UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

Deep Q-Network (DQN) & Double DQN (DDQN)

Instructor: Luong Ngoc Hoang

Student: Ha Huy Hoang - 22520460

Class: CS211.P11

Table of Contents

1. Introduction **1**

2. Statistics **1**
2.1. CartPole-v0 2
2.2. BreakoutNoFrameskip-v4 3

3. Conclusion **4**

1. Introduction

DQN is a Deep Reinforcement Learning algorithm that combines Q-Learning with deep neural networks. It was developed to address reinforcement learning problems with large state spaces, where storing and querying a Q-Table is impractical. DQN uses a neural network to approximate the Q-function, predicting Q-values for each action at a given state. Key techniques in DQN include Experience Replay and Target Network updates to stabilize training.

This is how Q-values update in DQN:

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

```
with torch.no_grad():
    target_max, _ = target_network(data.next_observations).max(dim=1)
    td_target = data.rewards.flatten() + args.gamma * target_max *
                                        (1 - data.dones.flatten())
old_val = q_network(data.observations).gather(1, data.actions).squeeze()
```

Double DQN is an enhancement of DQN aimed at addressing the issue of overestimation bias in Q-value predictions. Instead of using a single network to both select the best action and estimate its Q-value, Double DQN separates these processes. One network selects the best action (based on current Q-values), while another network is used to estimate the Q-value corresponding to that action. This separation can reduces estimation bias, leading to improved performance.

Q-values will be updated by Q_{online}(choose actions) and Q_{target}(estimate the Q-value) in DQN:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q_{\text{target}}(s', \arg\max_{a'} Q_{\text{online}}(s',a')) - Q(s,a) \right]$$

```
with torch.no_grad():
    next_actions = q_network(data.next_observations).argmax(dim=1)
    target_q_values = target_network(data.next_observations)
    target_max = target_q_values.gather(1, next_actions.unsqueeze(1)).squeeze()
    td_target = data.rewards.flatten() + args.gamma * target_max *
                                        (1 - data.dones.flatten())
old_val = q_network(data.observations).gather(1, data.actions).squeeze()
```

Key differences between DQN and Double DQN:

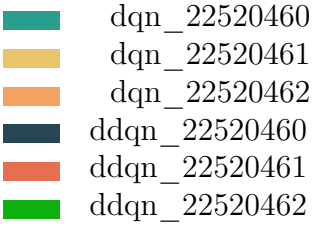
- **DQN**: Uses a single network for both selecting the best action and estimating its Q-value.
- **Double DQN:** Separates the action selection and Q-value estimation processes, either with two networks or by assigning distinct roles within the same network.

2. Statistics

As the teacher has guided, I will use TensorBoard to compare different runs. However, there is an issue with large fluctuations between steps, making the graphs difficult to read and analyze. Therefore, I have adjusted the smoothing to 0.99 to make the analysis easier.

Due to the redundancy of plotting SPS (as experiments have shown it does not affect the model or other parameters), I will not present or display any charts related to SPS.

Each color lines presents a algorithm with seed that have the structer likes algorithm_seed, the seeds and algorithm i used for experiment is listed below:



