demonstrates that the agent's greedy behavior leads to a stable and consistent result. If the line is flat, it means that the rewards don't change from episode to episode when the greedy policy is used. This means that either the policy learned is probably the best one for the given environment, or the environment is fixed and

can't change in a way that would affect the agent's performance.

3) Provide the analysis after tuning at least two hyperparameters from the list above. Provide the reward graphs and your explanation for each of the results. In total, you should have at least 6 graphs for each implemented algorithm and your explanations. Make your suggestion on the most efficient hyperparameters values for your problem setup.

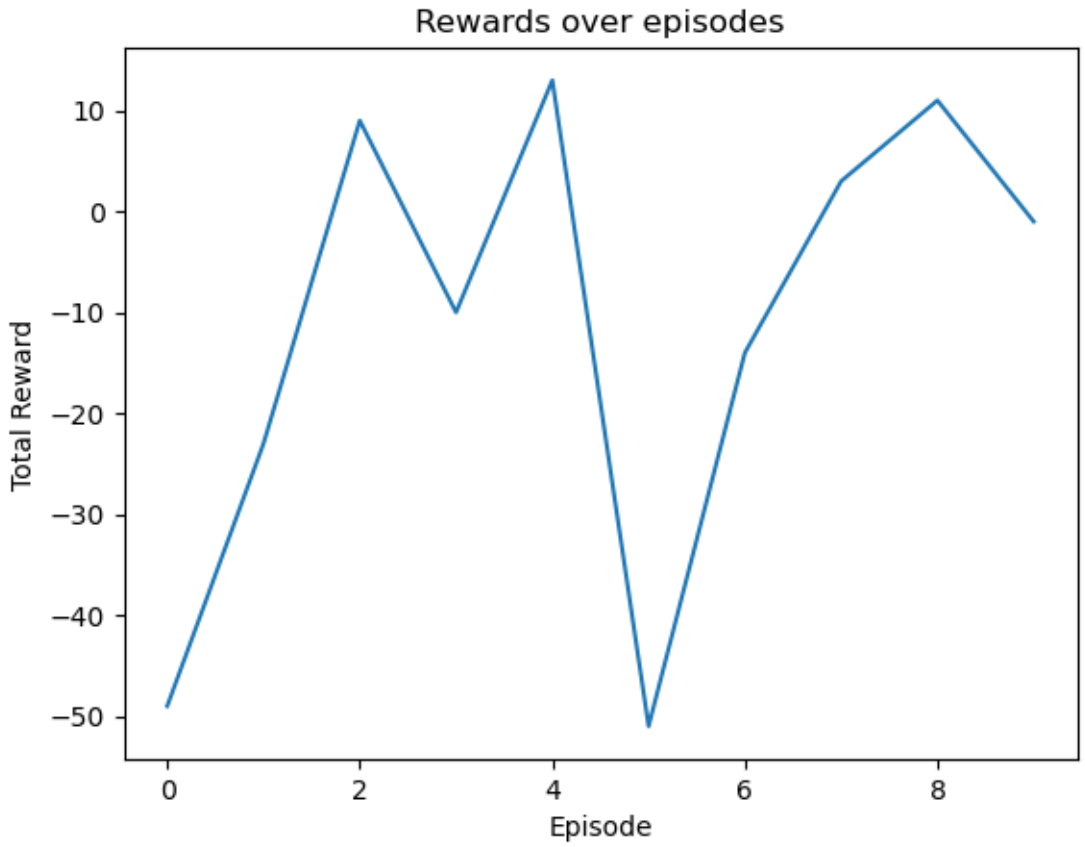
SARSA

Training with γ=0.8 and ε_decay=0.95



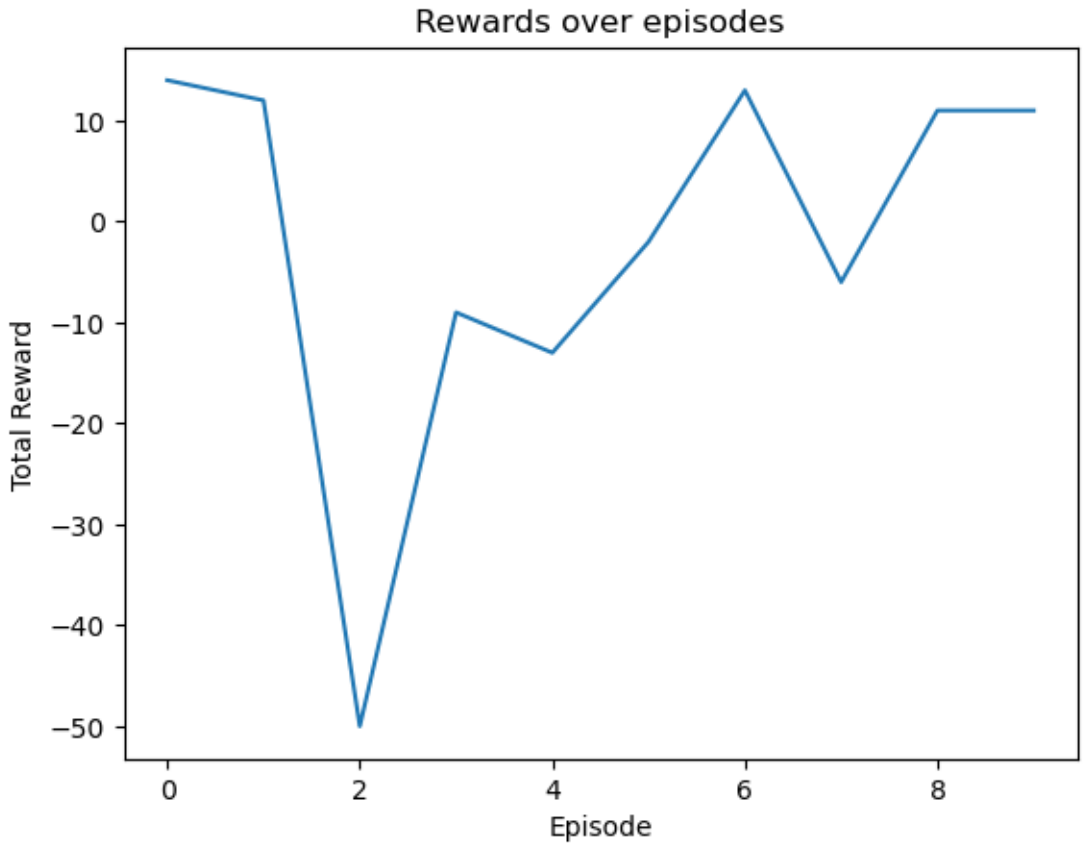
The above graphic shows a fluctuating reward pattern with peaks and troughs that re present the agent's fluctuating performance in reaching large rewards. This unpredict ability might indicate that the environment contains some degree of stochasticity, whi ch influences the rewards earned at the end of each episode, or it could mean the ag ent is still learning. This graph is representative of the training phase of reinforcement learning, when an agent investigates and picks up knowledge from its surroundings, which may result in decreased short-term rewards.

