**Homework 4:**

**Reinforcement Learning**

**Part I. Implementation (-5 if not explain in detail):**

* **Please screenshot your code snippets of Part 1 ~ Part 3, and explain your implementation**
* **Part 1, taxi**

**一張含有 文字, 螢幕擷取畫面, 功能表 的圖片

自動產生的描述**

* **Part 2, cartpole**

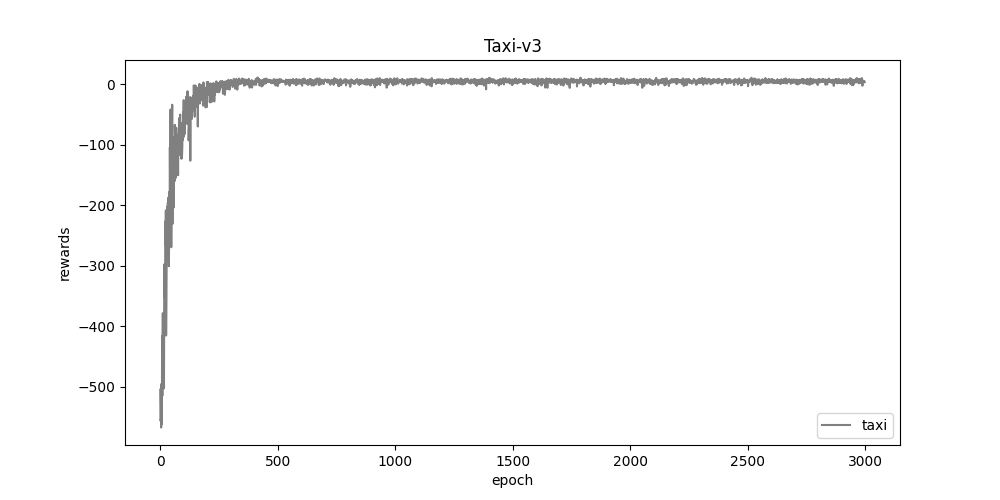
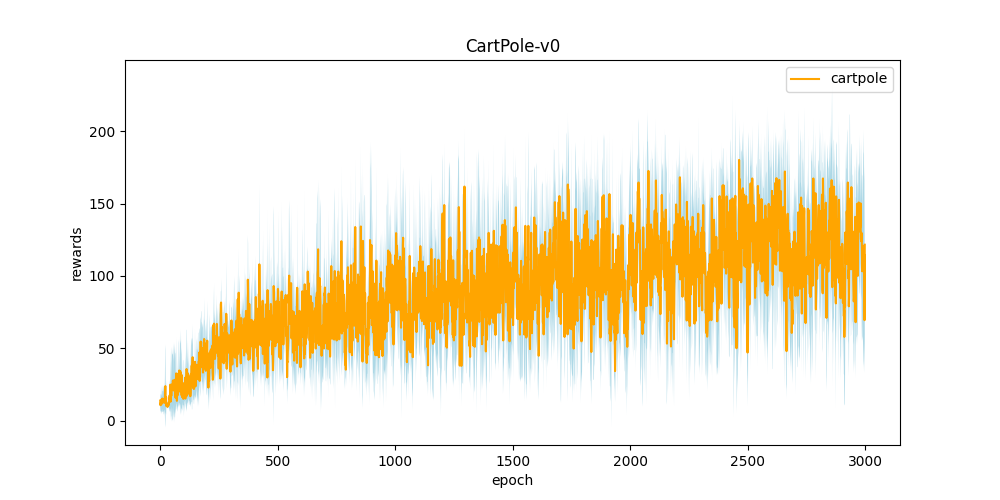
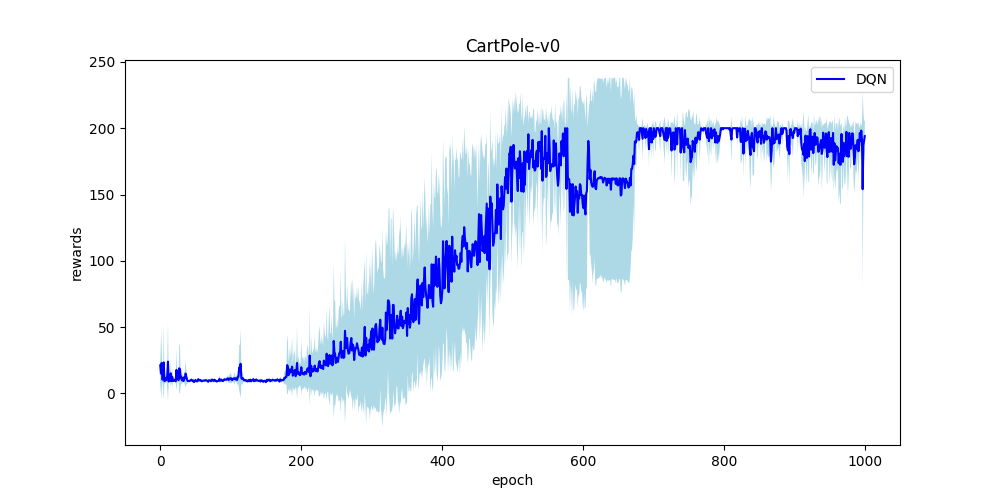
**一張含有 文字, 螢幕擷取畫面, 功能表 的圖片

