Contents

Objective

The purpose of this experiment was to compare two reinforcement learning methods:

- **Monte Carlo (MC)**
- **Q-Learning**

We wanted to understand:

- Which method is more stable?
- Which one learns faster?
- Which one has higher variability (variance)?
- How their results differ over time?

In simple terms:

Do Monte Carlo methods produce more unstable (high variance) learning compared to Q-Learning?

Experimental Setup

The experiment was performed in a small **4×4 grid environment**.

Environment Details

- Grid size: 4 × 4
- Step reward: **−1** (penalty for each move)
- One goal (terminal state)
- Discount factor: $\gamma = 0.95$
- Total training episodes: **10,000**

This means the agent is punished for taking longer paths, so it learns to reach the goal in as few steps as possible.

Algorithms Used

- **Monte Carlo (MC)**
  - Updates values only after the episode ends.
  - Uses the **total reward from start to finish**.
  - No learning rate (uses average of returns).

- **Q-Learning**
  - Updates values **after every step**.
  - Uses a learning rate **α = 0.1**
  - Uses exploration rate **ε = 0.2**
  - Updates using an estimated future reward (bootstrapping).

Results

- **Rolling Variance (Window = 200)**

From the rolling variance graph:

- The empirical variance of Monte Carlo returns was 43.33, approximately 7.15 times higher than Q-learning (6.06).
- Q-Learning variance stabilizes quickly around **3 to 5**
- Q-Learning has an early spike, then becomes stable
- Rolling variance was computed over a sliding window of 200 episodes to measure short-term stability.

