Comparison of SARSA and DQN Algorithms in a Deterministic and stochastic Frozen Lake Environment

Presented to professor: **Sidney Givigi**

Group: 3

By: **Amr Mostafa Ibrahim**

1. Introduction

The motivation for this project comes from the need to address complex navigation tasks in unpredictable conditions. The Frozen Lake environment provides a practical example of how agents can be trained to make decisions that avoid pitfalls and achieve goals, like real-life situations such as guiding autonomous vehicles or robots along tricky paths. By comparing how well these methods work in this grid, we can better understand their strengths and how they might be improved for more complex and unpredictable environments.

This report investigates how two learning methods can handle the challenge of navigating a Frozen Lake environment. In this setting, agents must find their way from a starting point to a goal while avoiding dangerous holes.

2. Problem Formulation

The Frozen Lake environment is set up using OpenAI's Gym library. In this setup:

State-Space: The environment is an 4x4 grid where each cell represents a state. The agent starts at a designated start position, aims to reach a goal position, and must avoid specific cells that represent holes.

Action-Space: The agent can choose from four possible actions at each step: up, right, left, or down.

Reward Scheme: The rewards are designed as follows:

- Each step taken by the agent results in a reward of zero.
- Reaching the goal state results in a reward of +10.
- Moving into a hole or making an invalid move results in a penalty of -10.

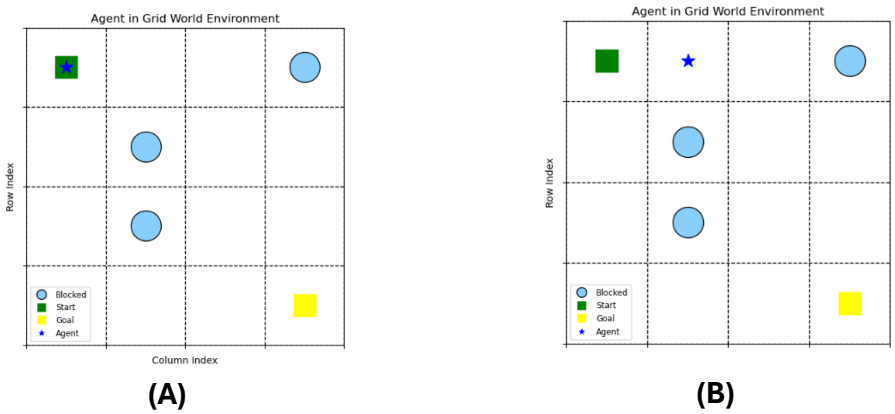


Fig.1 The Frozen Lake environment. **A**) Initial state for agent (0, 0). **B**) agent reached state (0, 1), received reward (0) and it will receive -10 in case of holes and 10 if it reaches to goal at (3,3).

3. Solution Overview

The solution involves applying two methods to help an agent navigate an 4x4 grid world with specific start, goal, and blocked cells. The first method, SARSA, helps the agent learn by updating its knowledge based on its own experiences and exploring different strategies. Key aspects include adjusting how quickly it learns and balances exploration with exploiting known paths. The second method, Deep Q-Network (DQN), uses a neural network to estimate the best actions for the agent to take. This method improves learning stability by using past experiences and a separate target network to avoid instability.

We tested various parameters for both methods to see how well they work. This included how many steps it takes and visualizing its journey through the grid. By comparing the results, we can determine which method and settings provide the best performance in navigating the grid world.

The remainder of this report is structured as follows: **Section 4** discussed algorithms that we implemented **Section 5** illustrate criteria that we used in our comparison between different algorithms **Section 6** mention results and discussed it with deterministic and stochastic environment then show the effect of changing parameters such as (epsilon, alpha, gamma and noise) Finally, **Section 7** concludes the report with a discussion on findings.

4. Algorithms

SARSA Algorithm (State-Action-Reward-State-Action)

It is a technique in reinforcement learning where an agent learns through its interactions with the environment. The agent follows a specific plan, earns rewards, and gradually improves its decision-making ability. This method is known as "on-policy" because it updates its knowledge based on actions taken under its current plan. The process begins by setting initial Q-values, which indicate the value of performing certain actions in specific situations, and these values are adjusted as the agent explores. At each stage, the agent notes its current state, chooses an action, receives a reward, observes the new state and the action it would choose next, and updates the Q-value using these observations. This update considers both the immediate reward and the expected value of future actions, ensuring the learning stays consistent with the current plan.

DQN Algorithm (Deep Q-Network**)**

It is a technique in reinforcement learning where an agent learns by interacting with its environment using a neural network to approximate its value function. The agent follows a strategy, accumulates rewards, and refines its approach to improve future actions. This method is known as "off-policy" because it updates its knowledge based on actions that may not follow the current strategy. The process begins by initializing a neural network to approximate Q-values, which represent the value of taking specific actions in given states. As the agent explores, it stores experiences in a replay buffer. At each step, the agent observes its current state, selects an action based on the network's predictions, receives a reward, and observes the new state. It then updates the network by sampling from the replay buffer and adjusting the Q-values according to the observed rewards and estimated future values. This update aims to minimize the difference between the predicted and target Q-values, ensuring the learning process aligns with the historical experiences stored in the buffer.

5. Evaluation Criteria

To effectively compare the performance and efficiency of SARSA and DQN algorithms, we use various evaluation criteria. These criteria offer a detailed analysis of each algorithm's strengths and limitations, assisting in assessing their overall effectiveness and practical utility. The primary evaluation metrics are:

Learning Time: This measures how long it takes for the algorithm to reach an optimal policy. It reflects the efficiency of the learning process and how swiftly the agent can determine the best actions to take in the environment.

Steps: This represents the total number of steps the agent takes to achieve the goal from the starting point. It indicates the efficiency of the agent's path, with fewer steps signifying a more effective route.

Average Reward: This metric computes the average reward obtained by the agent over several episodes. It provides an overview of the agent's performance and the effectiveness of its policy, with higher average rewards signifying better performance.