自動產生的描述**

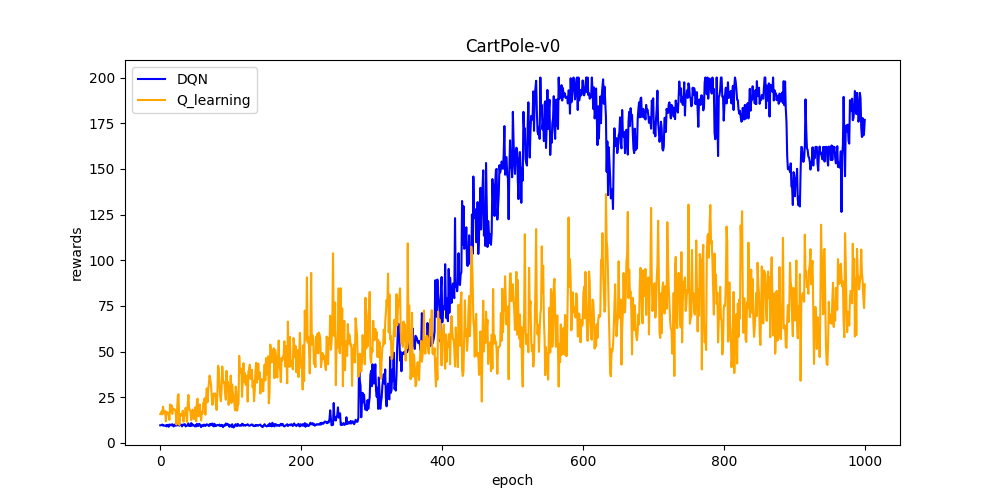
* **Part 3, DQN**

**Part II. Experiment Results:**

* **Please paste taxi.png, cartpole.png, DQN.png and compare.png here.**

1. **taxi.png:**
2. **cartpole.png**
3. **DQN.png**

**4. compare.png**

****

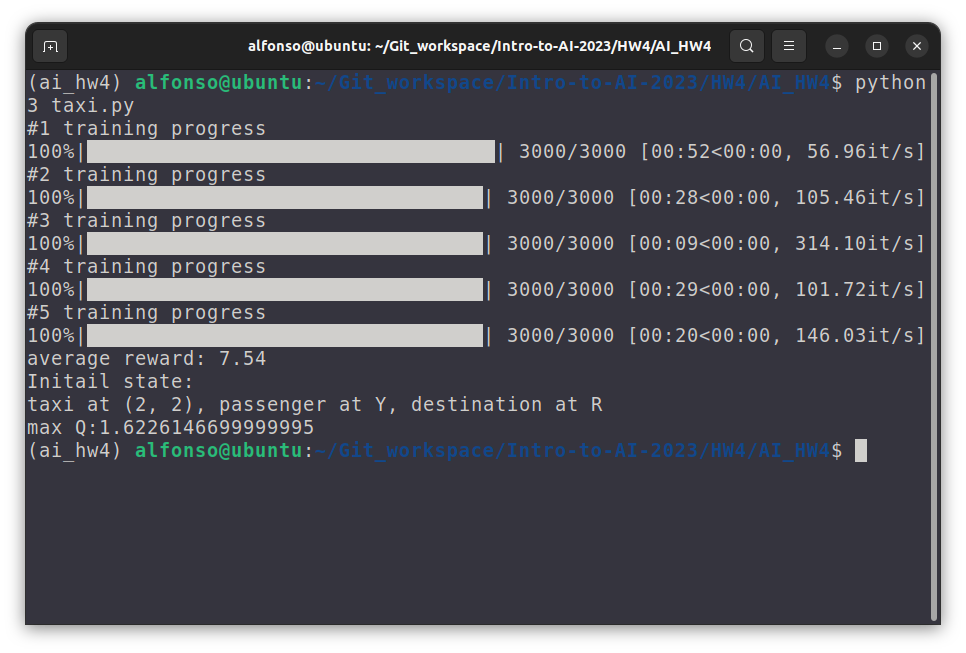
**Part III. Question Answering (50%):**

1. Calculate the optimal Q-value of a given state in Taxi-v3, and compare with the Q-value you learned (Please screenshot the result of the “check\_max\_Q” function to show the Q-value you learned). **(10%)**

一張含有 字型, 文字, 白色, 筆跡 的圖片

自動產生的描述

,approximately equals to the max Q we get.



