# Data Science HW4 Report

這次作業是使用助教放在ilms的cartpole來進行修改，其中implement的algorithm為DQN，主要修改的地方為network及reward的部分，在下面做說明。

1. 在這次LunerLander的作業中我增加了兩層hidden layer，使模型複雜一些，比較容易train成功。

```
class network(nn.Module):

    def __init__(self , num_state , num_action):

        super().__init__()
        self.fc1 = nn.Linear(num_state , 50)
        self.fc2 = nn.Linear(50,64)
        self.fc3 = nn.Linear(64,64)
        self.out = nn.Linear(64 , num_action)

    def forward(self , x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.out(x)
        return x
```

2. Reward shaping的部分我嘗試了以下幾種，

- Success1:利用x座標、y座標、y方向速度來做，其中主要目標是要讓他降落(-y座標*2)、越靠中心越好(-x座標)，在降落時y的速度不可以太快不然容易墜毀(-y速度*0.5)，而參數則是代表了這三項的重要性。（success1.gif）

```
reward = float(reward) - abs(next_state[1])*2 - abs(next_state[0]) - abs(next_state[3])*0.5
```

- Fail:跟前一個很像，只是將x的部分拔掉，並且將y速度參數調為1，但這個卻train不起來，我認為是因爲y方向速度和y座標本身是反向的關係，會導致agent不知道往哪個方向是好，造成不穩定。

```
reward = float(reward) - abs(next_state[1])*2 - abs(next_state[3])
```

- Success2:從fail那次做修正，將y速度的參數調為0.5，就train成功了，我認為是因爲讓agent更注重y座標，將y速度作為輔助，就可以降低fail那次產生的問題。（success2.gif）

```
reward = float(reward) - abs(next_state[1])*2 - abs(next_state[3])*0.5
```