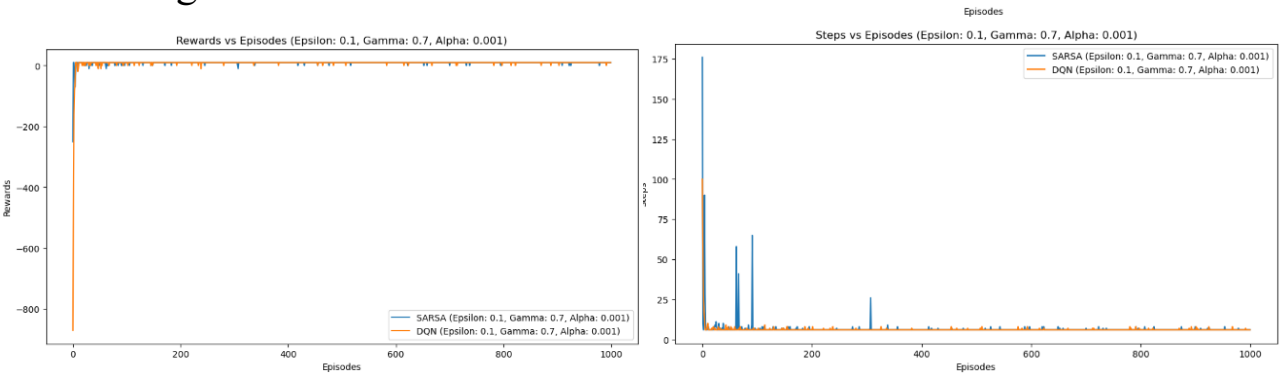
6. Results and Discussion

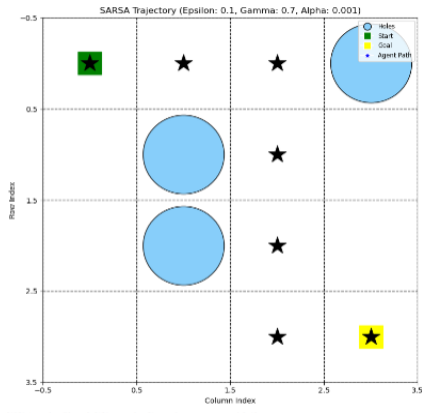
6.1 Deterministic Environment (Noise =0)

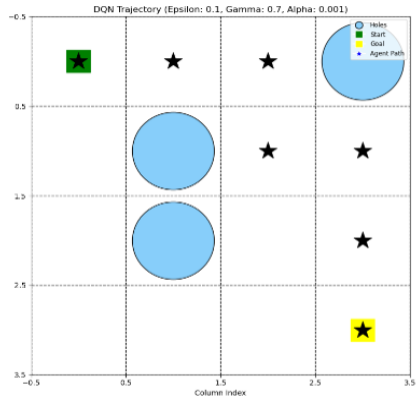
➢ Effect of changing epsilon value

Epsilon = 0.1

- **SARSA** reached an optimal policy in 0.0181 seconds, demonstrating rapid convergence. The agent took 6 steps to achieve the goal, showing an efficient path, though not the shortest possible. With an average reward of 9.12, it consistently navigated towards the goal successfully, avoiding penalties.
- **DQN** completed its convergence in only 0.0180 seconds, highlighting its effective learning efficiency. The agent reached the goal in just 6 steps, demonstrating a direct and optimal path. However, despite its speed and directness, DQN achieved a lower average reward of 7.76 compared to SARSA. This difference may indicate that DQN took more risks or encountered slightly more penalties during its path to convergence.

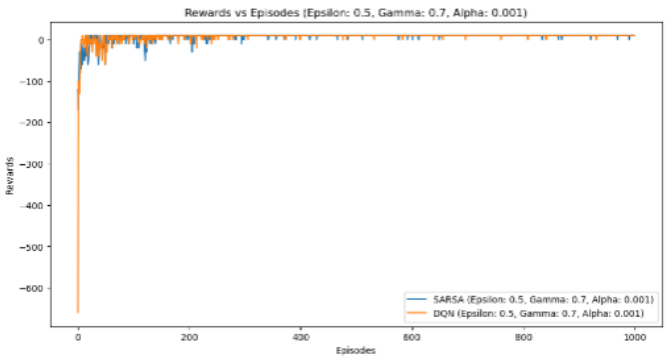


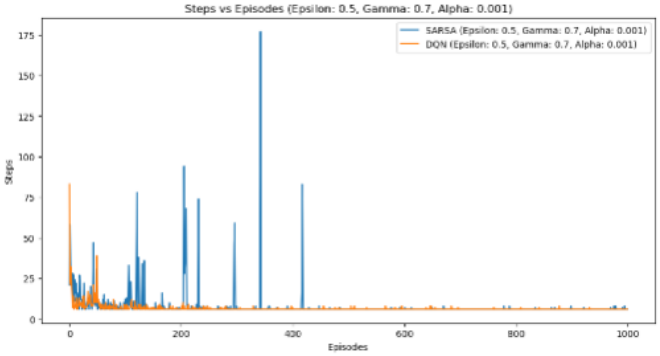


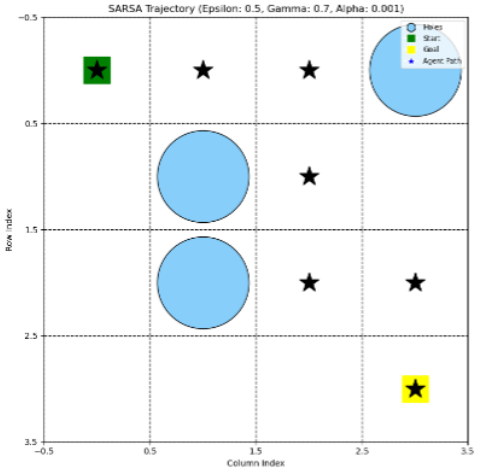


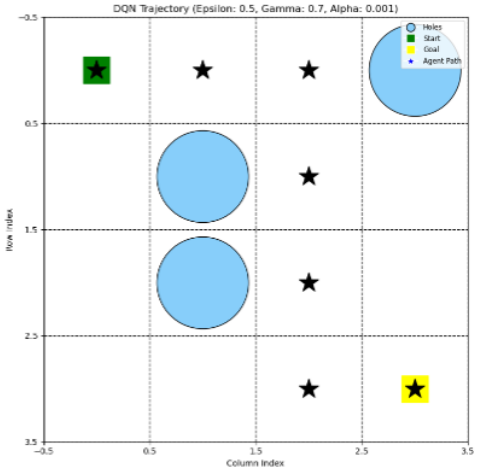
Epsilon = 0.5

- **SARSA** reached convergence in 0.0173 seconds, demonstrating efficient learning. The agent took 6 steps to achieve the goal, maintaining a consistent pathfinding performance. However, with an average reward of 6.99, the reward decreased compared to the lower epsilon value of 0.1, indicating that the increased exploration led to less optimal decisions and potential penalties.
- **DQN** achieved convergence in 0.0167 seconds, reflecting effective learning efficiency. The agent also reached the goal in 6 steps, like SARSA. With an average reward of 7.02, DQN outperformed SARSA under the same epsilon setting, suggesting better management of the exploration-exploitation balance. This slightly higher reward indicates that DQN was able to mitigate some of the penalties associated with higher exploration rates.