Training with γ=0.8 and ε_decay=0.99



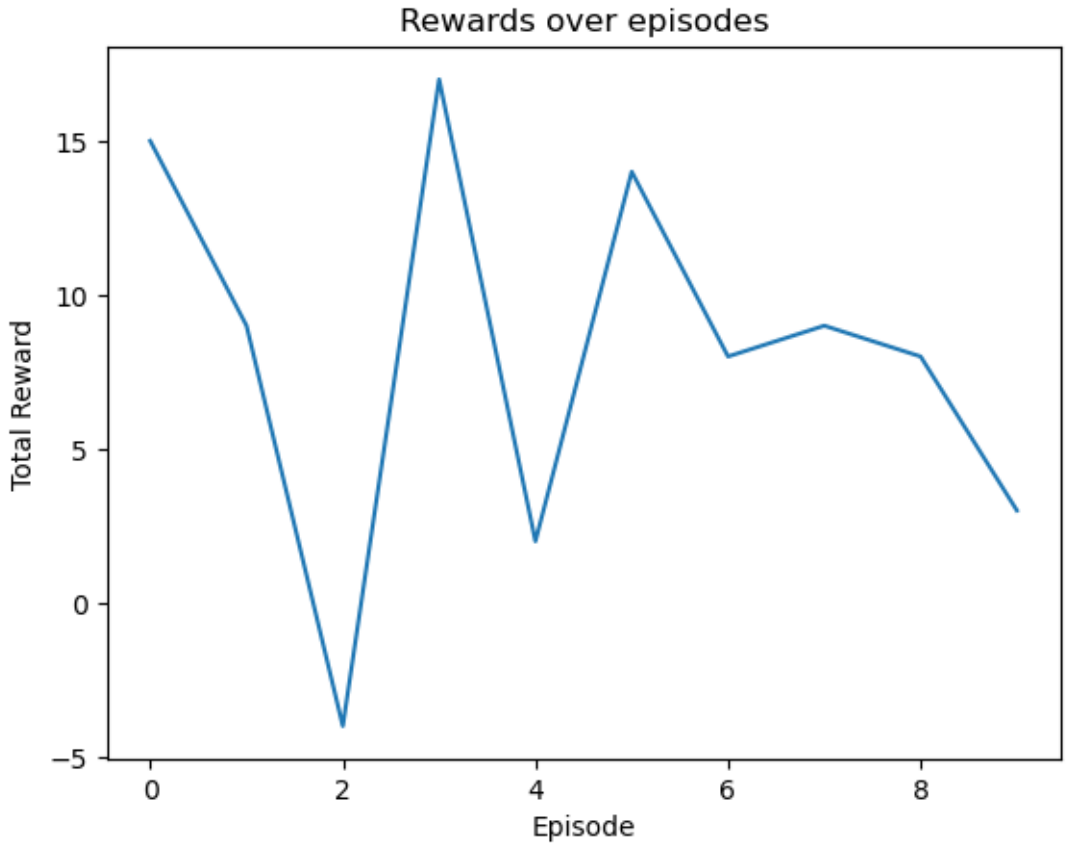
Significant variations in reward values, including negative incentives that suggest possible fines or losses, are shown in the graph. Its positive and negative peaks indicate that the agent's interactions with the environment will have different results. Negative incentives might point to difficult situations or learning procedures where errors bring forth unfavorable comments. This graph represents common scenarios in which agents experiment with various tactics or face various obstacles. It suggests that the agent's policy is changing and that the environment is not always producing favorable results. This graph aids in determining how the agent has learned and improved its approach over time.

Training with γ=0.8 and ε_decay=0.995



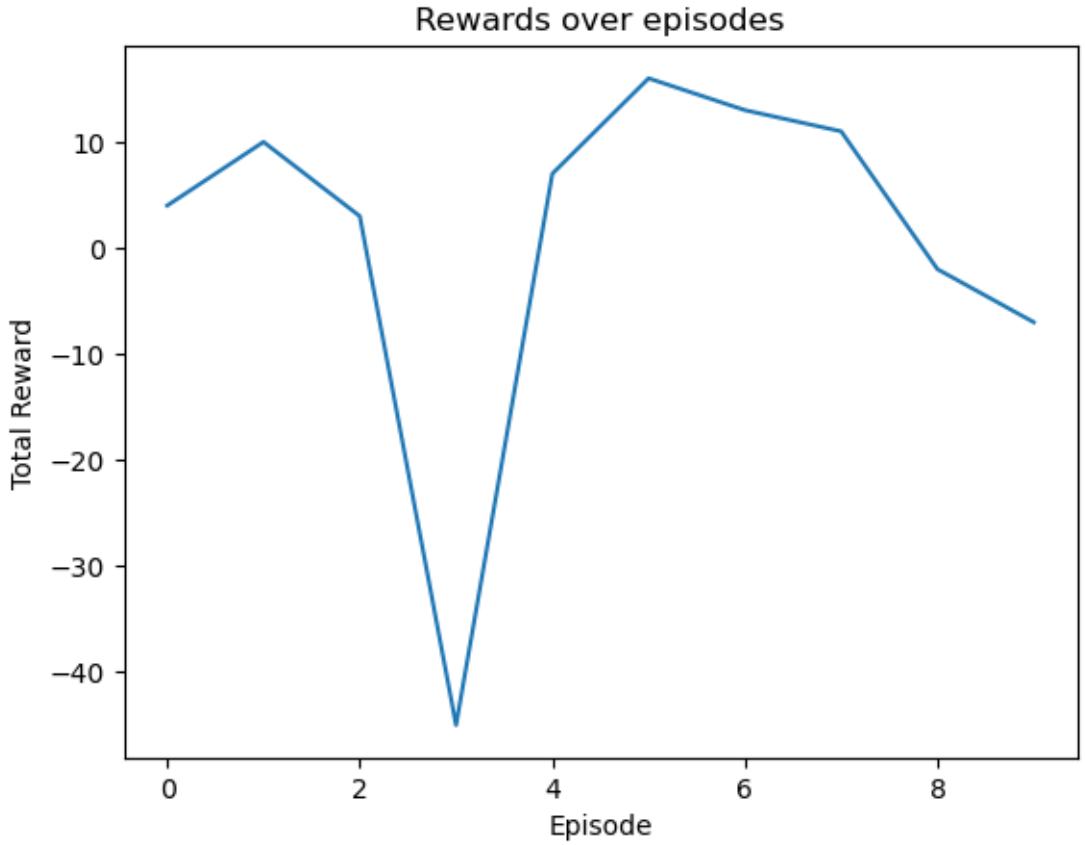
The agent receives both positive and negative incentives, with negative rewards denoting penalties or losses, according to the graph. This implies that the agent is going through a learning phase when it is facing different obstacles and experiencing both successes and failures. The large fluctuations in reward outcomes indicate that the agent's approach is not yet stable or optimal. These changes aid in the analysis of the agent's learning curve and degree of adaptation to the complicated environment. The need for a strong learning mechanism to enhance policy based on environmental input is highlighted by the existence of negative incentives.