2.1. CartPole-v0

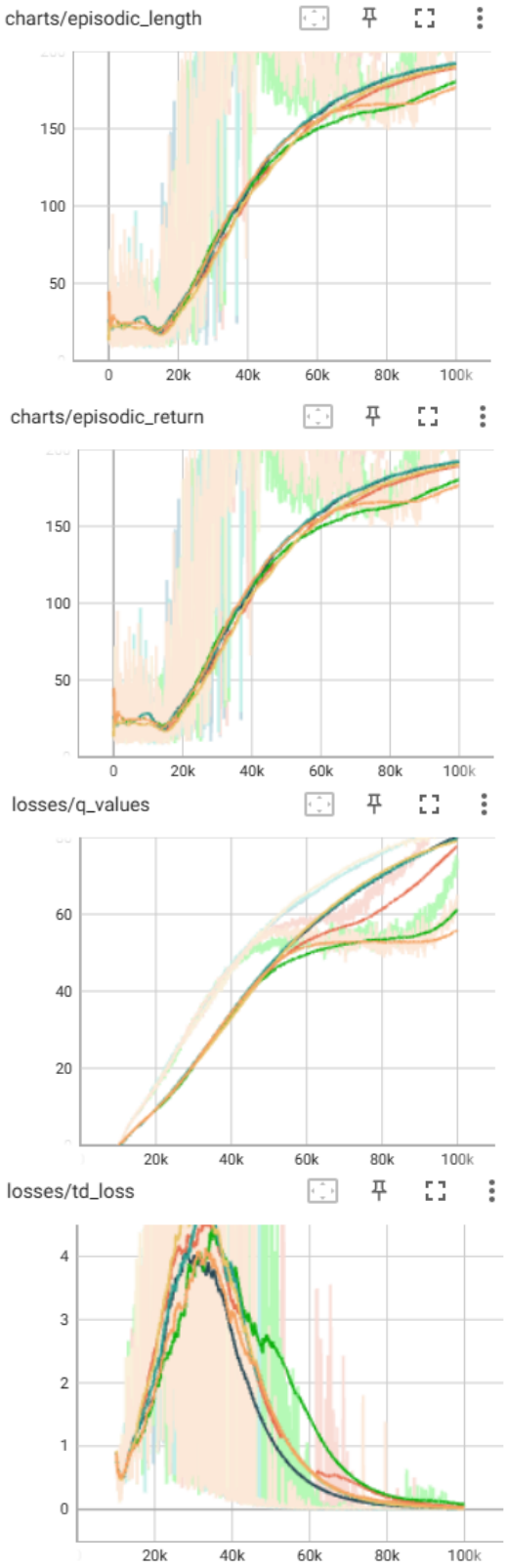


Figure 1. CartPole-v0

In CartPole-v0, both DQN and Double DQN achieve nearly the same performance across all seeds, with the best episodes lasting 4 seconds because, in version v0 of CartPole, the episodes are truncated when the episode length exceeds 200[1].

In the two plots of **episodic_length** and **episodic_return**, for each seed, the algorithms exhibit nearly overlapping lines and consistently achieve the maximum result of 200 in this environment. The **td_loss** in all experiments converges to 0. However, in the **q_values** plot, at seed 22520461, we observe that the **q_values** of DQN are higher than those of Double DQN. This could indicate that DQN suffers from overestimation bias. However, this phenomenon is not observed with other seeds, and eventually, the **q_values** of both algorithms converge for this seed as well. Ultimately, this environment cannot effectively assess the overestimation bias issue, as CartPole-v0 is considered relatively simple, with a small state and action space. In more complex environments, the differences between DQN and Double DQN regarding overestimation bias may become more apparent, such as in BreakoutNoFrameskip-v4.