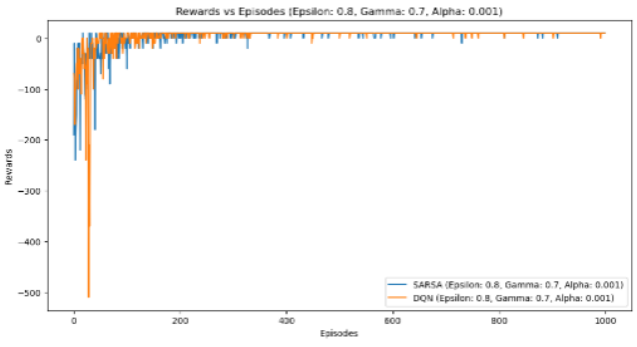


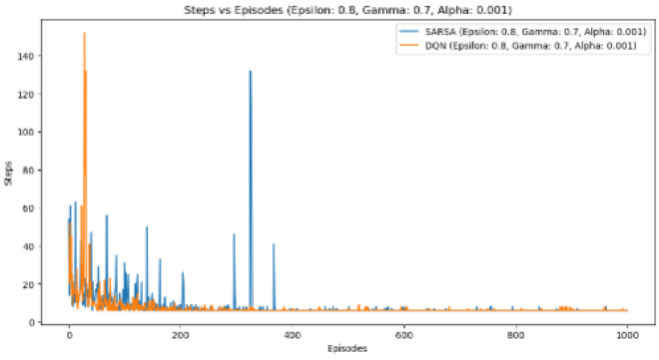


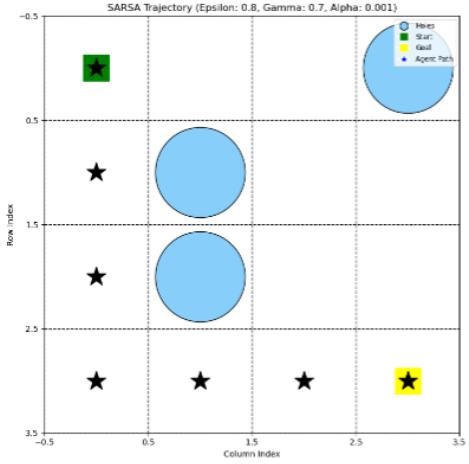


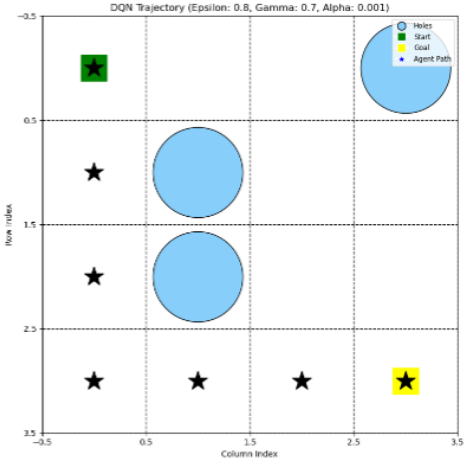
Epsilon = 0.8

- **SARSA** achieved convergence in 0.0164 seconds, demonstrating rapid learning. The agent took 6 steps to reach the goal, maintaining consistent pathfinding performance. However, with an average reward of 4.47, there was a noticeable drop compared to lower epsilon values, indicating that the increased exploration led to suboptimal decisions and higher penalties.
- **DQN** reached convergence in 0.0165 seconds, showing comparable learning efficiency to SARSA. The agent also reached the goal in 6 steps. With an average reward of 4.59, DQN slightly outperformed SARSA under the same epsilon setting, suggesting a marginally better balance between exploration and exploitation. The higher average reward indicates that DQN managed exploration-related penalties slightly better.







