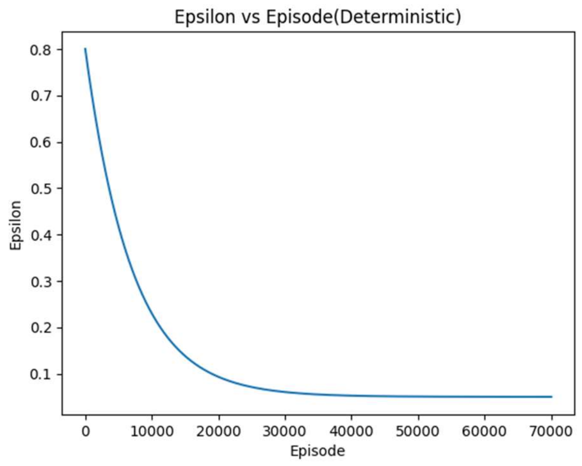
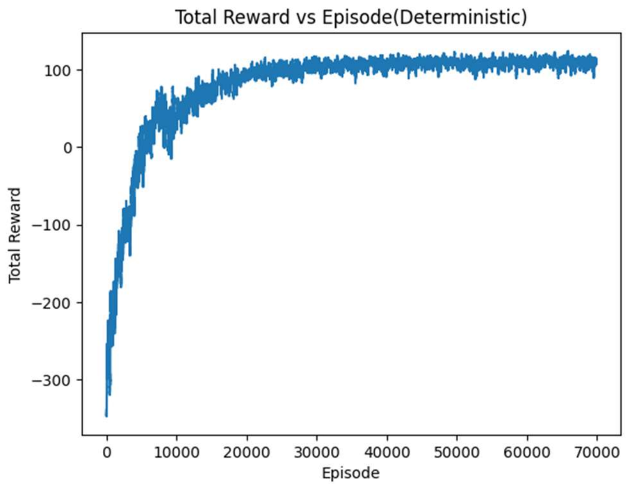
Part II: Results and Discussion

1. Applying Q-learning to Solve the Deterministic Environment

Epsilon Decay and Total Reward per Episode

The Q-learning algorithm was applied to the deterministic environment defined in Part 1. The following plots illustrate the epsilon decay and the total reward per episode during training:





Discussion

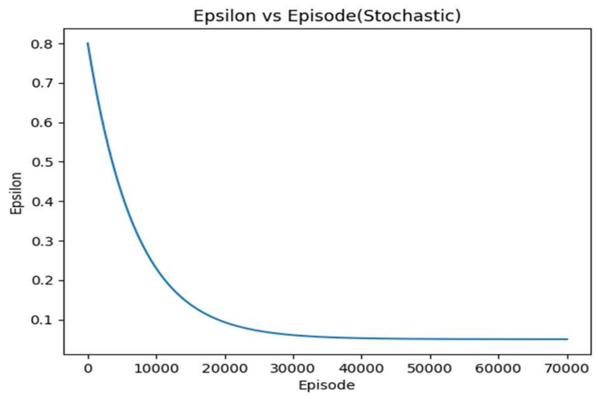
The epsilon decay plot illustrates how exploration diminishes with time, resulting in increased exploitation of acquired policies.

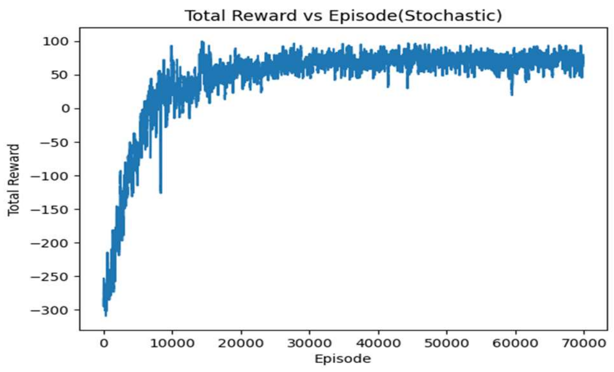
The plot of total reward per episode illustrates the learning of the agent, with rewards converging as it learns an optimal policy.

