謝忱 R06921088
ADL HW3

1. Basic Performance (6%)

Describe your Policy Gradient & DQN model (1% + 1%)

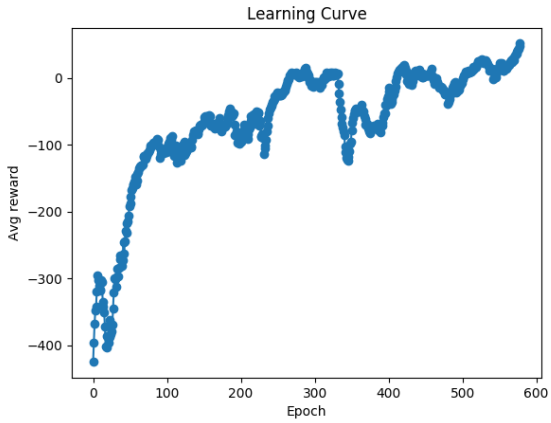
Policy Gradient:

model 的部分主要是輸入進環境的 output:state,經過兩層 NN 最後經過 softmax 輸出概率, 訓練時先讓環境的 state 進入 model,得到動作的概率,然後對動作做選擇.make_action 的 function 主要是處理動作的選擇,具體是對 model 產生的 class 概率做一個 Categorical 的 distributions,然後對分布採樣後輸出.輸出的動作經過環境後得到一個回合的 reward,當每一個回合結束後對整個回合裏面的 reward 進行 loss 計算.

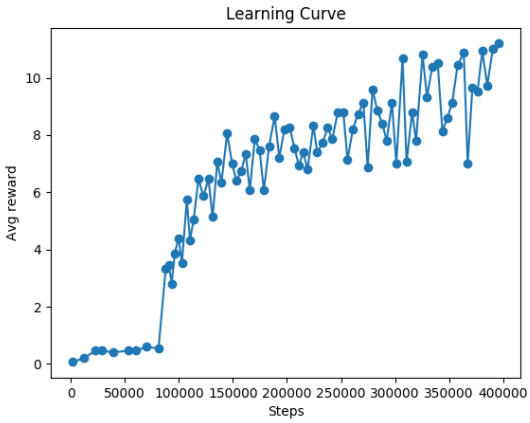
DQN model:

model 的部分爲普通的 cnn 架構, 有三層 cnn 最後加上一層 nn,輸出 dim 爲 action 的數量. 但是會有兩個同樣的 cnn model,一個爲當前的 net,另一個爲預測的 net.同樣是對環境 output 的 state 進入到當前的 net 做一個 action 的選擇,經過設定好的 update 的 freq, 再進行 loss 的更新, 這時環境的 output 會包含 next state 也就是下一步的狀態,這個 next state 輸入進 target net 中作預測 value 的動作,得到預測的 value,同時有之前的 reward,兩者相加後與目前狀態的 value 做 loss 的計算,就可以進行學習.

Plot the learning curve to show the performance of your Policy Gradient on LunarLander (2%)



Plot the learning curve to show the performance of your DQN on Assualt (2%)



X-axis: number of time steps

Y-axis: average reward in last n episodes. You can arbitrarily choose n to make your figure clear.

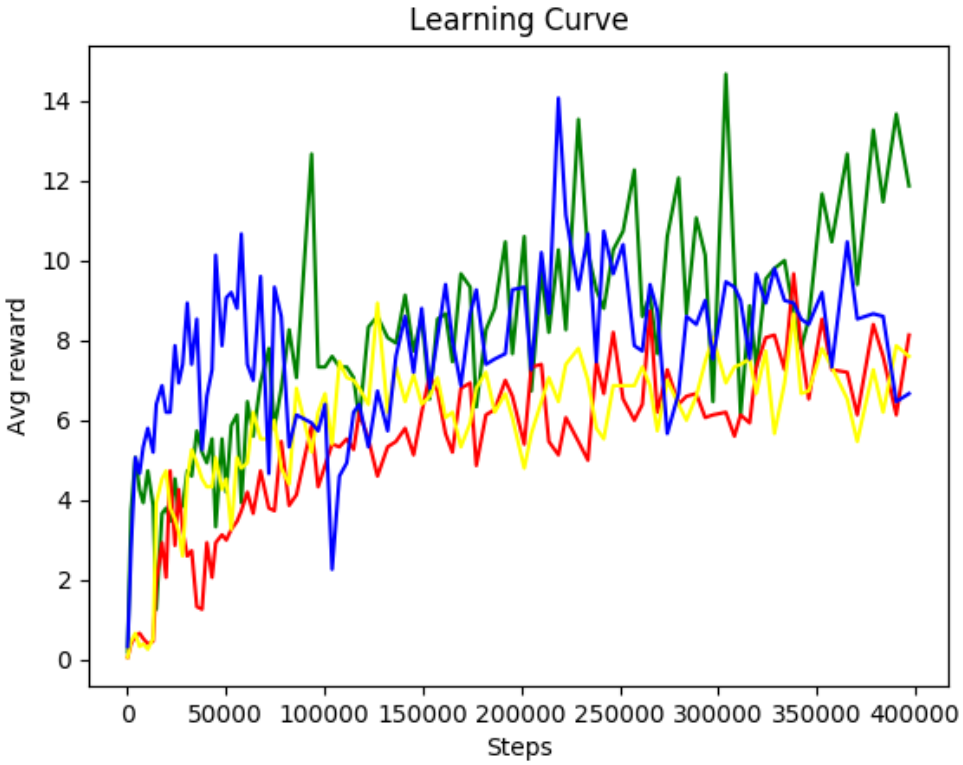
2. Experimenting with DQN hyperparameters (2%)