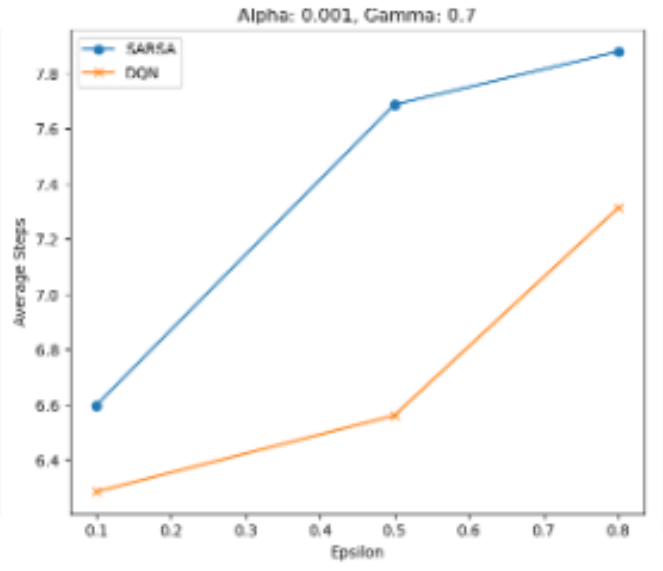


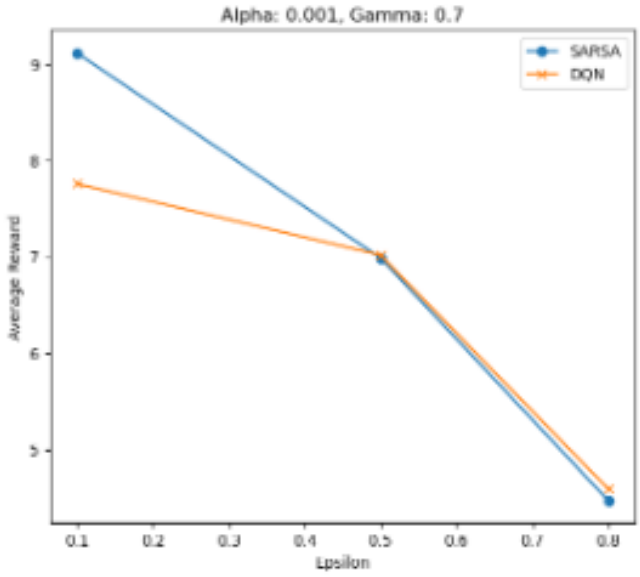
Effect of Changing Epsilon Value

As the epsilon increases, both SARSA and DQN show quicker convergence but with reduced rewards. With **epsilon = 0.1**, SARSA converged in 0.0181 seconds with a high reward of 9.12, while DQN converged slightly faster at 0.0180 seconds but had a lower reward of 7.76. At **epsilon = 0.5**, SARSA's convergence time improved to 0.0173 seconds, but its reward dropped to 6.99. DQN also improved its convergence time to 0.0167 seconds and achieved a better reward of 7.02. At **epsilon = 0.8**, SARSA converged in 0.0164 seconds but with a much lower reward of 4.47. DQN's convergence time was 0.0165 seconds with a slightly better reward of 4.59.

Optimal Policy

Both SARSA and DQN managed to reach the goal in 6 steps regardless of epsilon. With **epsilon = 0.1**, SARSA had a high reward, indicating efficient learning, while DQN had a lower reward, suggesting riskier actions. At **epsilon = 0.5**, SARSA's reward decreased, showing less optimal decisions due to increased exploration. DQN performed slightly better in this setting. With **epsilon = 0.8**, both algorithms had significantly lower rewards, reflecting that high exploration led to more penalties, with DQN slightly outperforming SARSA.



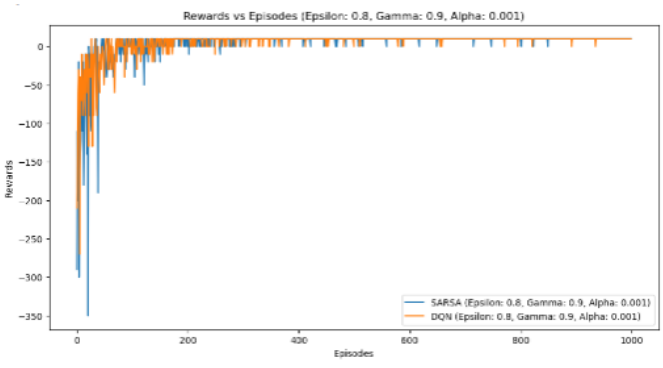


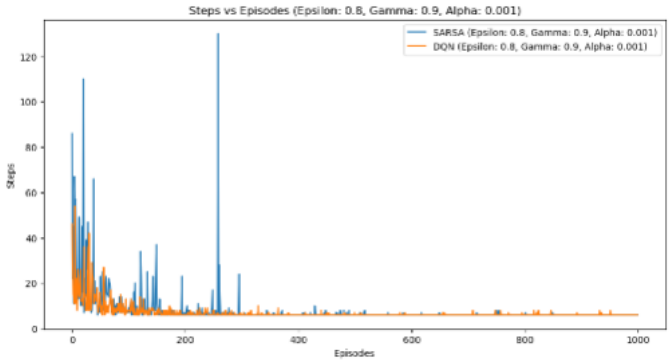
➢ Effect of changing alpha value

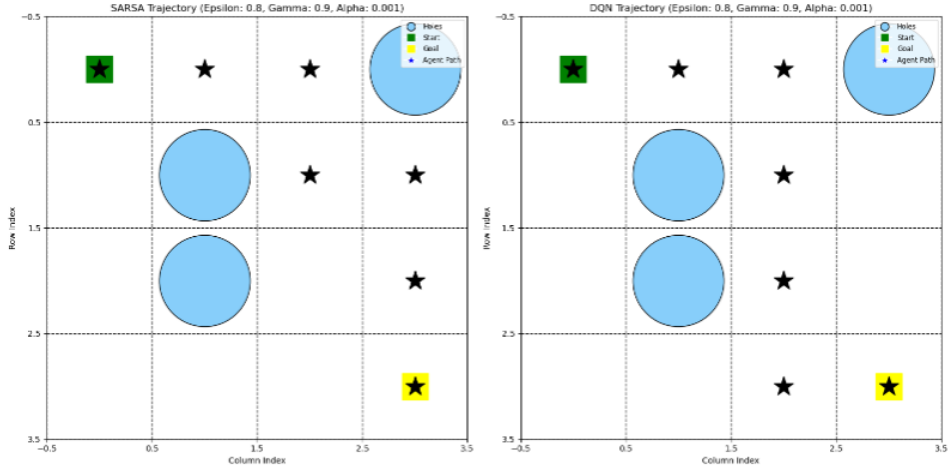
Alpha = 0.001

SARSA took 0.0173 seconds to converge, maintaining the goal-reaching steps at 6 but with a lower average reward of 4.12. This suggests that a very small alpha led to slower updates and less effective learning, resulting in a reduced reward.

DQN converged slightly faster in 0.0163 seconds with the same number of steps but achieved a higher average reward of 5.17. This indicates that DQN managed to balance learning efficiency and exploration better, even with a low alpha value.