Training with γ=0.9 and ε_decay=0.95...



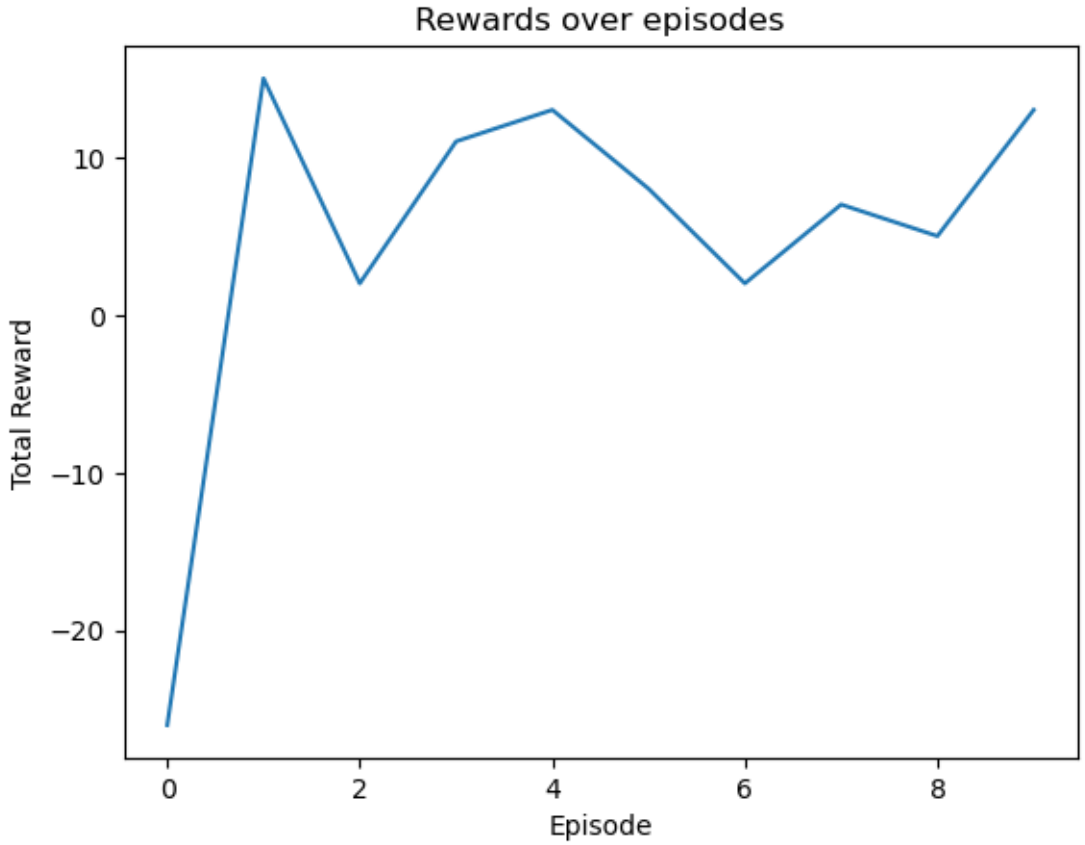
While there is considerable volatility in the trend, overall return is typically good and fluctuates in value. The plot shows an increasing tendency, with the biggest peak occurring in the middle, that begins with a negative reward in the first episode and swiftly recovers to a positive reward by the second. After the first episode, the payouts gradually decline but do not go below zero as the game progresses. This implies that, despite some performance unpredictability, the agent may be picking up new skills or adjusting to its surroundings.

Training with γ=0.9 and ε_decay=0.99...



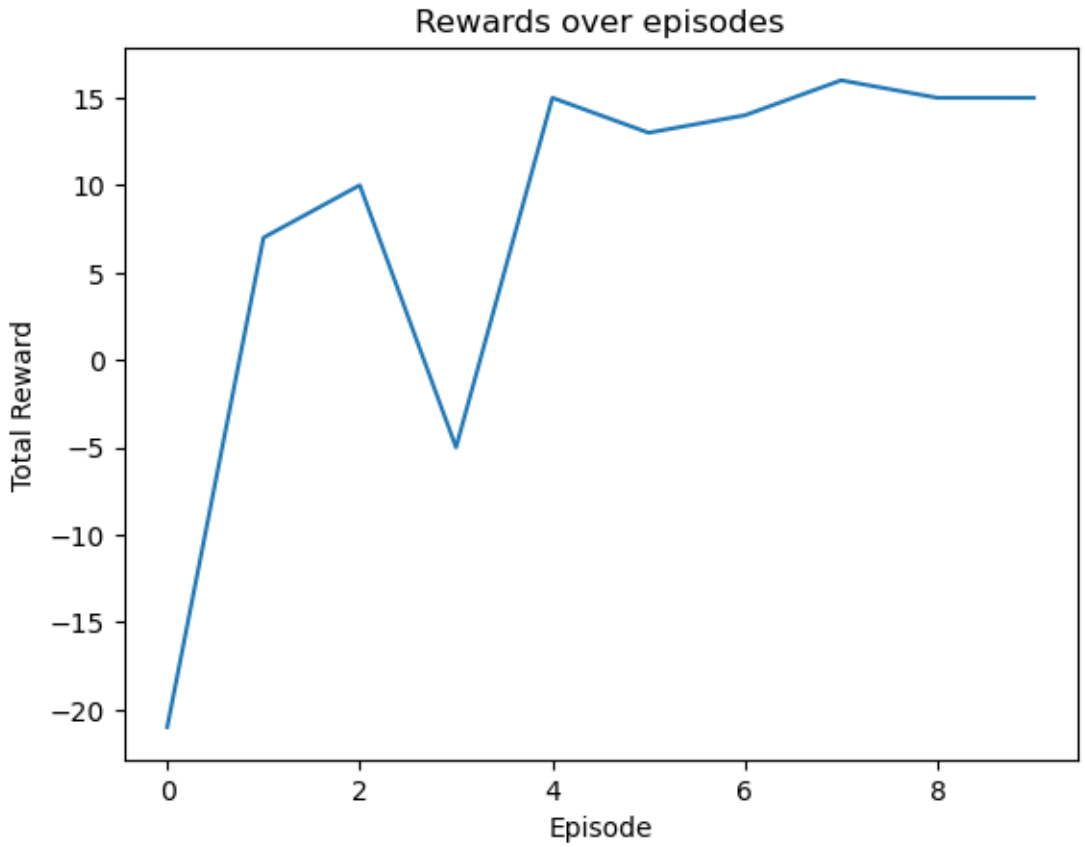
A comparable line plot with a distinct reward pattern is shown on the graph. The overall payout for each episode varies a lot in this case, even falling sharply into negative numbers. It begins with a somewhat favorable reward, falls into a significant negative value in the second episode, rises to a high in the middle episodes, and then begins to decline once again. The existence of both positive and negative incentives, with the negative rewards being somewhat large, implies that the agent could be running into various obstacles or dangerous situations in the surroundings that result in fines. This graph suggests a more complicated environment or a more unstable learning process where the agent's present approach is less successful in producing positive rewards.