2. Applying Q-learning to Solve the Stochastic Environment

Epsilon Decay and Total Reward per Episode

The stochastic environment introduces randomness in state transitions or rewards, making learning more challenging. The results are depicted below:



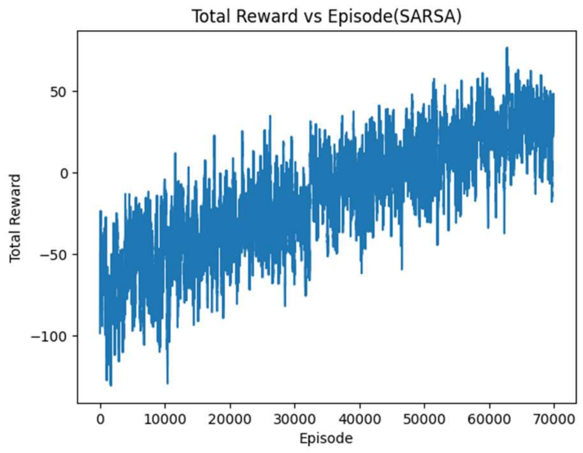


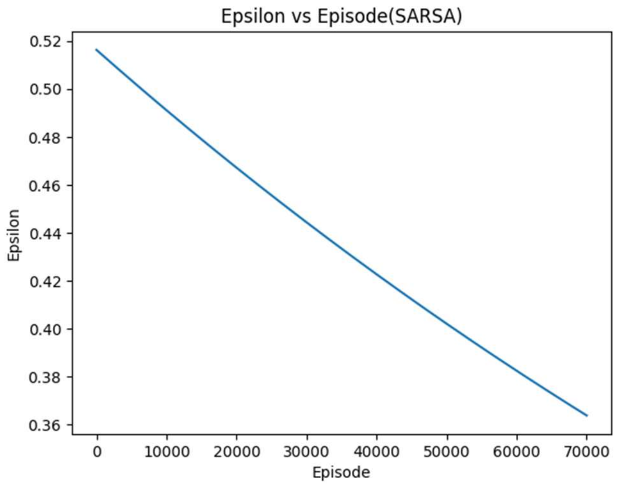
Discussion

The epsilon decay follows the same trajectory as the deterministic case, but the cumulative reward per episode differs due to the stochasticity of the environment.

Convergence is slower, and the reward curve is more variable compared to the deterministic case.

3. Applying an Alternative Algorithm SARSA to the Environment



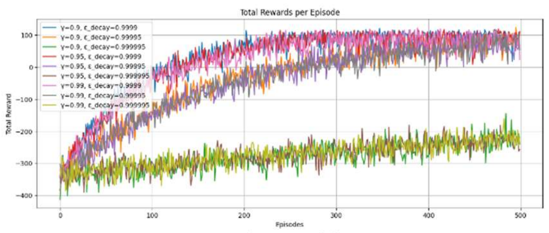


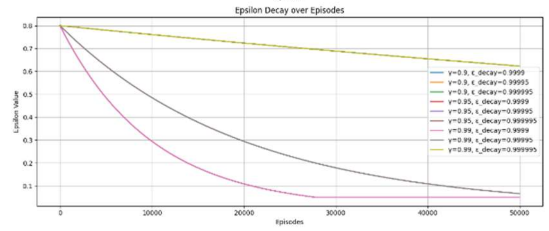
Discussion:

The SARSA algorithm results in a more conservative learning approach compared to Q-learning.

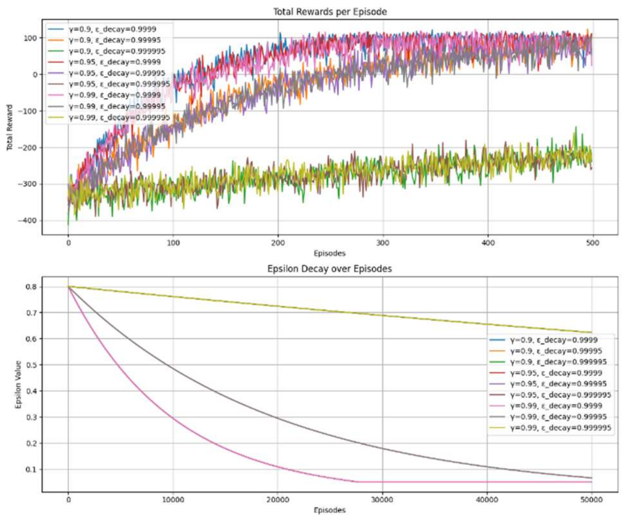
The total reward per episode stabilizes, but it may achieve a slightly different policy than Q-learning due to the on-policy nature of SARSA.

