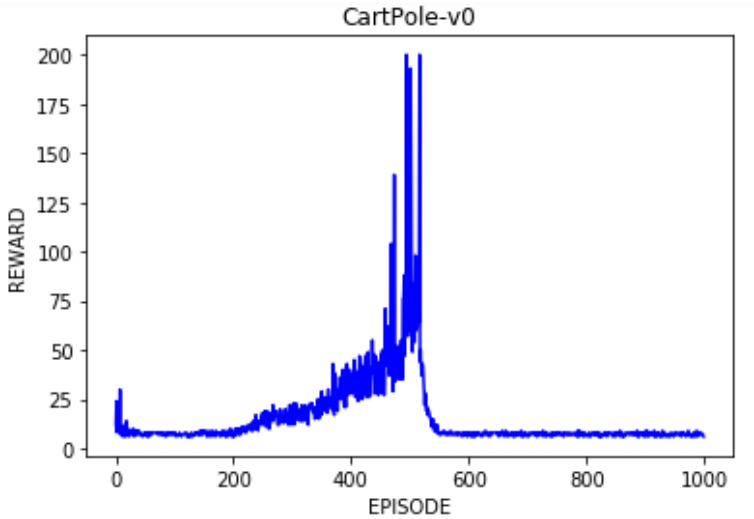
Lab6: Deep Q-Network and Deep Deterministic Policy Gradient

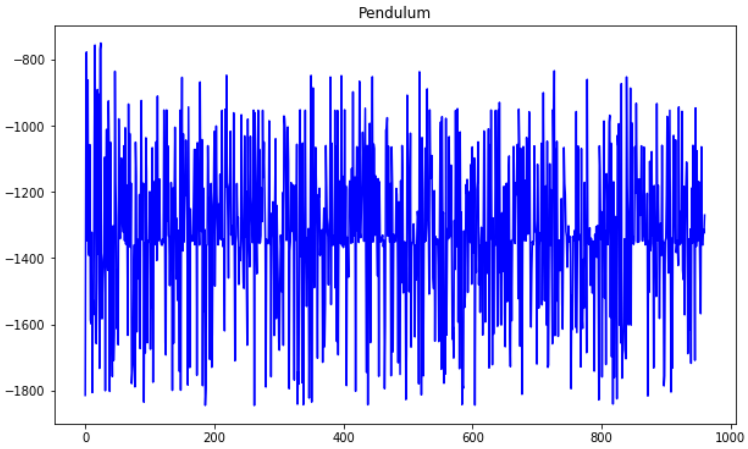
學號：0756616
系所：多媒體工程學系
姓名：周冠伶

1. Report

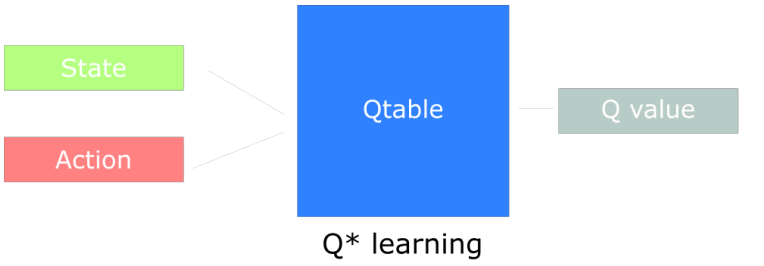
- A plot shows episode rewards of at least 1000 training episodes in CartPole-v1



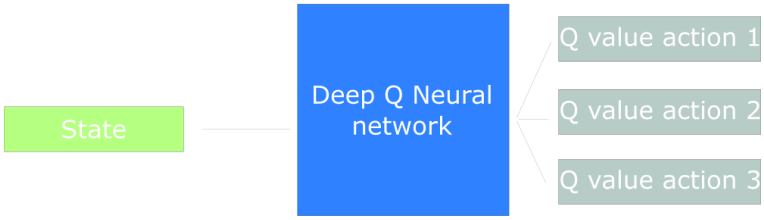
- A plot shows episode rewards of at least 1000 training episodes in Pendulum-v0



- Describe your major implementation of both algorithms in detail.



Q* learning



Deep Q* learning

DQN：由兩個相同的 Network 組成，分別對應到目標網路與行為網路。其中行為網路會每次都更新參數，但是目標網路不會每次都進行參數更新。

```
class Network(torch.nn.Module):
    def __init__(self, state, action):
        super(Network, self).__init__()
        self.fc1 = torch.nn.Linear(state, 32)
        self.fc1.weight.data.normal_(0, 0.1)
        self.output = torch.nn.Linear(32, action)
        self.output.weight.data.normal_(0, 0.1)

    def forward(self, x):
        x = self.fc1(x)
        x = torch.nn.functional.relu(x)
        # action value
        x = self.output(x)
        return x
```