Training with γ=0.9 and ε_decay=0.995



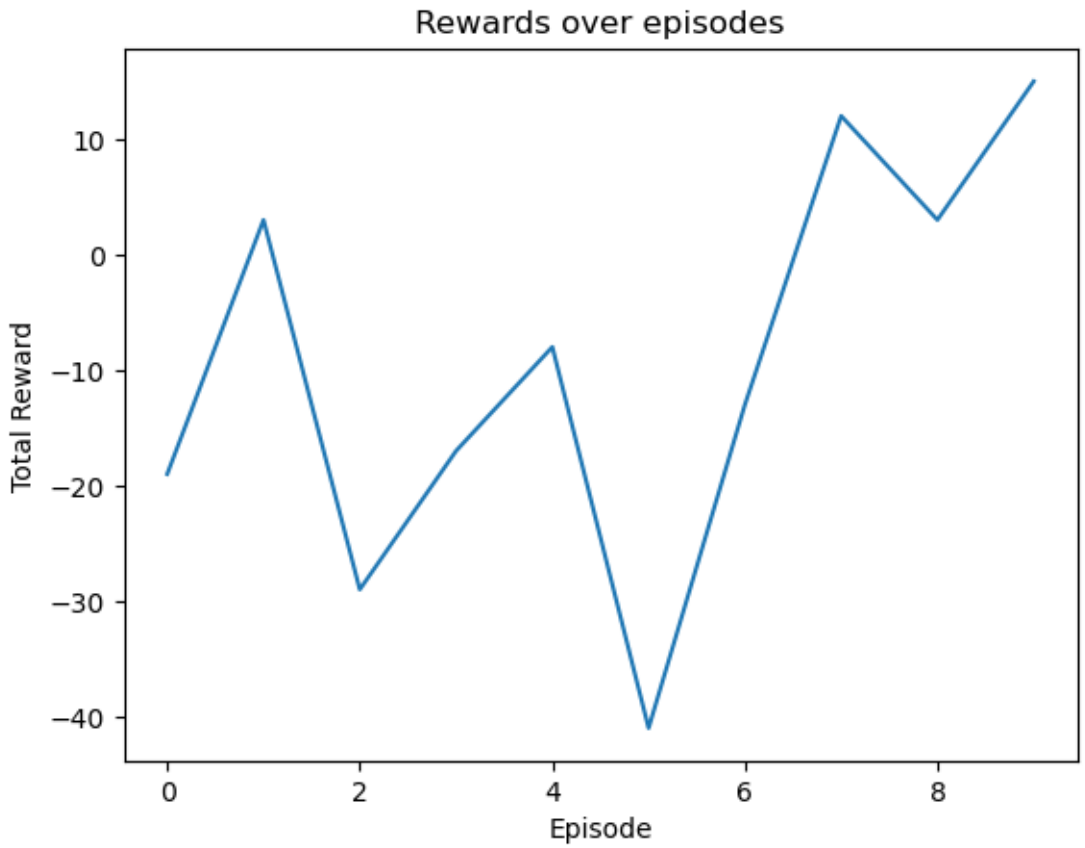
This graph indicates that the agent may have made poor decisions or faced penalties early on since it shows a rather erratic reward pattern with some initial negative payouts. The rewards do, however, become better quite rapidly, suggesting that the agent is either learning or adjusting to its surroundings. As the episodes go on, there is a noticeable fluctuation in the prizes, which go above and below a baseline. This might be a reflection of the agent exploring the state-action space and encountering costs associated with certain choices in addition to good results. Following the early instances, there has been a generally favorable trend that indicates some learning or policy optimization.

Training with γ=0.99 and ε_decay=0.95



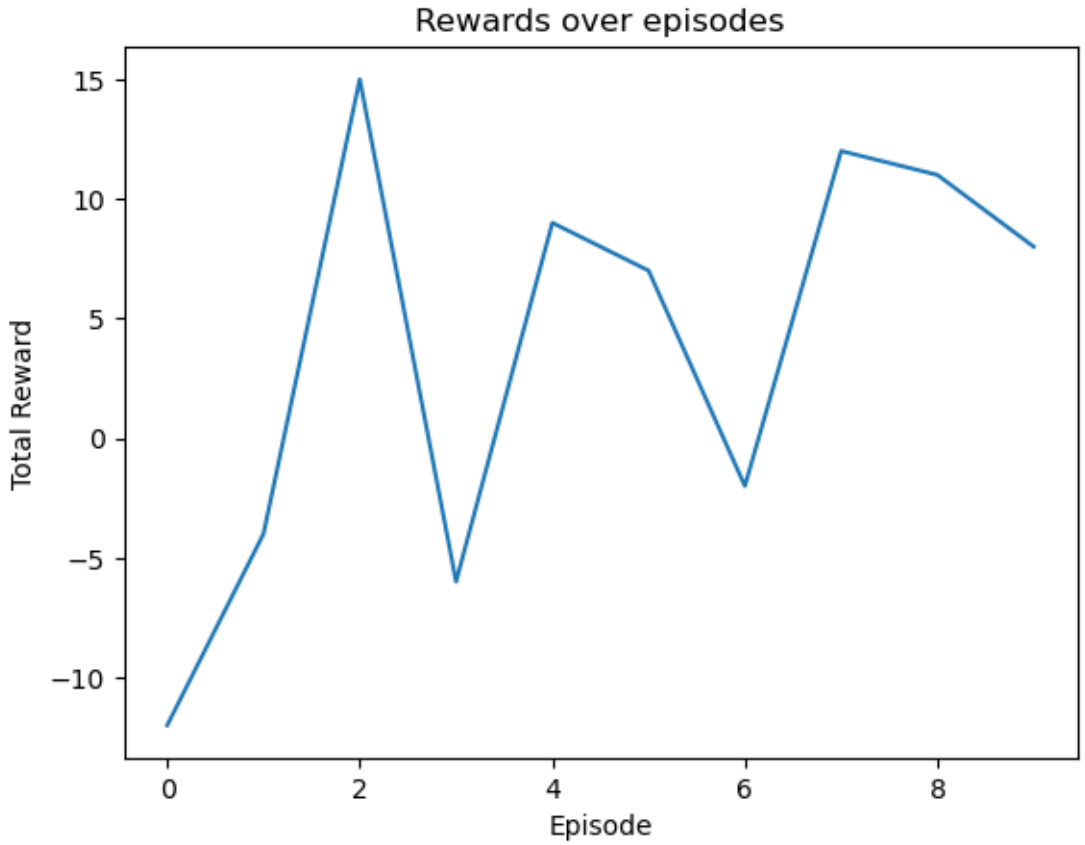
A similar trend can be seen in the above graph, which begins with a steep decline in rewards and then starts to rise. When compared to the left graph, the total rewards then stay positive and rather consistent. This implies that the agent may be constantly implementing a strategy that produces favorable results after learning from its early failures. In contrast to the left graph, the agent's activities seem to be producing a continuous state of rewards with less variation, suggesting a more stable or dependable policy.