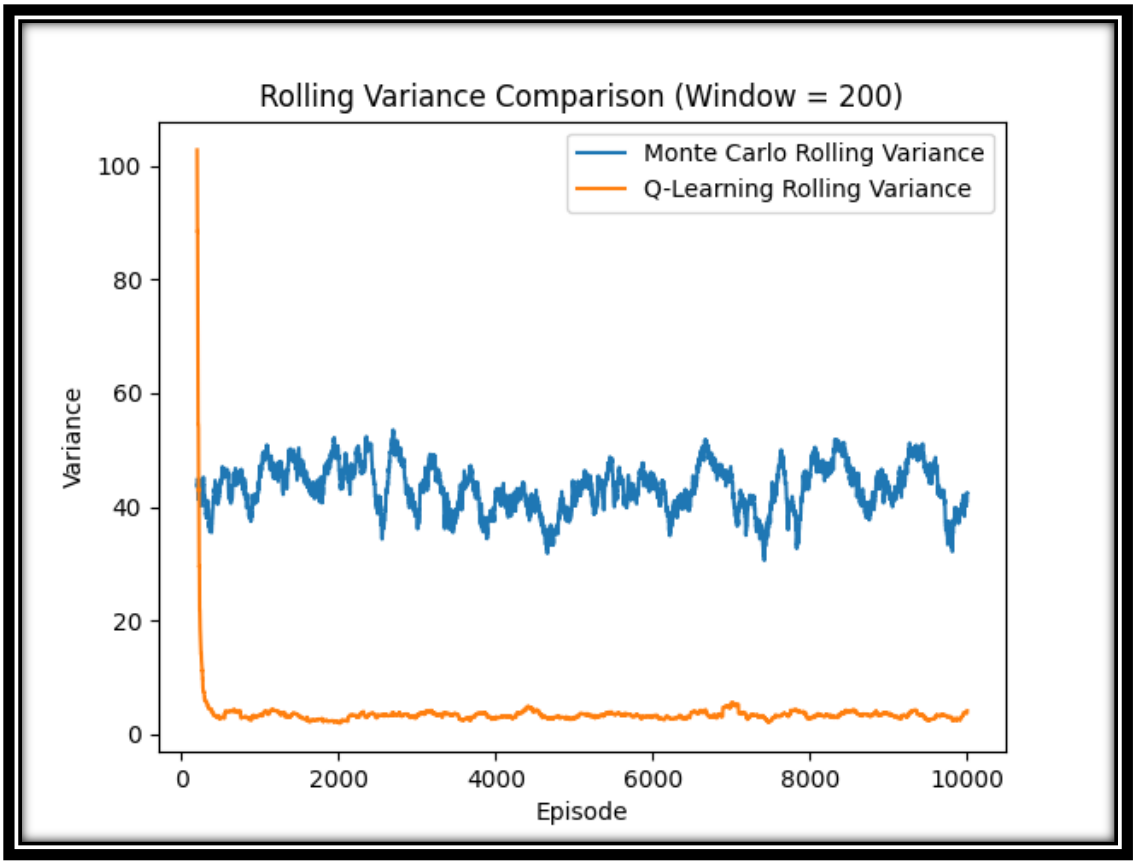

Figure 3 compares rolling variance. Monte Carlo maintains significantly higher variance throughout training.

This shows Monte Carlo is much more unstable over time.

➢ **Episode Return Over Time**

From the episode return plot:

- ❖ Monte Carlo returns jump up and down continuously.
- ❖ Q-Learning becomes stable earlier.
- ❖ Q-Learning returns are more concentrated around similar values.

