Hw3 Report

學號：r05921012　　姓名：吳宗澤

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

PG :

input(80,80,1) -> conv(8,8,4,16) with relu-> conv(4,4,2,32) with relu -> dense(128) with relu -> dense(6) with softmax

RMSPropOptimizer, learning rate : 0.0001 with decay 0.99

DQN :

Evaluation Net

input(84,84,4) -> conv(8,8,4,16) with relu -> conv(4,4,2,32) with relu -> conv(3,3,1,64) with relu -> dense(512) with relu -> dense(4) with linear activation function

Target Net

input(84,84,4) -> conv(8,8,4,16) with relu -> conv(4,4,2,32) with relu -> conv(3,3,1,64) with relu -> dense(512) with relu -> dense(4) with linear activation function

RMSPropOptimizer, learning rate : 0.0001 with decay 0.99

Memory size : 50000

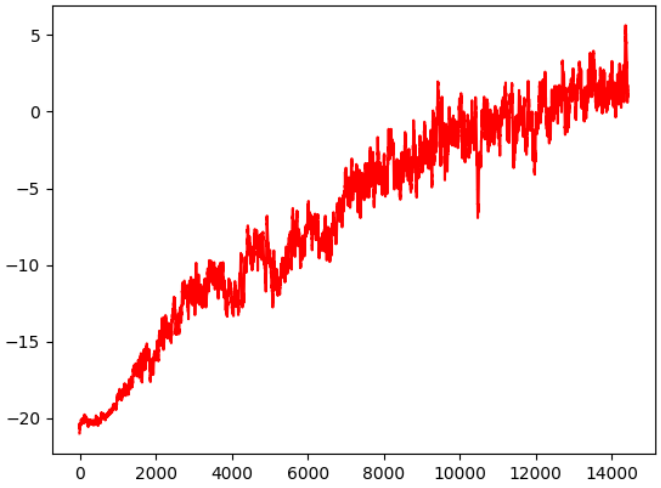
Target net update frequency : 10000

Batch_size : 32

Epsilon : 1.0 -> 0.05 till 1 million time steps

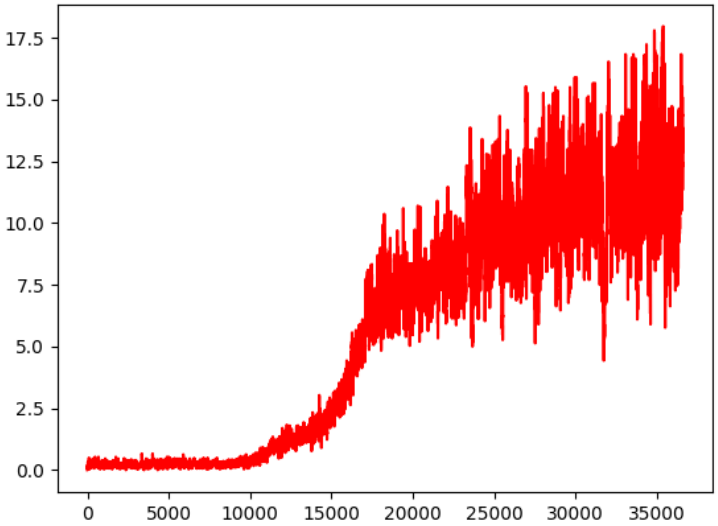
2. Plot the learning curve to show the performance of your Policy Gradient on Pong (2%)

因為少用到一個助教的 tips 就是每次有 reward 的時候要把 running reward 給歸 0，導致我要到很後面才會收斂到 baseline 的 performance，也發現因為一開始我是自己去設定 weight 的 initialization，也發現如果 weight 沒有設好的話，很容易卡在 local minimum 上不去，要將 weight 的 mean 設在 0,stddev 設在 0.02 才會一開始比較有 random 的 action，能增加 exploration 的機率。表現的較 dqn 穩定，variance 比較小