Training with γ=0.99 and ε_decay=0.99



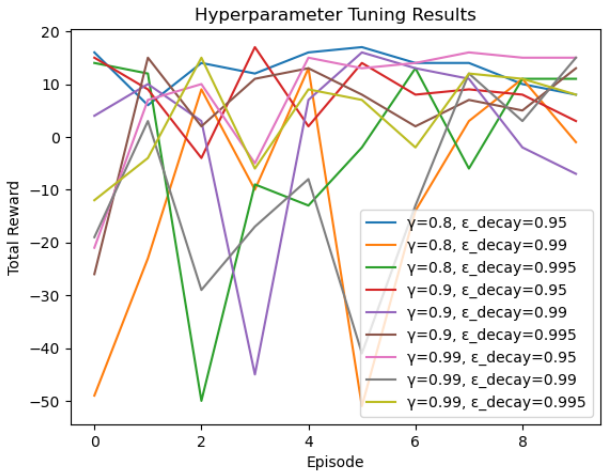
The graph shows a rewards pattern that is zigzag throughout the course of episodes. The reward total begins negatively at the beginning, suggesting that the agent may h ave made a number of poor judgments up front, but by the second episode, it has sw iftly rebounded to a positive value. The agent's erratic performance is shown by the f ollowing recovery and dip that follows this positive peak. This kind of pattern may be a reflection of the environment's complexity or variety, or it may arise during the learn ing phase while the agent is still figuring out which behaviors result in bigger rewards. The agent's performance does not return to the original low level despite the oscillatio ns, indicating some degree of learning and adaptability.

Training with γ=0.99 and ε_decay=0.995



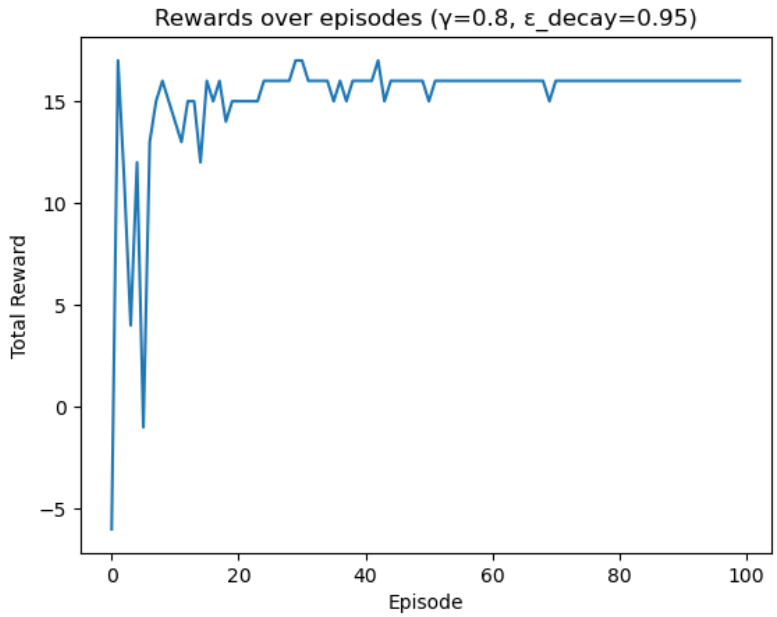
The graph displays varying patterns, but the left graph displays all values that are ov er the negative threshold. It has a notable positive reward at first, then falls just below zero, and then shows a pattern of peaks and troughs. Overall, the trend stays above t he baseline, and the agency seems to be closing on a high note. This trend might ind icate that the agent is improving its policy to maintain a typically good reward result, l earning from the environment, and making more educated judgments as it gets exper ience.

Based on the above observations, we can say that the best parameters were γ= 0.8 and ε_decay=0.95 with an average reward of 12.7



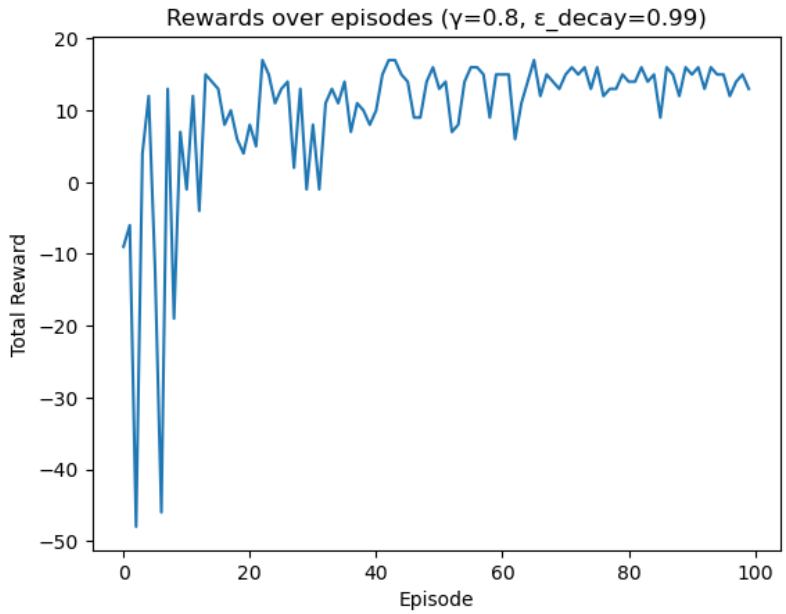
Double Q-learning:

γ=0.8 and ε_decay=0.95



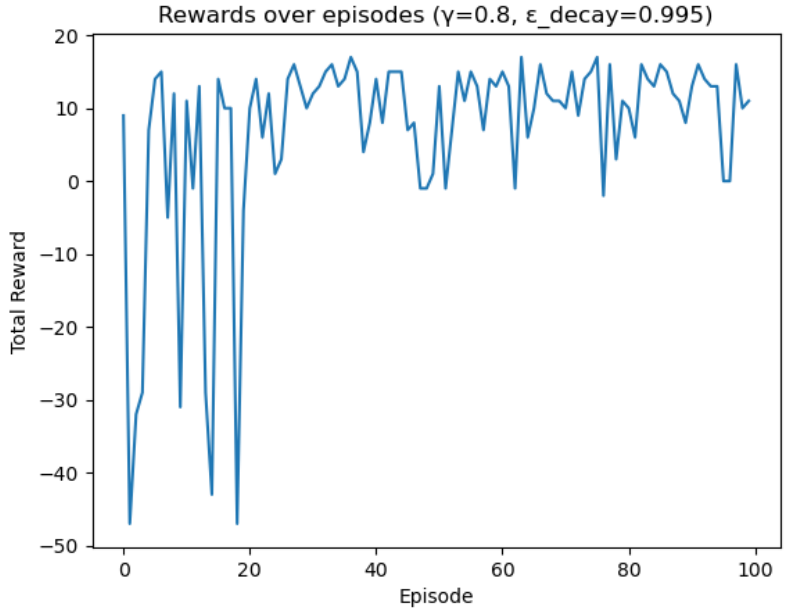
The pattern seen in this graph, with parameters γ=0.8 and ε_decay=0.95, is that the t otal reward initially displays great volatility, including negative rewards, but as the epi sodes go on, it stabilizes quite rapidly. The agent continuously receives positive rewa rds after the early oscillations, with very little variance in the latter stages. The compa ratively high ε_decay rate indicates that the agent swiftly steps back from exploration in favor of exploitation, which might result in fast policy stabilization but run the risk of leading to a suboptimal policy convergence if insufficient exploration is done at first.

γ=0.8 and ε_decay=0.99



The initial variance is comparable in the following graph with parameters γ=0.8 and ε _decay=0.99, but the rewards vary throughout the 100 episodes, although at a decre asing amplitude. This implies that because of the slower decay rate of ε, the agent st ays in the environment longer, which might result in a more gradual convergence and allow for a more exhaustive search for the best policy. By the conclusion of the 100 e pisodes, the agent's policy may not have totally stabilized, however, based on the on going shifts.

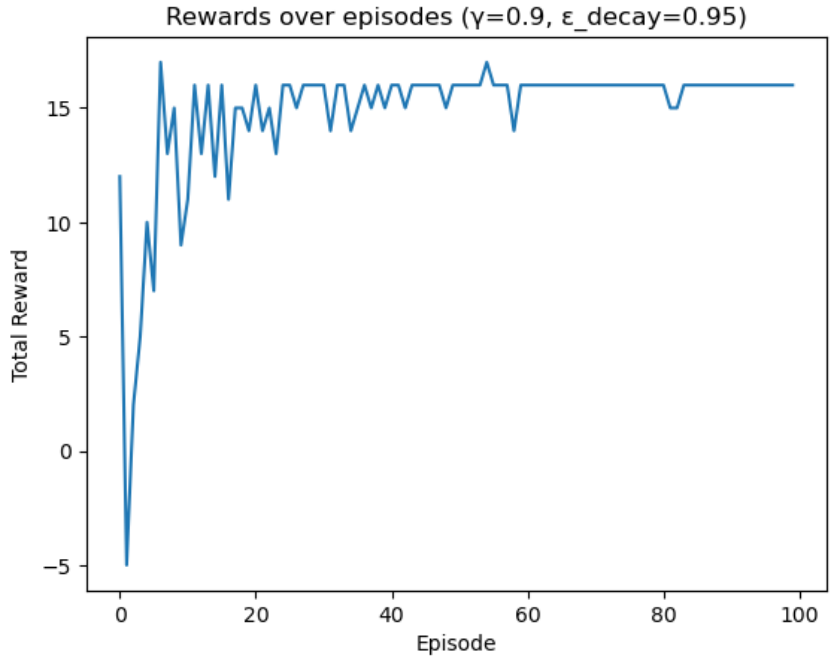
γ=0.8 and ε_decay=0.995



The first highly irregular reward pattern, which includes many dips into negative rewa rds, raises the possibility that the agent encountered challenging circumstances or m ade a string of bad choices early in the learning process. But after this early stage, as the episodes go on, the reward values exhibit less volatility and become more stable and favorable. With a high ε_decay number, the exploration rate seems to drop grad

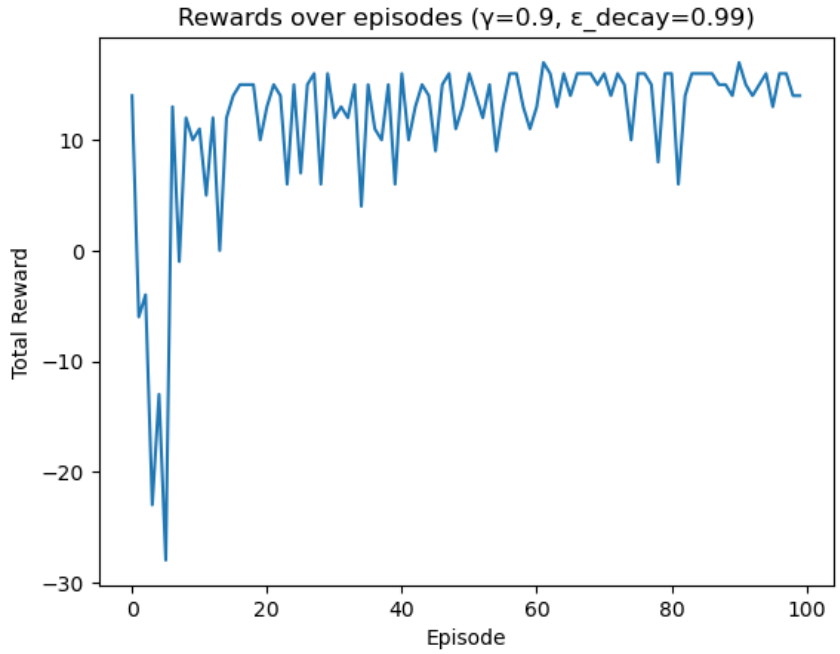
ually, enabling the agent to keep researching its surroundings and gradually enhanci ng its policy. The pattern suggests that the agent's performance is leveling off and m ay be approaching a stable policy that produces incentives.