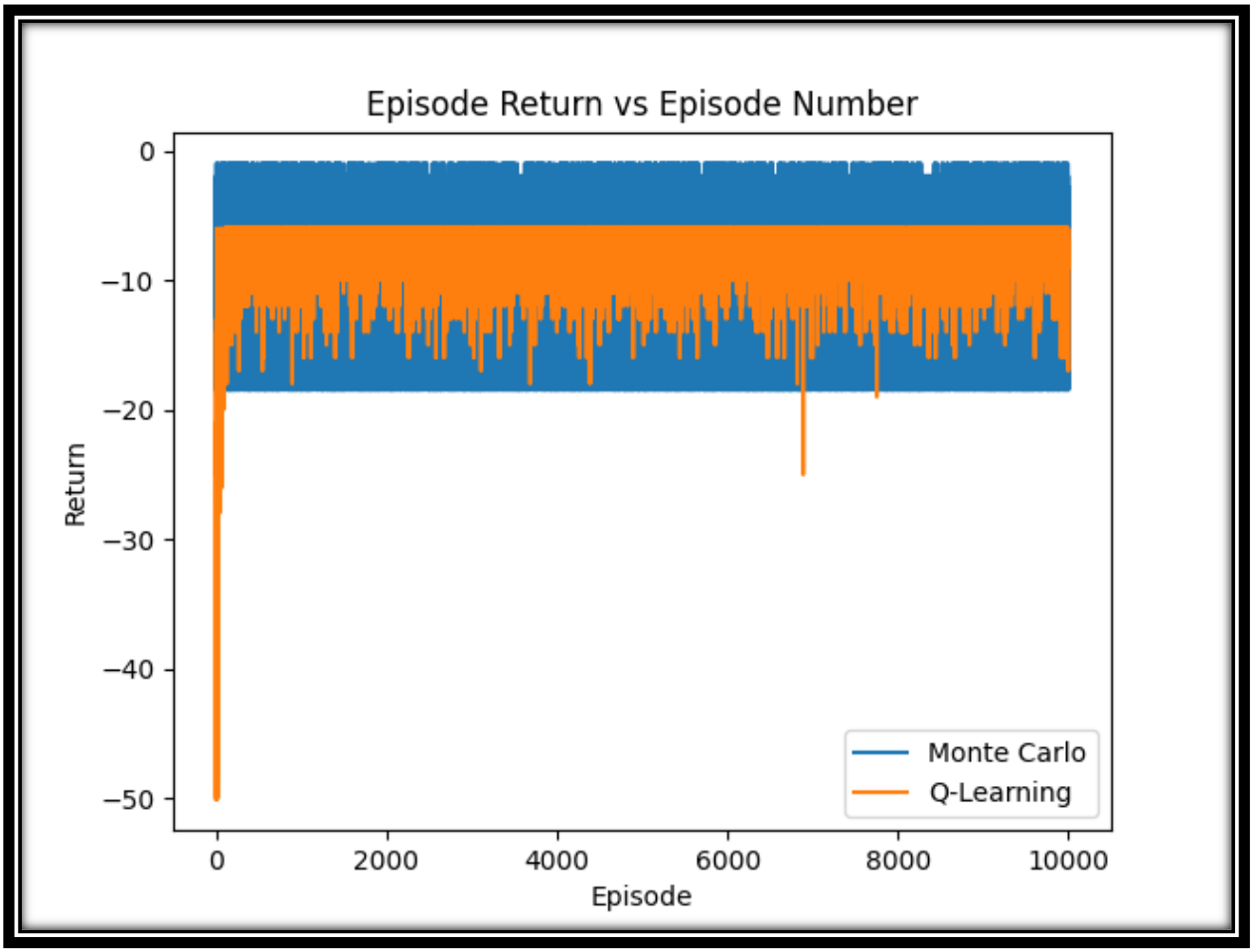

Figure 1 shows the episode returns over 10,000 episodes for both algorithms. Monte Carlo exhibits large oscillations, while Q-learning stabilizes earlier.

➢ **Histogram of Returns**

From the histogram:

- ❖ Monte Carlo returns are spread out (wider distribution).
- ❖ Q-Learning returns are tightly grouped (narrower distribution).

This confirms that Monte Carlo produces more variability.