3. Plot the learning curve to show the performance of your DQN on Breakout (2%)

表現得很不穩定時好時壞，而且 variance 還滿大的。但是平均表現在沒有 clip 的時候可以到達 60 多分。



4. Experimenting with DQN hyperparameters (4%)

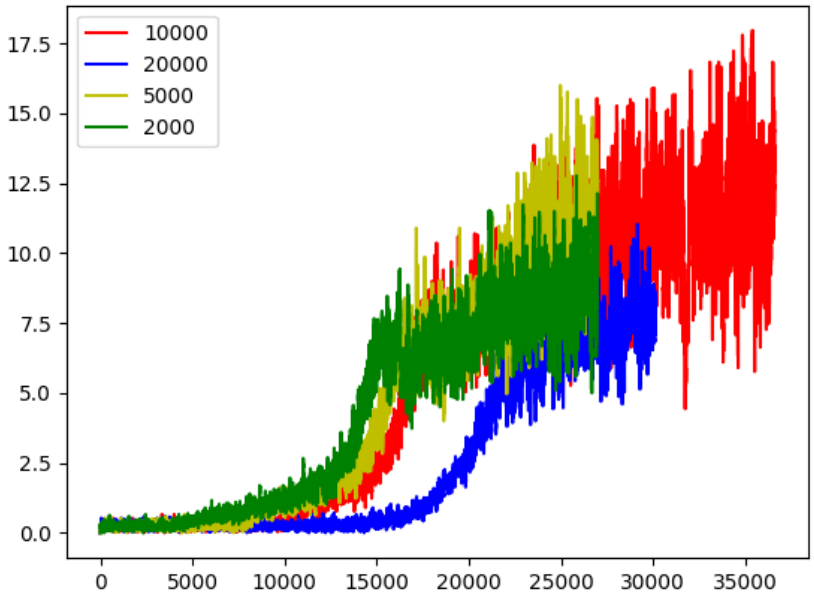
Choose one hyperparameter of your choice and run at least three other settings of this hyperparameter

You should find a hyperparameter that makes a nontrivial difference on performance

Plot all four learning curves in the same graph (2%)

Explain why you choose this hyperparameter and how it effect the results (2%)

我選擇的是 target net 的 update frequency: 圖上的 legend 代表的數字是多少個 time steps update 一次，可以看到在每 2000 步 update 一次效果最好，但是相對來說，我們所需要花費的時間會變得更多。原本要這樣 update 的原因是為了穩定 performance，因為每次的表現都會起起伏伏，如果 update 的頻率太高，會因為一次不小心學到不好的表現，而不小心讓 target net update 到不好的地方，因此可以看到跟 5000 頻率的比起來效果就後面就沒有比較好。而 update 頻率到 20000 時，也會因為更新的頻率太慢而導致一開始 exploration 的時候學到的次數太少，讓效果表現也不佳，因此要選擇一個剛剛好的 update frequency。



BONUS

- Improvements to DQN (2%)

Implement at least two improvements to DQN (p.9) and describe why they can improve the performance (1%)

我所選擇的兩種方式，一個是 Dueling DQN 另一個則是 Double Dueling DQN，對 Dueling DQN 來說，我是在最後一層拆解成兩個，一

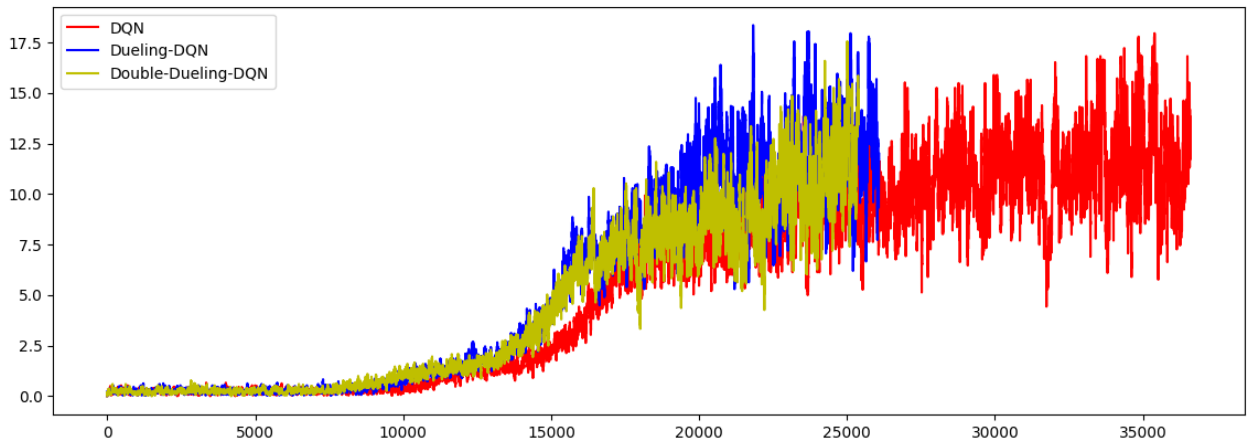
個是給 state 的 value 評斷，而另一個則是 action 的 value，好處就是給了他一個 advantage function 讓他知道說這次的表現到底跟平均的表現比起來是好還是壞讓他知道這次要更新的步伐要多大，也讓他知道說遊戲玩到這個 state 的情況是好還是壞，將這兩點融合進去新的 target value function 裡並利用 Bellman Equation 像 DQN 一樣去做學習。