γ=0.9 and ε_decay=0.95



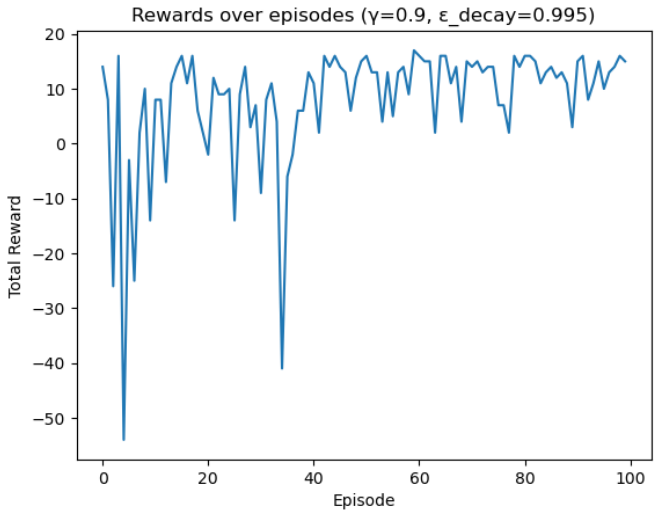
The graph shows a quicker drop in exploration with an ε_decay of 0.95 and a discoun t factor of γ=0.9, emphasizing future benefits. After experiencing some early volatility, the overall rewards quickly stabilize and eventually show a more constant level of po sitive payouts. This implies that the agent may identify an effective plan of action real ly fast and stick with it for the duration of the episodes. The agent may have taken ad vantage of the most well-known tactics earlier due to the exploration's quicker decay r ate, which produced a less erratic and more consistent reward pattern.

γ=0.9 and ε_decay=0.99



The graph displays the performance of a reinforcement learning agent with a modera te exploration decay rate (ε_decay=0.99) and a large discount factor (γ=0.9). At first, t he rewards exhibit considerable volatility, culminating in a notable decline into negati ve values, which suggests that the agent may have made some questionable choices or paid penalties. Nonetheless, the graph quickly stabilizes, preserving a usually posi tive reward with fewer oscillations. The slow decay rate shows that the agent keeps tr ying new things throughout the episodes, but it favors those that have already been r einforced. The large discount factor says the agent values future rewards highly.

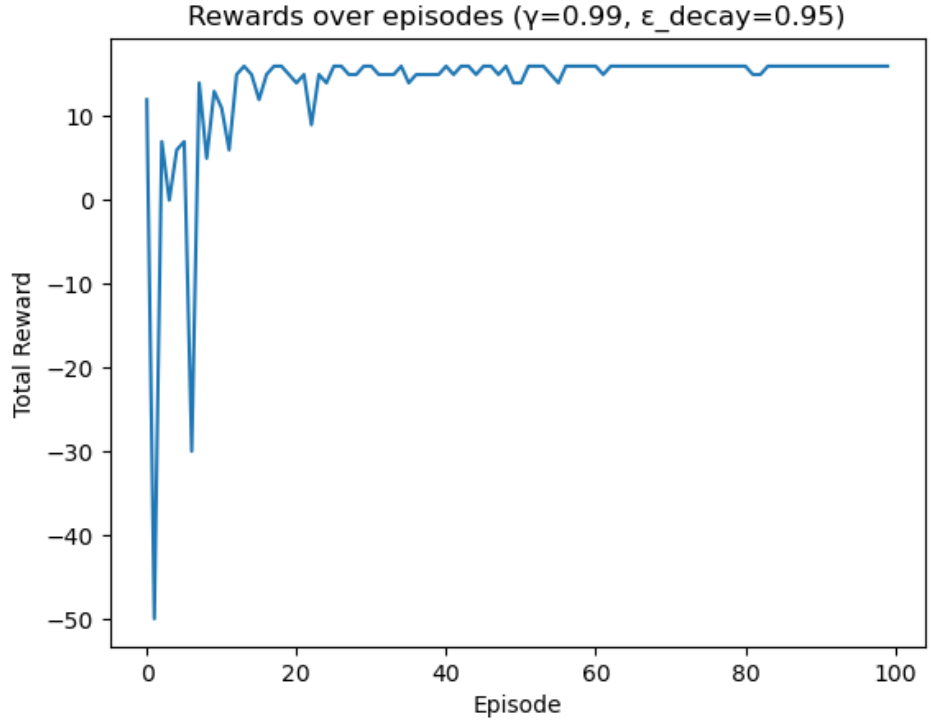
γ=0.9 and ε_decay=0.995



With a slower exploration decay rate (ε_decay=0.995) and the same discount factor (γ=0.9), the graph depicts an agent that experiences more notable swings in rewards, even as it moves towards later episodes. This decay rate is slower, allowing for longe r periods of investigation, which may be the reason for the ongoing variations in rewa rds. The general trend in the second half of the episodes is more steady and consiste ntly positive, despite the early volatility, indicating that the agent is still learning from t

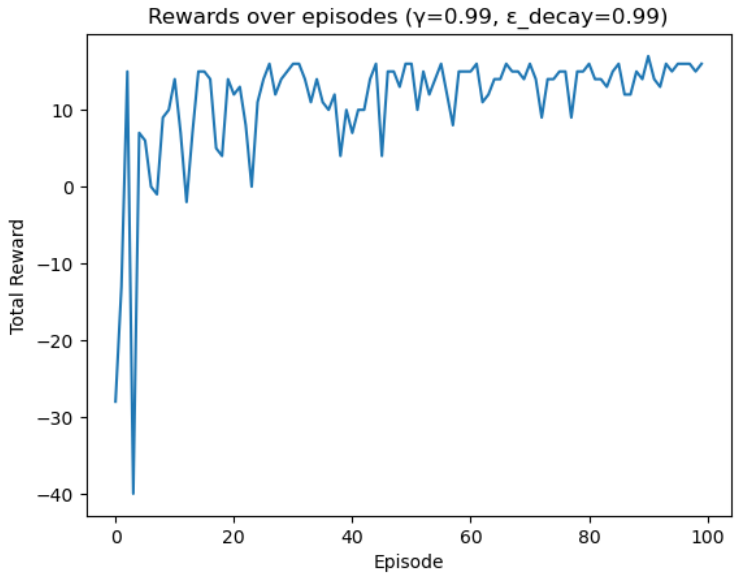
he environment and making adjustments to the policy. This suggests that learning is continuing.

