Experiment Report

Yuan Yibo 3220241645 yuanyibo7100@gmail.com BIT

Code Availability：
The source code for this project is available in the src directory of this repository. You can also access it directly via the following GitHub link: https://github.com/yuanyibo666/RL-Assignment

1. Experiment 1

1.1 Requirements

You need to plot the mean of cumula9ve return across the number of runs for SARSA, Expected SARSA and Q-learning for the frozen lake environment, with discount factor = 0.99 in the same figure. Consider the following common parameters for the learning phase for both the algorithms.
- Number of 9mesteps per episode=400
- Number of episodes = 6000
- Number of independent runs=10

Observe the variance of the cumula9ve return across the number of runs for SARSA , Expected SARSA and Q-learning and plot the return on the same figure.

1.2 Experimental Results and Analyses

I implemented the two algorithms SARSA and Q-learning on Frozen lake domain respectively as per the experimental requirements and the results are shown in Fig1.

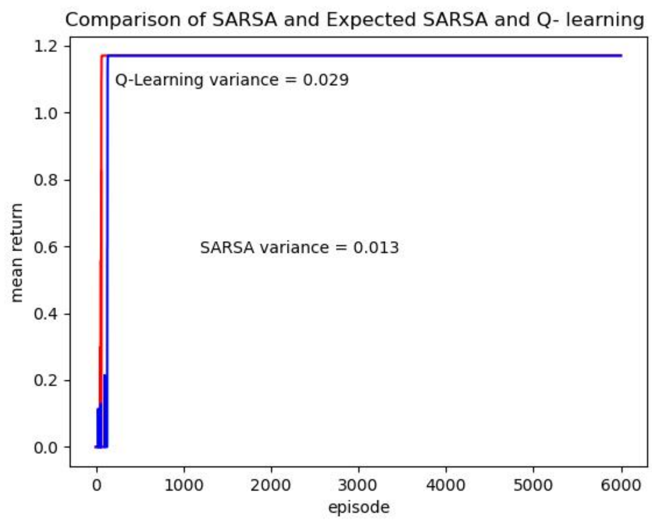

Fig1

In Figure 1, the blue dash represents the SARSA algorithm, while the red dash represents the Q-Learning algorithm. Despite being trained for 6000 episodes, what we can see is that both algorithms converge quickly within about 200 episodes. By printing the results, we are confident that both algorithms find a reasonable route that avoids all obstacles.

At the same time, we can also see that SARSA converges a little faster compared to Q-Learning, which is directly reflected in the fact that the variance of the returns of SARSA (0.013) is smaller than the variance of the returns of Q-Learning (0.029) .

For the above results, we make the following analyses:

1. First of all, SARSA is an on-policy algorithm that uses the same policy as the current learning process to select actions. This means that SARSA uses the current policy directly during the learning process, so its updates reflect the performance of the current policy. Q-Learning is an off-policy algorithm, which means that Q-Learning uses a hypothetical optimal policy during the learning process, which may lead it to be more aggressive between exploration and exploitation.

2. Secondly, in the Ice Lake problem, exploring inappropriate actions may lead to immediate failure of the agent.SARSA, due to the nature of its off-policy, may be more inclined to explore actions that perform better under the current policy, which helps it to find a better balance between exploration and exploitation. Q-Learning may more frequently explore actions that currently seem to have the highest value, even though they may not be the best choice under the current strategy.

3. The Q value updates used by SARSA in the learning process are based on the actual actions taken, which makes the learning process more stable because it is not subject to the extreme values that may be introduced by a greedy strategy. Q-Learning uses maximum Q-value updates, which can lead to a more unstable learning process, especially in the face of high variance rewards

2. Experiment 2

2.1 Requirements

In this question you will implement a Deep Q- Network (DQN) and Deep SARSA using a library of your choice (for e.g., pytorch, tensorflow, keras, chainer, etc). You will train a Cartpole environment from open ai gym using both the algorithms.

- Plot the mean cumula9ve return with discount factor 0.95, across 3 independent runs for both the algorithms.
- It is a possibility that Deep SARSA might not be able to learn a good policy if you observe something similar write a short descrip9on on why you think deep SARSA fails in this case.

2.2 Experimental Results and Analyses

I implemented and ran the Deep SARSA and DQN algorithms separately, following the parameter settings in the experimental requirements. Here, I used pytorch version 2.3.0+cu121 and used RTX3060 for training.

The results of the three independent runs of the DQN are shown in Figure 2,3,4.