而在 Double Dueling DQN 裡面除了保有 Dueling DQN 的好處以外，我們將原本在 target value 裡面 Q 值都是由 target net 對下一個 state 的 argmax 值所決定，但因為每次都選擇最 greedy 的決定不一定是最好的選擇(如：很多人在家的 cliff walking)，因此在這裡我們是利用 evaluate net 對下一個 state 最大值的 action index 來選擇 target net 對下一個 state 的 Q 值出來，這樣有時可以讓表現更穩定一些。

Plot a graph to compare and analyze the results with and without the

improvements (1%)

從下圖我們可以清楚地看到 Dueling-DQN 跟 Double-Dueling-DQN 比 DQN 還要好，且學習的 efficiency 比較高，而在 Dueling-DQN 和 Double-Dueling-DQN 中在一開始的確是不分上下，但在後期可以看出 Dueling DQN 可以達到稍微好一些的成績但是表現的 variance 比較大不像 Double 的可以比較穩定的上升，穩定上升的確是符合當初的理論，而效果比較好我是認為因為在 MDP 的理論裡面有 paper 的確有證明過每次都選擇最 greedy 的 choice 有一個機率保證讓他有比較好表現。



- Improvements to Policy Gradient (2%)

  Implement at least two improvements to Policy Gradient (p.8) and describe

  why they can improve the performance (1%)

  我第一個 implement 的是 advantage actor critic，它結合了 policy based 與 value based 的 RL，讓 actor 在更新的時候有 critic 來輔佐他更新，在實現時我發現，我們必須讓 critic 的更新速度要比 actor 還要快，因此我讓 critic 的 learning rate 比 actor 還要大，除此之外因為原本 value based 的方法像 DQN 有個缺點就是 converge 到好的結果的速度很慢，而 policy based 的像 Reinforce 的缺點是雖然 converge 的比較快，但是很容易卡在 local minimum 的地方，而 actor critic 就像介於兩著之間，結合彼此的好處來進行更新的動作。

  我第二個 improvement 做的是模仿 Actor Critic style 裡面 Proximal Policy Optimization 裡面 Critic 的 objective function 的形式，讓我的 advantage function 變為取未來幾個 step 並做 discounted 當成 target value，與原本的 objective function 是不同的。

  $$K\text{-steps: } \hat{A}_t = \sum_{i=1}^{K} \gamma^{i-1} r_{t+i} \quad + \quad \gamma^{K-1} V_\phi(s_{t+K}) - V_\phi(s_t).$$

  但當我要進行 actor 裡面的 loss 給 clipped 時，效果反而變差了，clip ratio 那我嘗試過 0.1, 0.2,0.05，因此我在這邊做的實驗室只有包含 Critic 的改變。

  Plot a graph to compare and analyze the results with and without the

  improvements (1%)

  首先，我們可以很清楚的看到 actor critic 一開始學習的效率就比 reinforce 好很多，但在後期的表現卡住，我認為是因為我參數可能沒有調得很好的問題，因為所有的參數我都跟 Reinforce 是相同的但畢竟現在多了 critic network，有些參數是要再調整一下。再來我們 PPO 那邊可以看到花不到一半的 epoch 就可以達到根 Reinforce 相同的境界，但

因為時間不足我沒有持續訓練下去，由此實驗可以知道 PPO 的 critic 的改變可以讓效果上升許多，甚至高過 a2c 的平均 3~4 分。