Hyper Parameter Tuning:



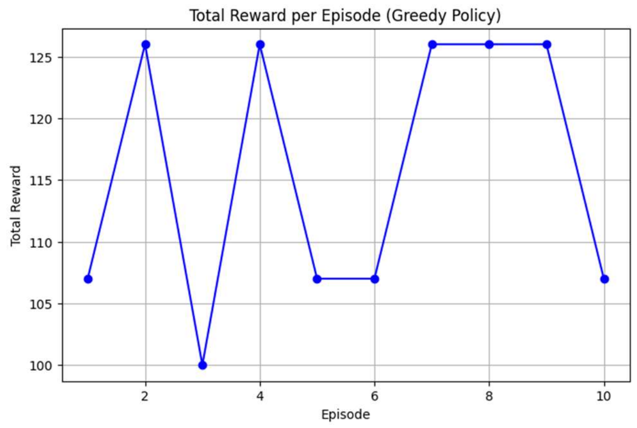


Error! No text of specified style in document.*-1Stochastic Environment*



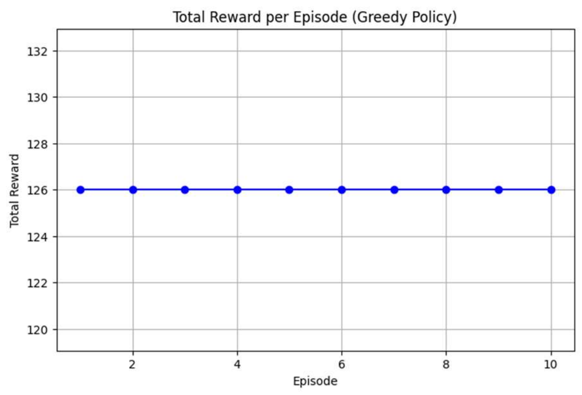
Error! No text of specified style in document.*-2Deterministic Environment*

5. Evaluation Results



Error! No text of specified style in document.*-3Stochastic Environment*

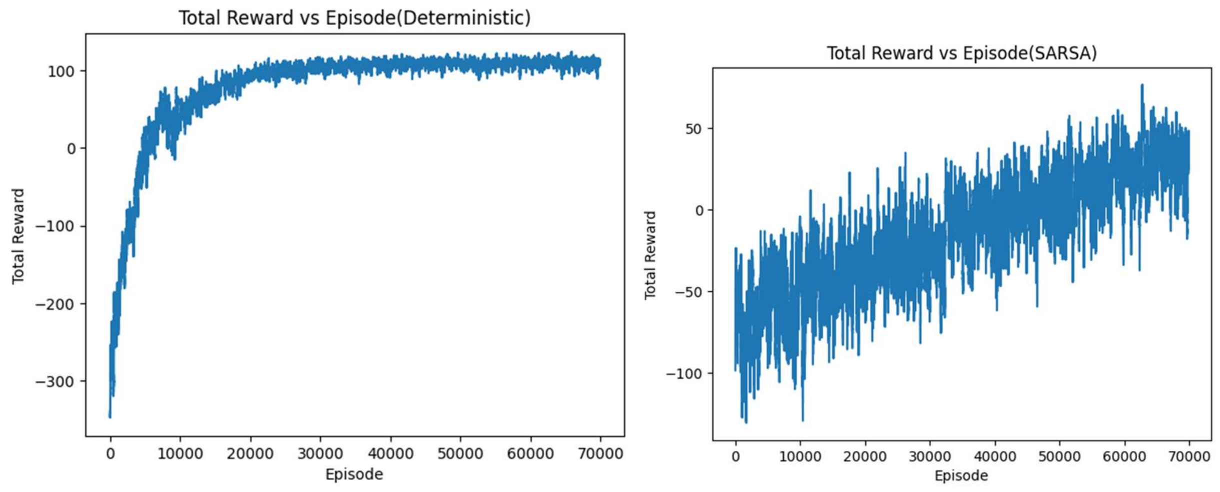
The total reward fluctuates significantly across episodes, indicating some inconsistency in performance.
The agent achieves high rewards in some episodes (around 125-127), but in others, it drops significantly.
This suggests that the learned policy might still have some instability, potentially due to the stochastic nature of the environment.