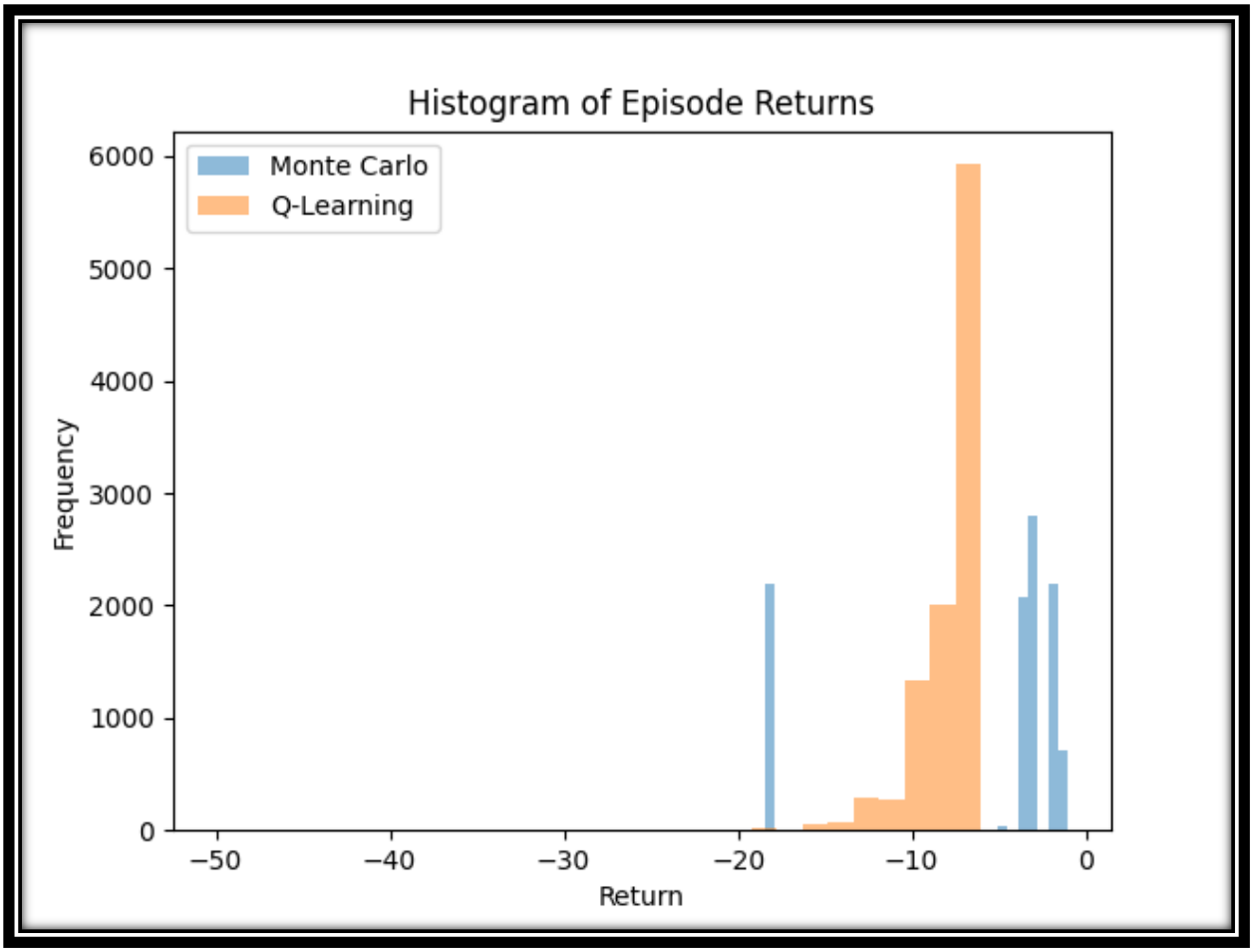

Figure 2 shows the return distribution. Monte Carlo displays a wider distribution, indicating higher variance.

Comparison Summary (Approximate from Graphs)

- **Numerical Summary**

Algorithm	Average Return	Variance Level
Monte Carlo	−6.14	43.33
Q-Learning	−7.53	6.06

Q-Learning not only has lower variance, but also slightly better average performance.

- **Detailed Comparison Table**

Below is a complete comparison in simple terms.

Aspect	Monte Carlo	Q-Learning
Learning Type	Episodic	Step-by-step (Temporal Difference)
Update Timing	Updates only at end of episode	Updates after every step
Uses Learning Rate (α)?	No (uses average of returns)	Yes (α = 0.1 in this experiment)
Exploration Strategy	Exploring Starts	Epsilon-Greedy (ε = 0.2)
Uses Future Estimate?	No	Yes (bootstrapping)
Waits for Episode to Finish?	Yes	No
Speed of Learning	Slower	Faster
Stability	Less stable	More stable
Variance (Empirical)	High (43.33)	Low (6.06)
Return Distribution	Wide	Narrow
Sensitivity to Randomness	High	Lower
Bias	Low bias	Slightly higher bias
Stability in This Experiment	Oscillating	Stable

- **Bias–Variance Tradeoff**

Method	Bias	Variance
Monte Carlo	Low	High
Q-Learning	Slightly higher	Low

Why Do They Behave Differently?

- **Monte Carlo**

  Monte Carlo waits until the episode ends.
  Then it uses the **total accumulated reward**.

  This means:

  - All randomness in the episode affects the update.
  - If one episode is longer than usual, it causes large changes.
  - This creates high variance.

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots$$

  In simple terms:

  Monte Carlo listens to the full story before learning, and if the story changes a lot, the learning changes a lot.

- **Q-Learning**

  Q-Learning updates after every step.

  Instead of waiting for the full return, it uses:

  - The immediate reward
  - An estimate of the best future reward

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

  This reduces noise.

  In simple terms:

  Q-Learning adjusts little by little instead of making big changes at the end.

Conclusion

From this experiment, we clearly observe:

- Monte Carlo produces **higher variance**
- Q-Learning stabilizes **much faster**
- Q-Learning has **more consistent returns**
- Monte Carlo continues oscillating even after many episodes

Therefore:

Q-Learning is more stable and practical for this GridWorld problem, while Monte Carlo shows higher variability due to relying on full episodic returns.