DDPG：由 Actor 與 Critic 組成，分別有對應的網路和目標。兩者在最後輸出層使用的激勵函式不同；輸入的部分，Actor 只輸入狀態、Critic 還會輸入 Action；輸出的部分，Actor 輸出 Action、Critic 輸出 Q Value。目標的更新依據評論家對下一個狀態和動作計算出的 Q Value。

```python
class Actor(torch.nn.Module):
    def __init__(self, state, action):
        super(Actor, self).__init__()
        self.linear1 = torch.nn.Linear(state, 400)
        self.linear2 = torch.nn.Linear(400, 300)
        self.linear3 = torch.nn.Linear(300, action)

    def forward(self, x):
        x = self.linear1(x)
        x = torch.nn.functional.relu(x)
        x = self.linear2(x)
        x = torch.nn.functional.relu(x)
        x = self.linear3(x)
        x = torch.nn.functional.tanh(x)

        return x
```

```python
class Critic(torch.nn.Module):
    def __init__(self, state, action):
        super(Critic, self).__init__()
        self.linear1 = torch.nn.Linear(state, 400)
        self.linear2 = torch.nn.Linear(400 + action, 300)
        self.linear3 = torch.nn.Linear(300, 1)
        self.linear4 = torch.nn.Linear(1, 1)

    def forward(self, x, act):
        # front
        x = self.linear1(x)
        x = torch.nn.functional.relu(x)

        x = torch.cat([x, act.type_as(x)], 1)

        # end
        x = self.linear2(x)
        x = torch.nn.functional.relu(x)
        x = self.linear3(x)
        x = torch.nn.functional.relu(x)
        x = self.linear4(x)
        x = torch.nn.functional.relu(x)
        return x
```

- Describe differences between your implementation and algorithms.

  每次開始進行遊戲前，都會進行初始化。每次選定動作後，就會進行遊戲並計算出該動作的得分；只要還沒有結束遊戲，就會把得分進行累加、並進行學習。

- Describe your implementation and the gradient of actor updating.

  初始化完畢後，使用目前狀態的動作(Forward)後，會預測出一組 Q Valuea 以及可以計算出一組損失數值；(Back-Propagation)反向進行繼計算後，使用 Soft 進行更新。

- Describe your implementation and the gradient of critic updating.

  初始化完畢後，會先隨機採樣，使用 Actor 的目標網路計算次個狀態的動作，並用損失函式計算評論家與目標評論的損失，並依此進行更新。

- Explain effects of the discount factor.

對於當前越遙遠的未來，對其之影響力越小。

- Explain benefits of epsilon-greedy in comparison to greedy action selection.

  Epsilon Greedy 主要用於探索前期，剛開始探索時，隨機探索的效果會比根據政策還要優，因為此時可以快速獲得經驗累積。但是到了中、後期，隨機探索的效果就會變得很差，反而需要依賴過去的經驗來探索較優。Epsilon 就是用於控制 Greedy 程度的數值，通常隨著時間會越來越低。

- Explain the necessity of the target network.

  實際在進行網路的訓練時，我們會有評估網路與目標網路，評估網路會在每一次訓練結束時進行更新、目標網路則否(定量次數後更新一次，參數源自於評估網路)。

  由於在 Q-Learning 的架構中，目前狀態與下一個狀態市有遞迴關係的，如果沒有目標網路，在每次訓練完後就進行更新，會導致目標也跟著變動，完全無法收斂。

- Explain the effect of replay buffer size in case of too large or too small.

  過去的經驗可以不斷重複使用於訓練網路，這對於網路本身的穩定與進步有所幫助，其中我們可以調整一次所要使用的批量。抽取出定量的數據後，我們會採取隨機抽樣方式進行更新。

  緩衝區大小類似於汲取過去經驗來進行更新，當使用過多參考，可能會導致無法進步(可能會收斂於過去的區域解，無法突破)、而使用過少則可能會變得沒有參考一樣(比較有突破過去的可能性，但有可能無法收斂或表現更差)。

2. Report Bonus

- Explain the choice of the random process rather than normal distribution.

  選擇常態分佈的話，幾乎可以保證可以獲得一定的收斂解，但是通常會陷入瓶頸無法突破；如果使用隨機過程，通常需要非常大量的測試才能找到收斂解，雖然耗費時間較多、但是有機會獲得更優的解。

3. Performance

- [CartPole-v1] Average reward of 10 testing episodes: Average ÷ 5

```
----------TESTING----------
[ 0 ] 9.0
[ 1 ] 10.0
[ 2 ] 9.0
[ 3 ] 11.0
[ 4 ] 10.0
[ 5 ] 13.0
[ 6 ] 10.0
[ 7 ] 10.0
[ 8 ] 10.0
[ 9 ] 9.0
--------------RESULT--------------
MEANS 2.02
```

- [Pendulum-v0] Average reward of 10 testing episodes: (Average + 700) ÷ 5 來不及 Train 1000 Epoch，所以沒有 Test。