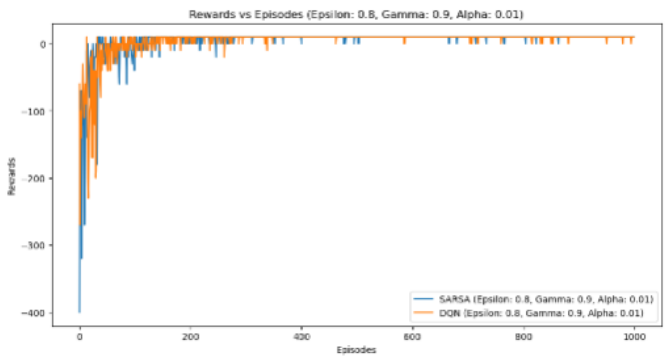


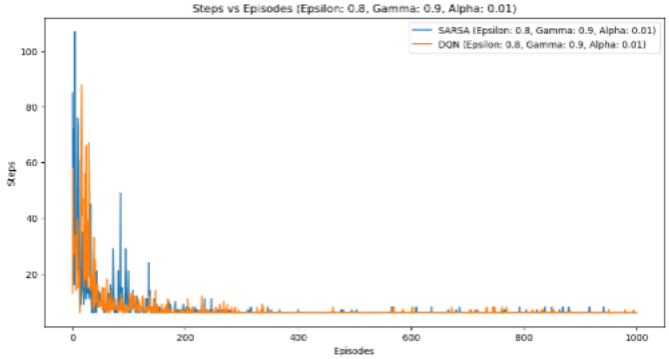


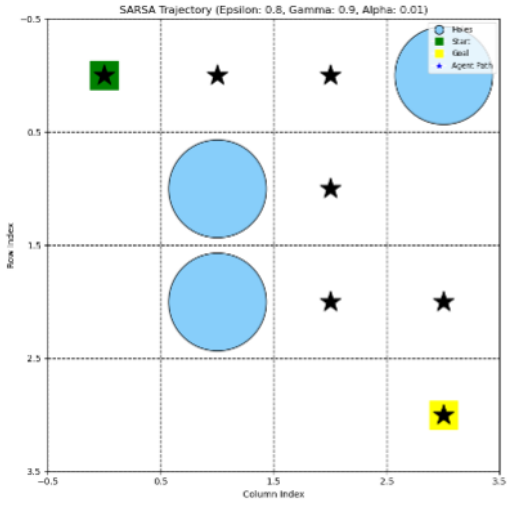
Alpha = 0.01

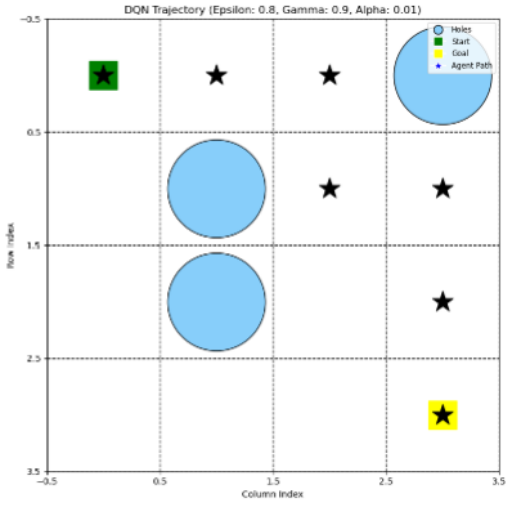
SARSA converged in 0.0161 seconds, taking 6 steps to reach the goal, and achieved an average reward of 4.68. This indicates that a slightly higher alpha value improved the learning speed and reward compared to alpha = 0.001. On the other hand,

DQN took 0.0174 seconds to converge with the same number of steps but had a lower average reward of 4.08. This suggests that, under the same conditions, DQN performed slightly worse in terms of reward, even though it had a slower convergence time.





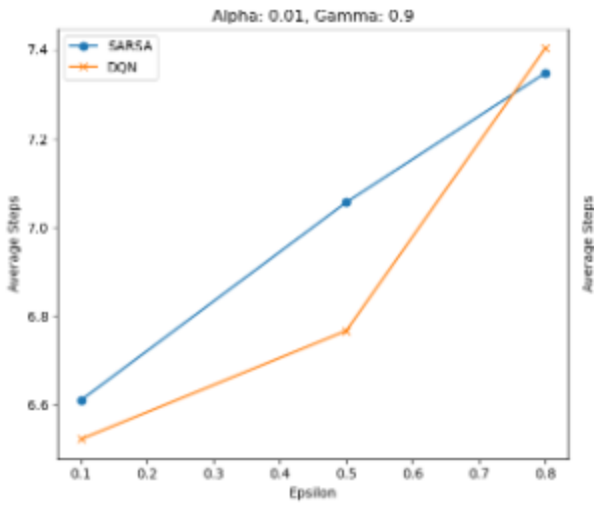


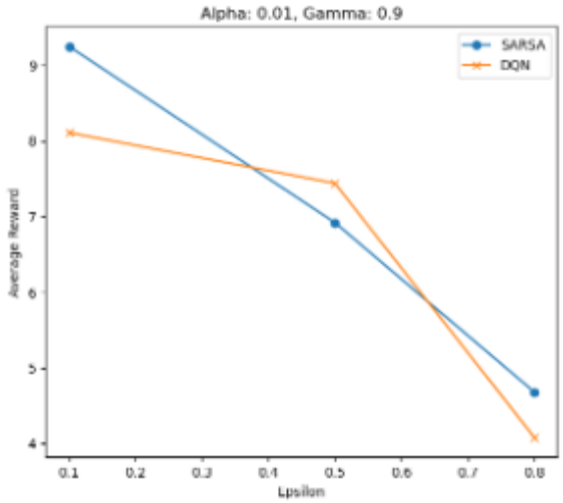


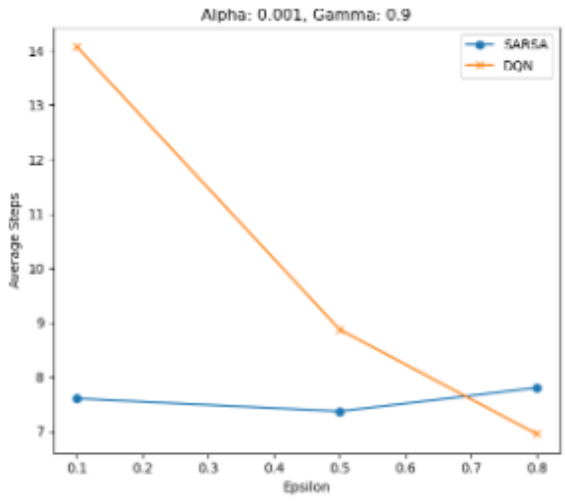
Effect of Changing Alpha

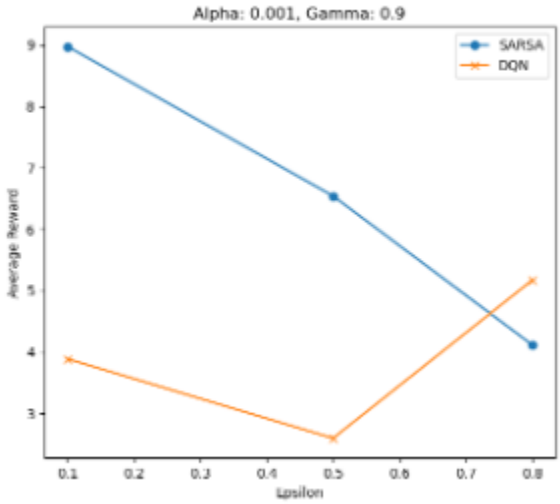
With **alpha = 0.001**, SARSA took 0.0173 seconds to converge and achieved a lower reward of 4.12. This indicates slower learning and less effective updates. DQN, however, converged faster in 0.0163 seconds and earned a higher reward of 5.17, showing better balance in learning efficiency.