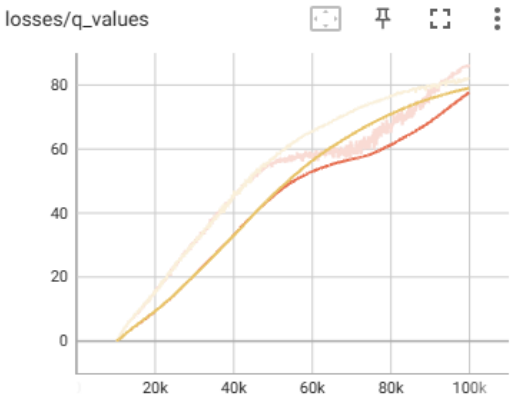


Figure 2. CartPole-v0 with seed=22520461

[1]: CartPole-Gym Documentation

2.2. BreakoutNoFrameskip-v4

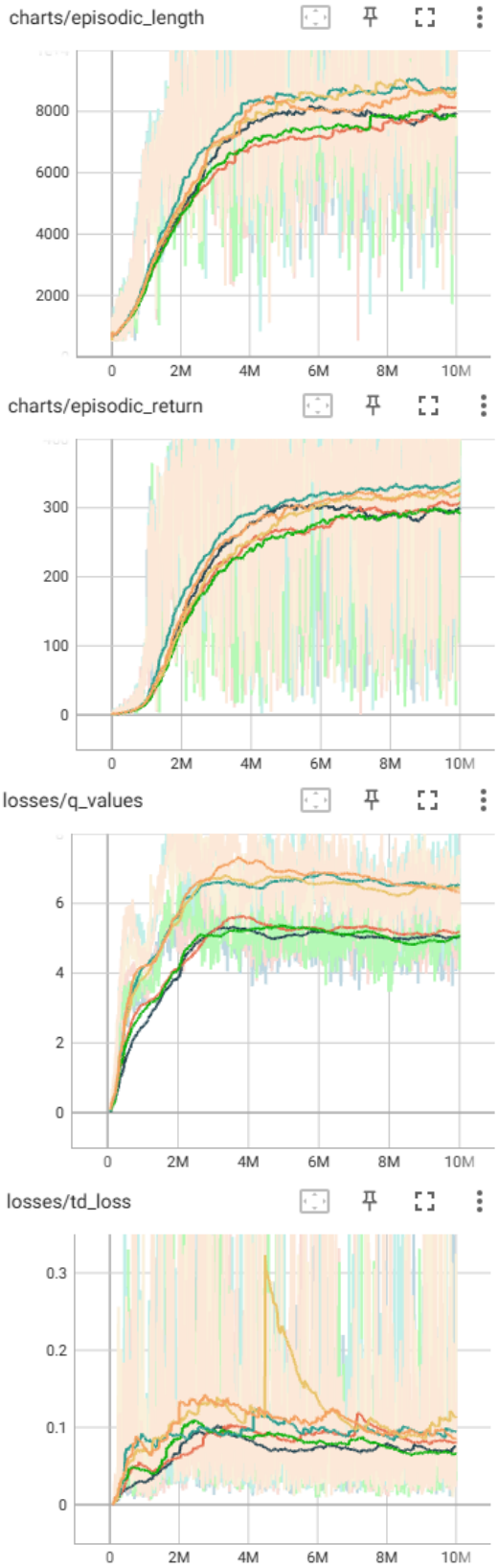


Figure 3. BreakoutNoFrameskip-v4

In the BreakoutNoFrameskip-v4 environment, the differences between DQN and Double DQN become more evident. In all three plots **episodic_length**, **episodic_return**, and **q_values**, Double DQN consistently shows lower curves compared to DQN. The fact that Double DQN has more stable and lower **q_values** demonstrates that DQN suffers from overestimation bias in this environment. This is further supported by the observation that the **td_loss** of DQN is always higher than that of Double DQN, particularly at seed 22520461, where the **td_loss** is significantly higher in the **global_steps** range of 4M to 6M compared to other curves. The output videos also illustrate the stability of Double DQN and its superior performance over DQN.

As mentioned in the introduction, the difference between DQN and Double DQN lies in the fact that Double DQN is effective at reducing the overestimation of Q-values and improving the stability of learning. This is achieved by separating the processes instead of using a single network to both select the best action and estimate its Q-value. One network selects the best action (based on current Q-values), while another network is used to estimate the Q-value corresponding to that action.

Algo_Seed	Max Score
ddqn_22520460	420
ddqn_22520461	411
ddqn_22520462	420
dqn_22520460	411
dqn_22520461	391
dqn_22520462	394

Table 1. Best Score in BreakoutNoFrameskip-v4

3. Conclusion

All the algorithms and seed i runned with this tempalte command:

```
python cleanrl/algo.py --seed seed --env-id game --total-timesteps num_steps --capture_video
```

- If game is 'CartPole-v0' the nums_steps will be 100000.
- If game is 'BreakoutNoFrameskip-v4' the nums_steps will be 10000000.

The best videos of each algorithm in environments is stored in this link.

Additionally, other studies have also found that DDQN can lead to improved performance on tasks such as playing the card game Hanabi, controlling a simulated robot arm, and navigating a simulated 3D environment. In general, DDQN has been found to be particularly effective at reducing the overestimation of the Q values and improving the stability of learning.

However, Double DQN is not a Panacea and may not always lead to improved performance. In fact, some studies have found that DDQN can perform worse than traditional DQN on certain tasks, such as playing the game of Go. Therefore, it is important to carefully evaluate the performance of DDQN on each specific task to determine whether it is likely to be beneficial. A comprehensive list of results performed by the authors of the DDQN paper is available on page 10 of the paper.

In summary, DDQN has been shown to improve performance on many reinforcement learning tasks, but it is not always the best choice and its effectiveness can vary depending on the specific task[2].

[2]: Double Deep Q-Networks (DDQN) - A Quick Intro (with Code)