γ=0.99 and ε_decay=0.95



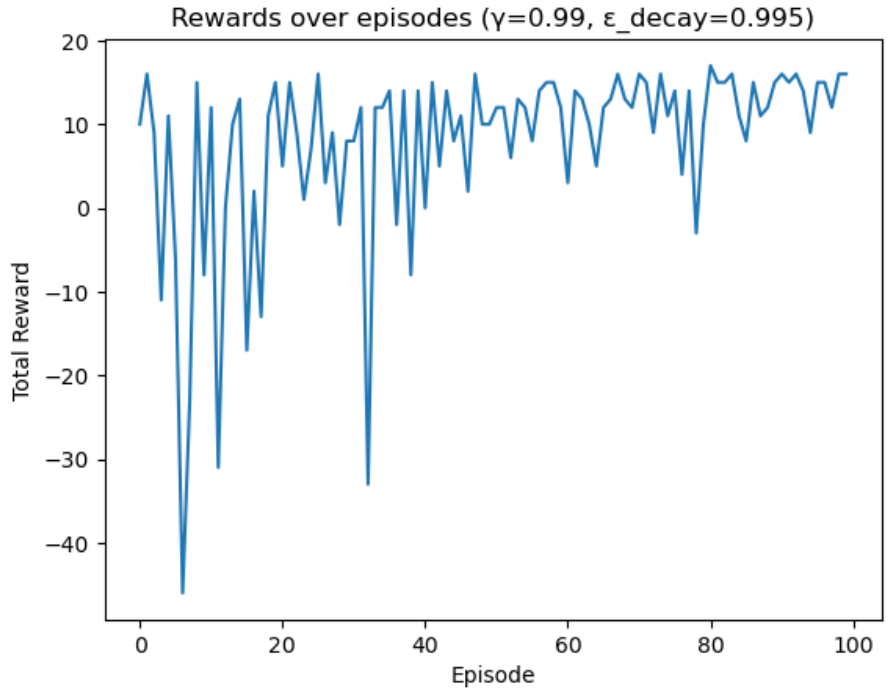
The graph shows an agent with a quicker exploration decay rate (ε_decay=0.95) and an even larger discount factor (γ=0.99) that strongly emphasizes future rewards. The performance of this agent declines at first, but it soon bounces back to a stable rewar d level with little oscillations. This may suggest that the agent is making good use of i ts learnt strategy, as the greater discount factor enables it to more efficiently optimize for long-term benefits. The more steady reward pattern shown after the first few episo des may be explained by the agent transitioning from exploration to exploitation more rapidly, as suggested by the quicker decay rate.

γ=0.99 and ε_decay=0.99

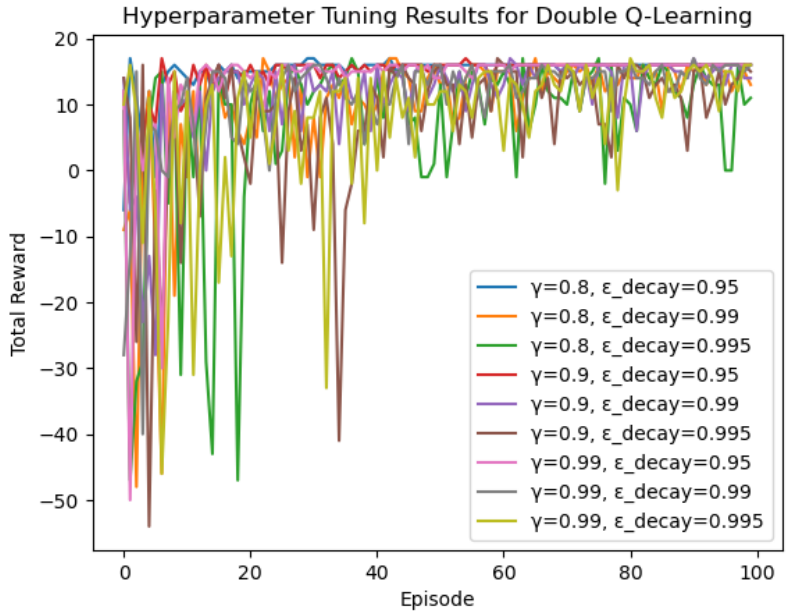


This graph indicates that the agent experienced a significant amount of volatility at th e beginning of the learning process, with some notable dips into negative rewards. H owever, as the episodes progress, the fluctuations in reward become less extreme, a nd the agent seems to reach a more consistent level of performance. The high disco unt factor (γ=0.9) suggests the agent values future rewards considerably, while the sl ow decay rate of exploration (ε_decay=0.99) points to the agent maintaining a higher level of exploration throughout the episodes. The graph demonstrates that the agent i s likely refining its policy over time and stabilizing its performance as it learns from th e environment.

γ=0.99 and ε_decay=0.995



In this graph, the agent has a somewhat slower exploration decay rate (ε_decay=0.9 95) and an even greater discount factor (γ=0.99), indicating that future rewards are hi ghly valued. Like the first graph, the rewards pattern exhibits early instability with sub stantial negative values. But as the learning process advances, the agent's performa nce shows less variation in reward values and ultimately levels off with an overall incr easing trend. The agent may be more inclined to consider the long-term effects of its decisions in this case due to the very large discount factor, which might result in a po licy that prioritizes rewards in the far future above the current course of events.



The image shows a line graph with the outcomes of the Double Q-learning reinforcem ent learning algorithm's hyperparameter adjustment. It displays over 100 episodes wi th different combinations of the exploration decay rate (ε_decay) and discount factor (γ). The graph displays a significant degree of early reward volatility, with many negati ve rewards indicating that choices were not ideal or that states were not as lucrative a s they might have been. The agent's policy is stabilizing and improving its ability to m ake profitable choices as episodes go on, as shown by the convergence of the lines a nd the reduction in volatility. Total payouts vary within a small range towards the conc lusion of the 100 episodes, indicating a steady policy producing steady benefits. This graph may be used to compare how well various hyperparameter settings work and t o see how they affect the learning process. Finding the configurations that result in th e biggest cumulative reward or the most reliable and stable policy is the aim.

4) the performance of both algorithms on the same environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.

Ans : We are comparing the Total reward per episode graph of SARSA and Double Q Algorithm

Learning Curve Shape: The learning curves on both graphs begin with a sharp rise in the overall reward for each episode, suggesting that both algorithms are picking up

knowledge from their early encounters. The curves plateau after the first few events, indicating that the agent is getting close to an ideal policy.

Initial Performance: When compared to Double Q-Learning, the SARSA algorithm seems to have a worse initial performance. The SARSA graph's deeper initial dip into the negative reward region serves as an indicator of this.

Convergence: The rewards for each episode in both graphs converge to a point where it appears the agents have figured out a stable strategy. The plateau region suggests that the agent is receiving the same amount of reward every episode, which may signal that while it learns more, it is not coming up with any more effective techniques.

Consistency and Variability: Early in the learning process, the rewards on the SARSA graph appear to be more variable, but they eventually stabilize. The Double Q-Learning graph, on the other hand, displays a more seamless transition to a stable policy. Double Q-Learning's smoother slope raises the possibility that it has a more reliable learning update mechanism or a less exploratory policy.

Final Performance: The overall reward plateaus at roughly the same value, suggesting that both algorithms achieve a similar level of performance by the conclusion of 1000 episodes.

Algorithm Comparison: Based on these graphs, it seems that Double Q-Learning takes less time than SARSA to arrive to a stable policy, as evidenced by the early peak on its curve. This may indicate that Double Q-Learning works better for this specific activity or setting.

References :

Q-Learning vs. SARSA | Baeldung on Computer Science

Temporal-difference RL: Sarsa vs Q-learning | by Kim Rodgers | Medium

Reinforcement Learning - Algorithms (unsw.edu.au)

Reinforcement Learning (DQN) Tutorial — PyTorch Tutorials 2.1.1+cu121 documentation

Contribution Chart :

Name	Assignment Part	Contribution(%)
Sricharan	All parts & Bonus	**50%**
Mahitha	All parts & Bonus	**50%**