When **alpha = 0.01**, SARSA converged a bit faster in 0.0161 seconds and had a better reward of 4.68, indicating improved learning speed and effectiveness. On the other hand, DQN took longer to converge (0.0174 seconds) and had a lower reward of 4.08, suggesting it didn't handle the higher alpha value as well.







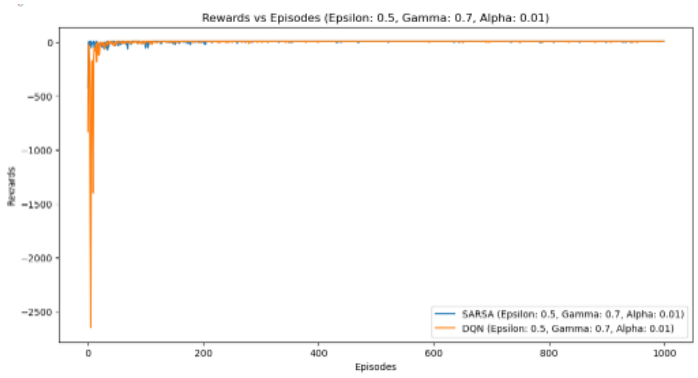


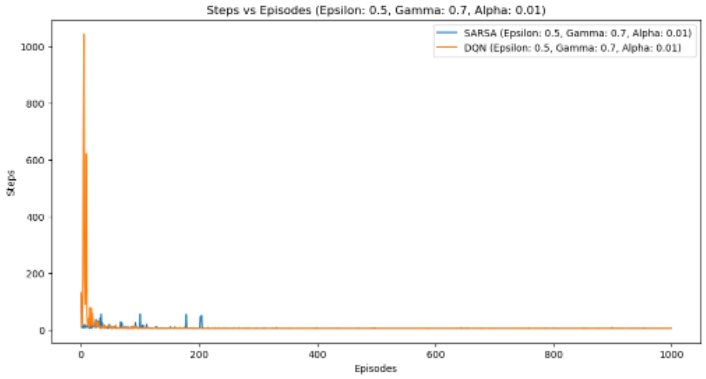
➢ Effect of changing gamma value

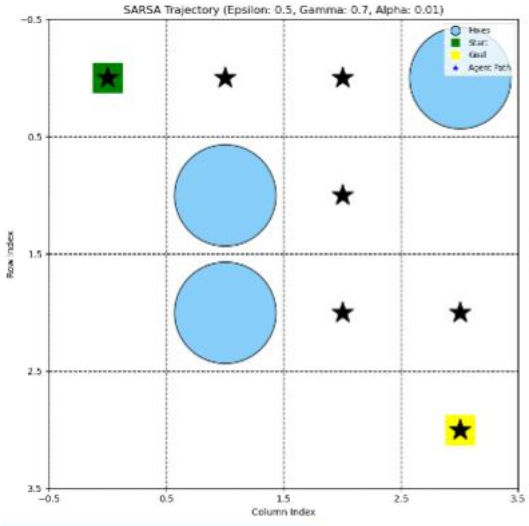
Gamma = 0.7

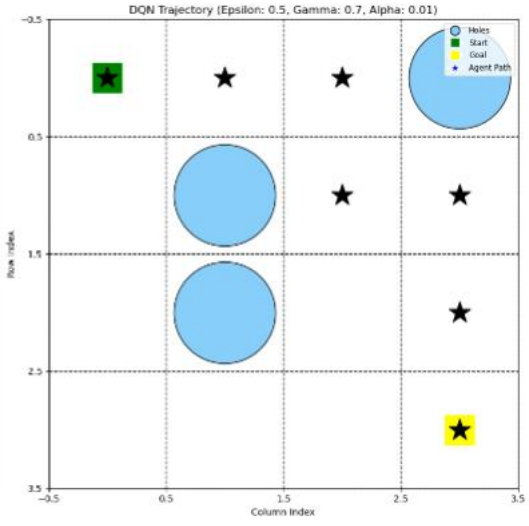
SARSA converged in 0.0169 seconds, took 6 steps to reach the goal, and achieved an average reward of 7.02. This shows effective learning and decent performance.

DQN also converged quickly in 0.0165 seconds with the same number of steps, but its average reward was significantly lower at -0.55. This suggests that, despite similar convergence times, DQN struggled with the reward optimization under this gamma setting.