1. Calculate the max Q-value of the initial state in CartPole-v0, and compare with the Q-value you learned. (Please screenshot the result of the “check\_max\_Q” function to show the Q-value you learned) **(10%)**

一張含有 文字, 電子產品, 螢幕擷取畫面, 陳列 的圖片

自動產生的描述

**一張含有 文字, 電子產品, 螢幕擷取畫面, 陳列 的圖片

自動產生的描述**

1. Why do we need to discretize the observation in Part 2? **(3%)**

A: Discretize the observation would make the algorithm more easier to converge. In this case, the dimension of the state space is very high, and using continuous observations would make the algorithm difficult to train.

1. How do you expect the performance will be if we increase “num\_bins”? **(3%)**

A: Better, because by increasing the number of bins, we can increase the granularity of the discretization and make the state space more fine-grained. This can make the policy more precise and enable better policies to be learned.

1. Is there any concern if we increase “num\_bins”? **(3%)**

A: Increasing the number of bins will increase the dimension of the state space, which will make training time longer and require more memory to store the Q-table.

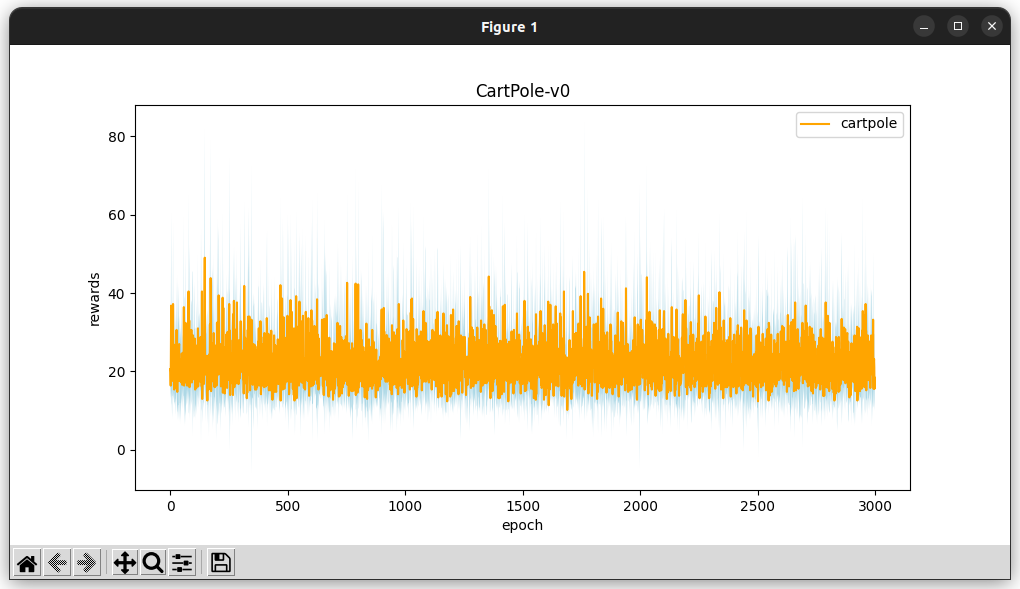
1. Which model (DQN, discretized Q learning) performs better in Cartpole-v0, and what are the reasons? **(5%)**

A: DQN, since CartPole-v0 has a continuous state space, which makes it challenging to create a discrete Q-table for Q-learning. Discretizing the state space can lead to a loss of information and reduced performance. On the other hand, DQN uses a neural network to approximate the Q-values, which can handle continuous state spaces more effectively.

1. What is the purpose of using the epsilon greedy algorithm while choosing an action? **(3%)**

The main purpose of the epsilon greedy algorithm is to choose between exploration and exploitation when the agent has none or limited knowledge about the environment.

1. What will happen, if we don’t use the epsilon greedy algorithm in the CartPole-v0 environment? **(3%)**



The result may looks like the plot above, the agent is not going to explore the unknown enviroment.

1. Is it possible to achieve the same performance without the epsilon greedy algorithm in the CartPole-v0 environment? Why or Why not? **(3%)**

Yes, softmax may reach same performance.

1. Why don’t we need the epsilon greedy algorithm during the testing section? **(3%)**

Because the agent has already known the enviroment.

1. Why does “with torch.no\_grad():“ do inside the “choose\_action” function in DQN? **(4%)**

A: It is used to disable gradient computation during the evaluation of the Q-values of the current state. During the testing phase, we only want to evaluate the learned policy and not update the weights of the network. Disabling gradient computation using "with torch.no\_grad():" saves memory and computation time by preventing the calculation of gradients.