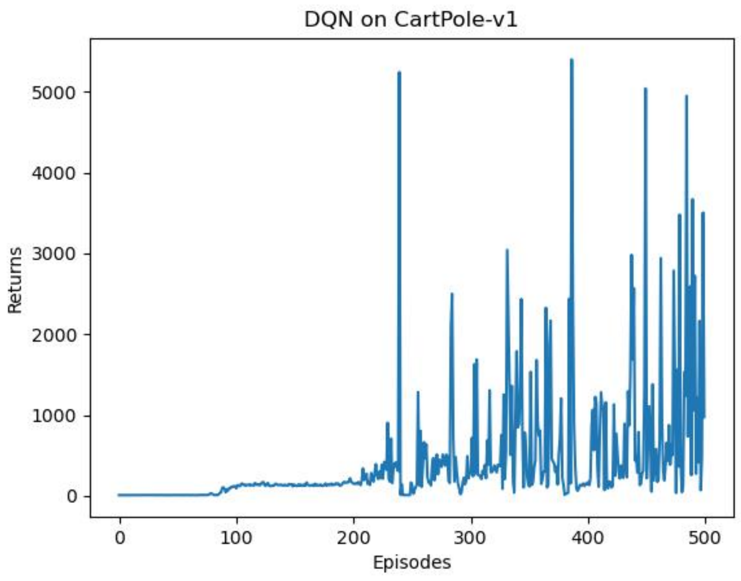

Figure 2

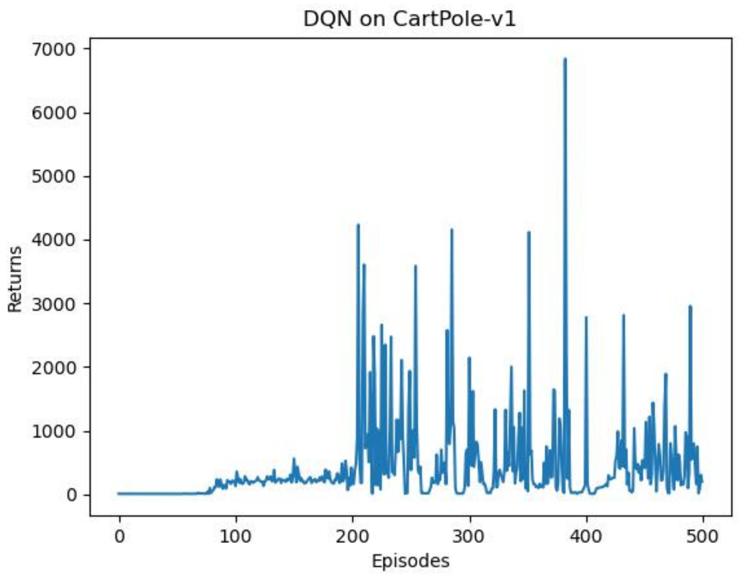

Figure 3

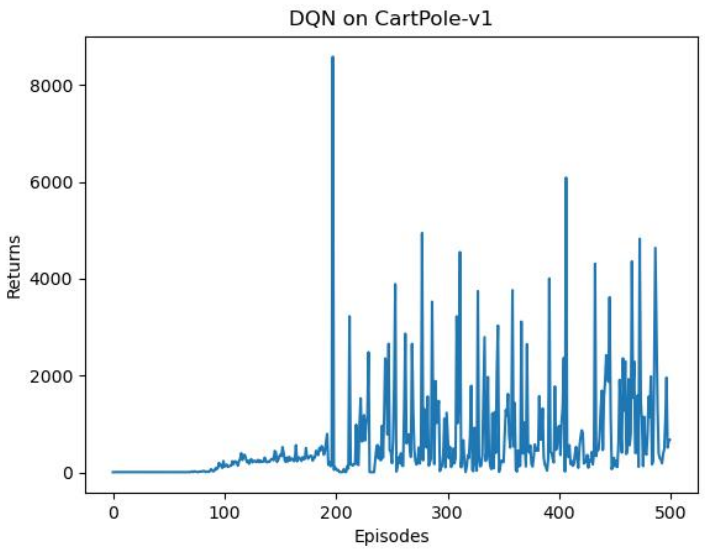

Figure 4

The results of the three independent runs of the DQN are shown in Figure 5,6,7.