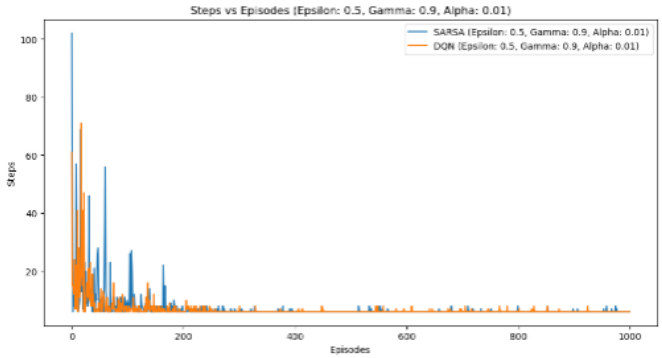


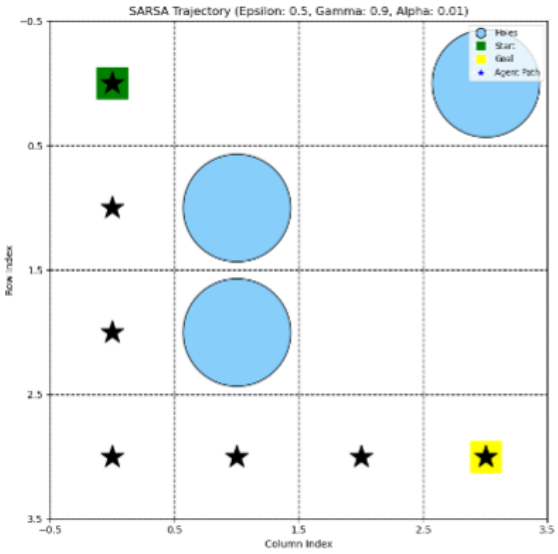


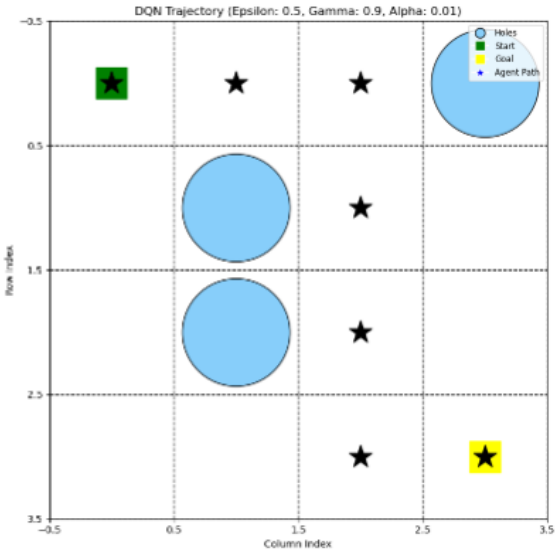
Gamma = 0.9

SARSA converged in 0.0164 seconds, took 6 steps to reach the goal, and had an average reward of 6.92. This indicates effective learning, though slightly lower reward compared to gamma = 0.7.

DQN converged in 0.0163 seconds, also took 6 steps, but achieved a higher average reward of 7.44. This suggests that DQN performed better in terms of reward under this gamma setting, indicating it managed the long-term rewards more effectively.







Effect of Changing Gamma:

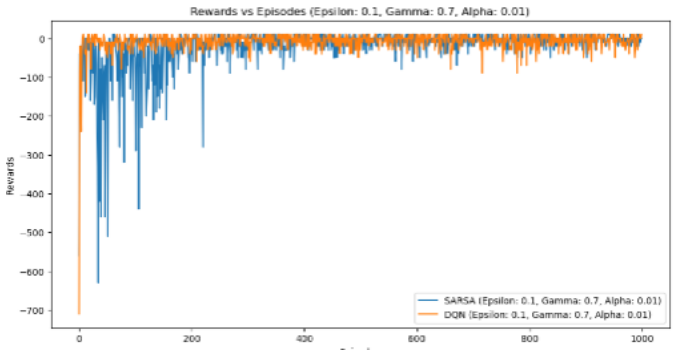
Gamma = 0.7: SARSA performed well with a reward of 7.02, while DQN's reward was much lower at -0.55, even though both algorithms converged quickly.

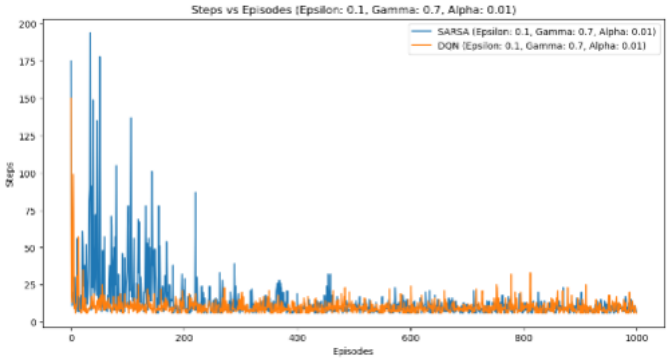
Gamma = 0.9: SARSA's reward dropped slightly to 6.92, but DQN improved significantly with a higher reward of 7.44. This shows that DQN better managed long-term rewards with a higher gamma value.

6.2 Stochastic Environment (Noise =0.4)

Epsilon = 0.1

SARSA: Convergence took 0.0215 seconds with the agent taking 10 steps to reach the goal. The average reward was -20.52, indicating that SARSA struggled significantly under the noisy conditions with this low exploration rate, leading to poor performance and higher penalties. **DQN:** Achieved convergence faster in 0.0186 seconds, with 8 steps to reach the goal and an average reward of -8.33. Although still negative, DQN performed better than SARSA in this noisy environment, suggesting it managed exploration and exploitation more effectively.