Choose one hyperparameter of your choice and run at least three other settings of this hyperparameter

You should find a hyperparameter that makes a nontrivial difference to DQN.

For example, if you just choose network hidden size in {256, 1126, 10000, 20}, you might not get full score in this part.

Plot all four learning curves in the same figure (1%)



plt.plot(x, y0, color='green', label='Origin')
plt.plot(x, y1, color='red', label='Gamma0.85')
plt.plot(x, y2, color='yellow', label='Update_freq5000')
plt.plot(x, y3, color='blue', label='New_architecture')

Explain why you choose this hyperparameter and how it affect the results (0.5% + 0.5%)

Candidates: gamma, network architecture, exploration schedule/rule, target network update frequency, etc.

1.綠線爲初始的 hyperparameter, GAMMA = 0.99, target_update_freq = 1000, model 爲三層 CNN 加上一層 NN.

2.紅線調整 GAMMA=0.85,其餘參數一致,GAMMA 在 Q-Value 扮演着未來系數的角色,Q values: rewards + gamma * max(Q(s_{t+1}, a))Q 的大小決定了其要考慮多少未來的成分,對未來考慮的越多,預測也相對會更準確,因爲 0.99 avg reward 相對比較高.

3.黃色爲調整更新的頻率,意義就是 step 到達一次這個頻率,才會進行更新,value base 的 RL 方法的優勢是對於下一個 value 的快速準確的估計,可見相對較慢的更新並不能幫助到 model 的學習.

4.藍線在原有的 model 基礎架構上增加了一層 cnn 和一層 nn,更深層的 cnn model 對於 feature 的提取能力越強,也可以看到藍線初期的學習速度相較於綠線更快.

3. Improvements to Policy Gradient & DQN / Other RL methods (2% + 2%)

Choose two improvements to PG & DQN or other RL methods.

Other RL methods include

Actor-Critic series (A2C, A3C, ACKTR etc.)

DDPG, Curiosity-Driven Learning, AlphaStar etc.

For each method you choose,

describe why they can improve the performance (1%)

PG, Actor-Critic on the LunarLander_V2

Actor-Critic 可以分爲兩部分來看，Actor 的部分前身就是 PG，都是基於回合的更新來學習，對於連續的動作空間可以選擇出合適的動作。同時 Critic 可以基於單步更新評分，Actor 可以根據評分來調整選擇動作的概率。

DQN, DDQN on the PongNoFrameskip-v4

Q-learning 會存在過度估計的問題,DQN 本質上是基於 Q-learning 的,所以也會存在過度估計的問題.double DQN 其基本思想为：将 target Q 中选择和评估动作分离，让它们使用不同的 Q 网络。例如現在需要計算 Q1,同樣使用另一個 Q2 進行選擇,然後帶入到 Q1 中,因爲此時的 Q1 不像單純的 DQN 取 MAX 的操作,所得的結果會進行平均的操作,這樣就可以得到相對沒有太多偏差的結果.

plot the graph to compare results with and without improvement (1%)

