HW6 心得:

本次作業我大致沿用 HW5 的設定，使用的是 **REINFORCE algorithm**

env 我原本嘗試了 'Acrobot-v1', 'FrozenLake-v0', 'NChain-v0'，但不知道為什麼，Acrobot-v1 愈跑愈久，FrozenLake 跟 NChain 完全 Build 不起來，於是放棄。

接著 env 改成 **'MountainCar-v0'**

而神經網路的部分我把 HW5 的 **nn 改成如下**:

```
class policyNet(nn.Module):
    def __init__(self):
        super(policyNet, self).__init__()
        self.L1 = nn.Linear(2, 16)
        self.L2 = nn.Linear(16, 8)
        self.out = nn.Linear(8, 3)
    def forward(self, x): # 預設 forward
        x = F.relu(self.L1(x))
        x = F.relu(self.L2(x))
        x = self.out(x)
        x = F.softmax(x)
        return x
```

原本想多新增幾層 NN 看效果會不會比較好，但我的電腦不是挺給力的，實在是 train 太~~~久~~~~了~~~~ 眼看著 Deadline 就要到了，GG……。

--- 拿到助教最新上傳的 Baseline Code 後 ---

由於我一執行就會當機……，最後發現把下面這兩行註解掉後程式就跑的很順利了

```
# if e % 100 == 0:
#   env.render() # 這行是用來顯示訓練畫面的，但是很吃資源，我一開就會當機.....
```

但是訓練時間仍然要很久，於是我又把 num_epsoide 調低到 3000，減少一點訓練時間。

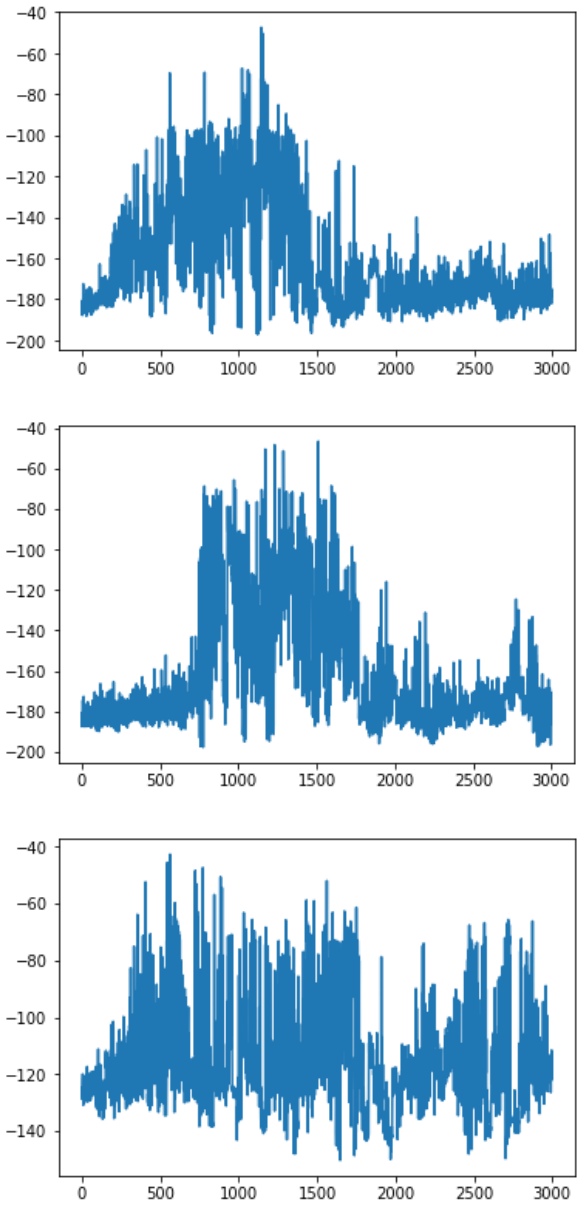
並嘗試了三種不同的 Reward Shaping，參數分別為:

c = 0.3, k = 5

c = 0.6, k = 8

c = 0.9, k = 11

最後**依照順序**訓練出來的 Model 比較圖如下:



我發現到 c & k 這兩個參數的選擇並不是愈大愈好，也不是愈小就會愈差，RL 訓練的穩定度以及整體效能會隨著不同的 reward shaping 而改變，看來要找出最佳的參數果然不是一件簡單的事情。