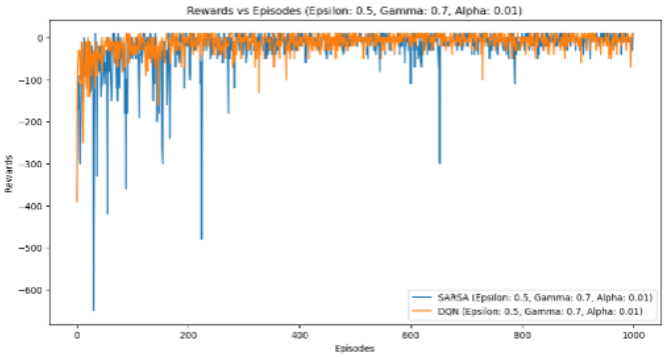


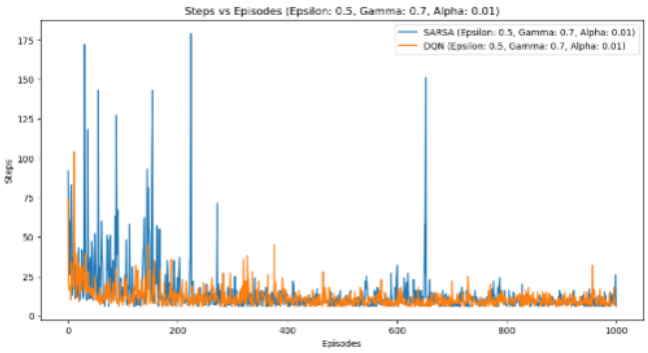


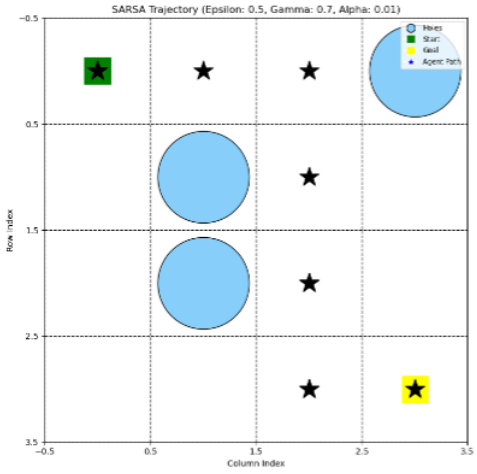
Epsilon = 0.5

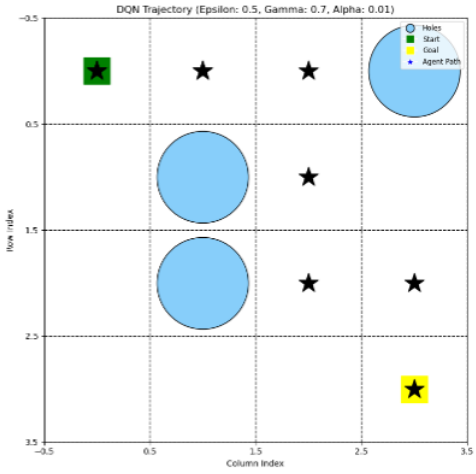
SARSA: Achieved convergence in 0.0196 seconds, required 9 steps, and had an average reward of -20.94. Although convergence time was slightly improved compared to epsilon = 0.1, SARSA's average reward decreased further, indicating that higher exploration still led to suboptimal outcomes.

DQN: Converged in 0.0188 seconds with 8 steps and an average reward of -11.92. DQN managed to improve its reward compared to epsilon = 0.1, showing that a moderate exploration rate helped in navigating the noisy environment more effectively than SARSA.





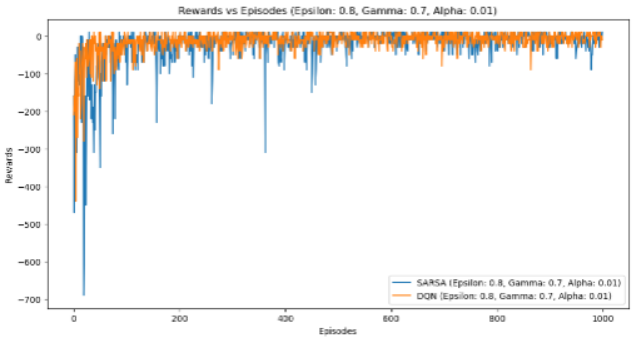


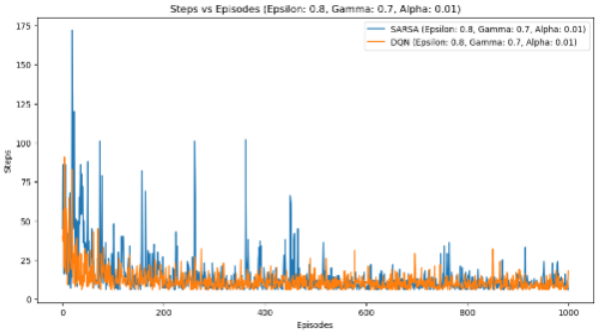


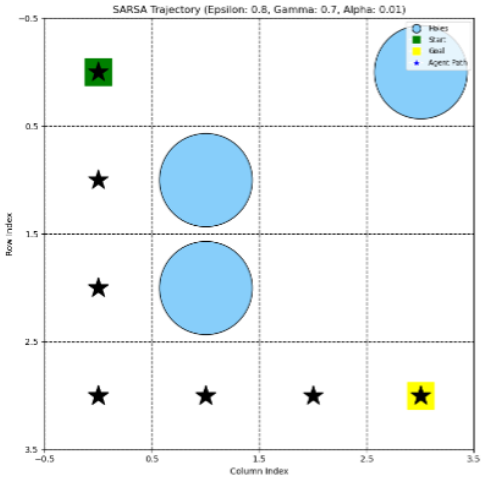
Epsilon = 0.8

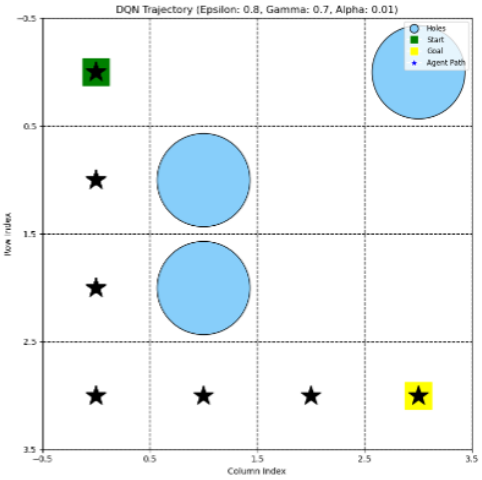
SARSA: Achieved convergence in 0.0217 seconds, with 10 steps and an average reward of -22.42. Increased exploration led to more steps and worsened rewards, indicating that too much exploration resulted in more penalties.

DQN: Took 0.0255 seconds to converge, with 13 steps and an average reward of -13.61. Despite more steps and a longer convergence time, DQN still performed better than SARSA, suggesting that it managed exploration better under high epsilon but with increased penalty.









Comparison Across Environments

In the deterministic environment, lower epsilon values resulted in higher rewards and efficient convergence. SARSA generally achieved better rewards than DQN at lower epsilon values. However, in the stochastic environment, both algorithms struggled with increased noise. DQN consistently outperformed SARSA across different epsilon values by managing exploration more effectively. High epsilon values in the stochastic environment led to more steps and poorer rewards for both algorithms, with DQN still showing a relative advantage over SARSA despite higher penalties.

V. Conclusion

This report compared two reinforcement learning algorithms, SARSA and DQN, in the Frozen Lake environment. Here's a summary of what we found:

In a predictable (deterministic) environment, SARSA generally performed better with lower exploration values (epsilon). It reached the goal more efficiently and earned higher rewards compared to DQN. SARSA was better at avoiding mistakes when exploration was limited.

In a noisy (stochastic) environment, both algorithms had a hard time, but DQN did better. It managed the noise and exploration more effectively than SARSA, leading to better rewards and quicker results.

In summary, SARSA is more effective in predictable situations with low exploration, while DQN handles noisy conditions better, making it more versatile for real and different types of environments.