PG, Actor-Critic on the LunarLander_V2

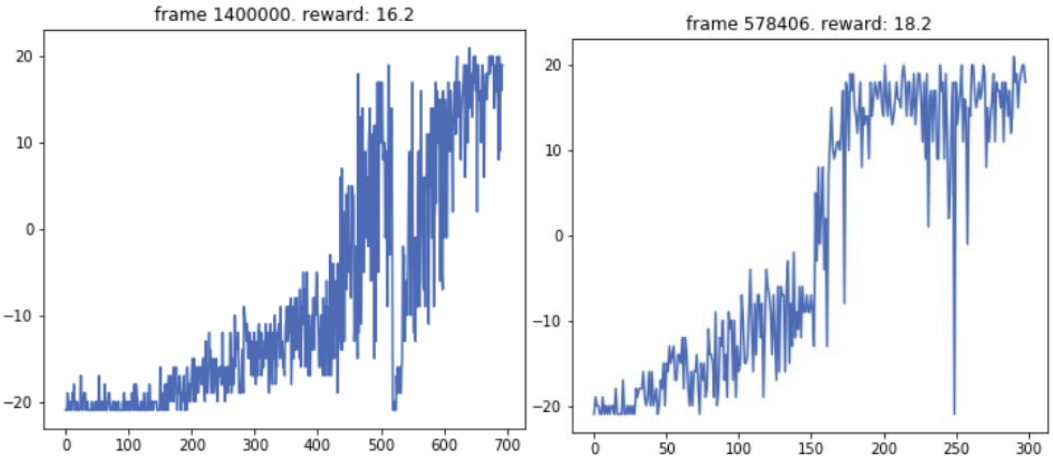
Mean Avg Reward

PG = 42.756　　Actor-Critic = 256.21

```
load model from pg.cpt
Episode 0       Reward: -82.72179285269034
Episode 1       Reward: 178.9586400592239
Episode 2       Reward: 4.356841424074275
Episode 3       Reward: -56.32147420725079
Episode 4       Reward: 229.45282717545143
Episode 5       Reward: 55.2040460352209
Episode 6       Reward: 212.15153236845896
Episode 7       Reward: -32.27301704894971
Episode 8       Reward: -19.026645515434083
Episode 9       Reward: 188.93015931142725
Episode 10      Reward: 42.52683964179522
Episode 11      Reward: 32.22293422609019
Episode 12      Reward: 220.44201500565993
Episode 13      Reward: -89.79674759962174
Episode 14      Reward: 33.811649192386284
Episode 15      Reward: -86.12508432161657
Episode 16      Reward: 0.9109758678798272
Episode 17      Reward: 32.23851455148517
Episode 18      Reward: 52.26186209011854
Episode 19      Reward: 227.92322639205264
Episode 20      Reward: -16.33705856596569
Episode 21      Reward: 235.91996448026273
Episode 22      Reward: -38.84541597448923
Episode 23      Reward: -29.71279273486256
Episode 24      Reward: -18.15233779674365
Episode 25      Reward: -13.97740578564823
Episode 26      Reward: -17.13744206647189
Episode 27      Reward: -24.297426401493468
Episode 28      Reward: 20.279584872099733
Episode 29      Reward: 39.81676959409555
Run 30 episodes
Mean: 42.75612471388481
xiec@xiec-System-Product-Name:~/ADL/hw3$
```

```
  action_probs = F.softmax(self.action_layer(
Episode 1       Reward: 255.8048672334179
Episode 2       Reward: 266.76515777414767
Episode 3       Reward: 220.83652960669258
Episode 4       Reward: 276.0579039663611
Episode 5       Reward: 264.4747210944043
Episode 6       Reward: 271.82755600296287
Episode 7       Reward: 265.7047282024406
Episode 8       Reward: 280.91746733640304
Episode 9       Reward: 234.09138713020153
Episode 10      Reward: 270.69454392445465
Episode 11      Reward: 232.05168861169915
Episode 12      Reward: 298.15238895063004
Episode 13      Reward: 257.1000254536382
Episode 14      Reward: 280.14497175944547
Episode 15      Reward: 235.48268545762565
Episode 16      Reward: 238.66906710182184
Episode 17      Reward: 250.41186315404823
Episode 18      Reward: 255.14631794414456
Episode 19      Reward: 255.37426005203233
Episode 20      Reward: 294.6275984616994
Episode 21      Reward: 234.5730528455052
Episode 22      Reward: 258.4481434219325
Episode 23      Reward: 212.66739308084271
Episode 24      Reward: 250.17478965636252
Episode 25      Reward: 251.08785561281937
Episode 26      Reward: 250.19092821771974
Episode 27      Reward: 287.1158882958108
Episode 28      Reward: 268.4602918814205
Episode 29      Reward: 224.2417918977332
Episode 30      Reward: 244.98085945546146
Mean: 256.20922411946265
xiec@xiec-System-Product-Name:~/ADL/hw3/Actor
```

DQN, DDQN on the PongNoFrameskip-v4
Mean Avg Reward
DQN = 16.2 DDQN = 18.2



You can train on any environment to show your results, so you should better choose environment where you can see significant difference between those methods.
Grading will simultaneously consider your description and actual model performance.