Error! No text of specified style in document.*-4Determinsitic Environment*

The total reward is consistent for all episodes, averaging about 126.
This shows that the agent has acquired a stable policy despite the stochastic environment.
A consistent reward implies that
the acquired policy executes consistently optimally in different episodes.

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



Q-learning Performance:

The cumulative reward rises quickly early on and becomes constant after around 20,000 episodes.

The curve is smoother, reflecting that the agent acquires a best policy with greater efficiency.

The last reward is always greater, reflecting enhanced general performance.

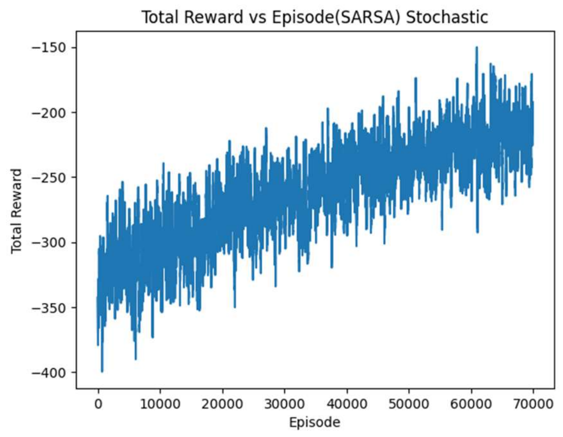
SARSA Performance:

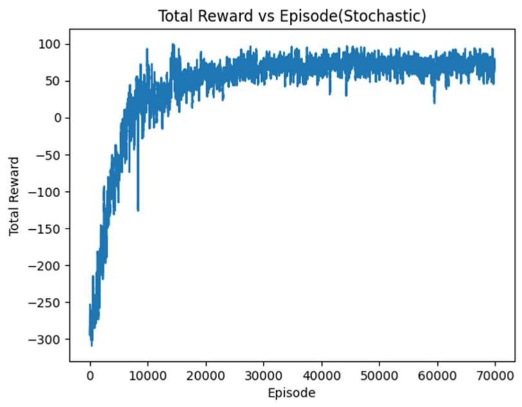
The reward rises steadily and has greater oscillations over training.

SARSA's on-policy character causes it to be more cautious, resulting in learning at a slower rate.

While SARSA does finally learn well, it stabilizes much more slowly.

Compare how both algorithms perform in the same stochastic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.





Q Learning (Stochastic):

The total reward increases rapidly in the early episodes, stabilizing around a higher reward threshold.
It reaches an optimal policy relatively quickly, demonstrating strong convergence properties.
The reward remains consistent with minor fluctuations after convergence, indicating that Q-learning effectively finds a near-optimal policy.

SARSA (Stochastic)

The reward increases more gradually and remains more volatile throughout training.
It does not achieve as high a final reward as Q-learning.
SARSA is an on-policy method, meaning it considers the policy's actual actions, leading to more cautious learning and slower convergence.

Tabular Methods of RL:

Tabular methods store and update value estimates in a table, typically for small state-action spaces. They include Q-learning and SARSA, both of which utilize the Bellman equation to update.

Q-Learning:

Update rule:

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max Q(s',a') - Q(s,a)]}$$

Key Features:

Off-policy: Learns the optimal action-value function independently of the followed policy.
Uses the max Q-value for next state: This choice renders it more aggressive in learning the optimal policy.
Converges faster but can be unstable in stochastic environments.

SARSA:

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]}$$

Key Features:

On-policy: Learns from real actions taken under a given policy.
More stable but slower in convergence.
More appropriate in situations where exploration-exploitation trade-off is relevant (e.g., avoiding dangerous actions).

Briefly explain the criteria for a good reward function. If you tried multiple reward functions, give your interpretation of the results.

Criteria for a Best Reward Function
An appropriate reward function is crucial for effective reinforcement learning (RL). Following are some significant criteria:

Meets the Objective – The reward should encourage the desired behavior leading to the objective.
Offers Ongoing Feedback – Ineffective sparse rewards can retard learning; well-shaped rewards lead to quicker convergence.
Prevents Unintended Exploits – The function should prevent agents from exploiting bugs to gain rewards in the inappropriate manner.
Balances Exploration & Exploitation – It should guide the agent to the optimal solutions without discouraging exploration.
Stable but Adaptive – It should avoid inducing extreme variations in learning, but should be able to do adaptive improvements.