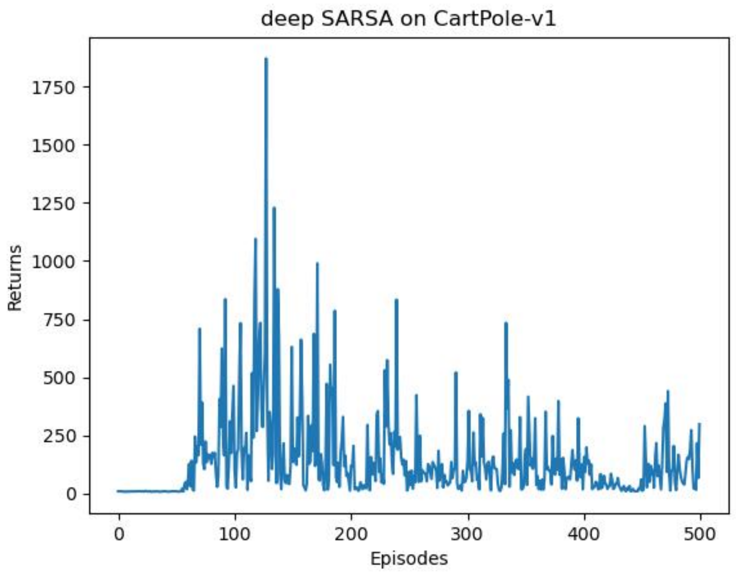

Figure 5

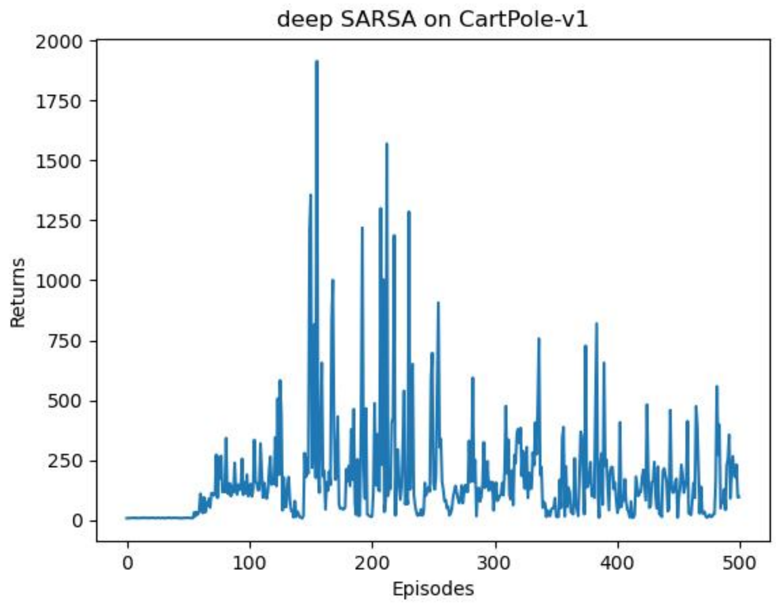

Figure 6

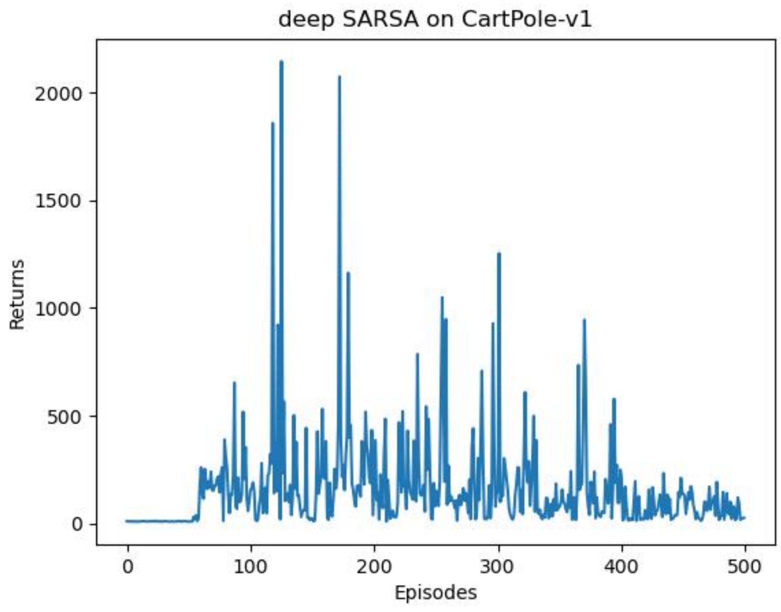

Figure 7

Through the above experimental results we can find that the highest return in a single episodes of DQN algorithm reaches more than 8000, while the highest return of Deep SARSA algorithm is only about 2000. Meanwhile, unlike the results demonstrated by DQN, Deep SARSA reaches the maximum RETURN in the pre-training period and oscillates lower in the next several episodes. The above experimental results show that Deep SARSA is not as effective as DQN in Cartpole environment, both in terms of the model's convergence ability and the performance of a single episode.

The different performances of the two algorithms are analysed as follows.

1. First of all, DQN utilises the experience replay mechanism, which allows it to learn from past experiences rather than relying solely on the most recent experience. This mechanism increases the efficiency of the data and helps to break the temporal correlation between samples, thus improving learning.

2. One of the design goals of DQN is to ensure convergence, even in the face of non-smooth data distributions. Deep SARSA, on the other hand, as an online policy algorithm, may be more susceptible to changes in data distribution, leading to convergence problems.

3. Both DQN and Deep SARSA are sensitive to hyper-parameters such as learning rate, discount factor, exploration rate, etc. DQN may have more flexibility in hyper-parameter tuning as its design allows more room for tuning.

4. DQN reduces the variance in the learning process by using two networks (one for generating Q-values and the other as a target network), which helps to improve learning stability. Deep SARSA does not usually use this two-network structure and therefore may be more susceptible to changes in data distribution, leading to an unstable learning process.