ADLxMLDS HW3 Report

R06922022 資工所碩一 曹燁文

（一）Basic Performance (6%)

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

PG:

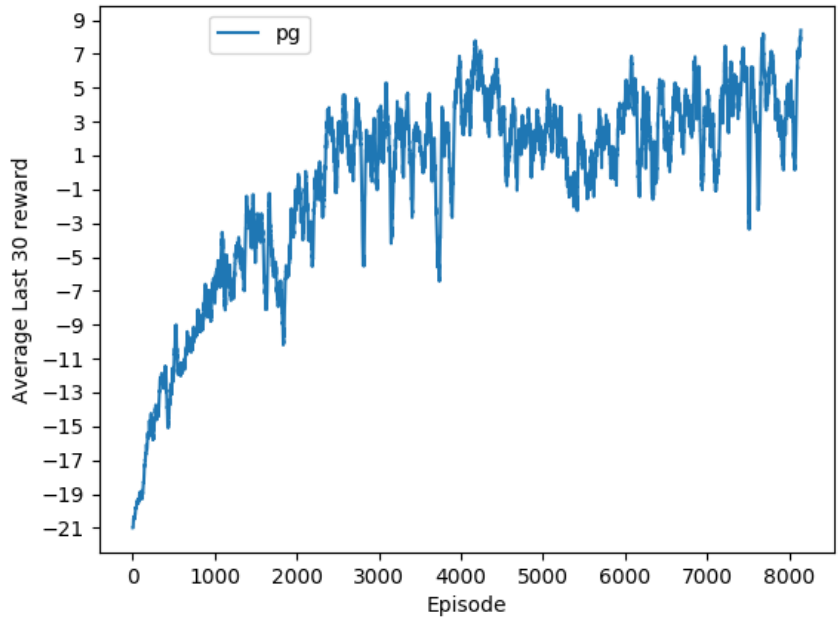
Model 第一層為 Conv2D，之後兩層是 Dense，最後接一層 action_size 的 Dense，使用 relu，initial 使用 he_uniform。Preprocess 使用 Karpathy 的方式，即切掉上下、downsample 成 80*80、去除 background、最後變成 0,1 二值化然後壓平成 6400 維的向量。其餘部分按照 REINFORCE 演算法實作，使用助教提供的 tips，包含 state 相減、discount and normalize reward。

DQN:

Model 參數與助教提供的相同，三層 Conv2D，使用 relu，壓平成 7*7*64=3136 維向量到一層 512 的 Dense，使用 leaky_relu(alpha=0.01)，最後輸出成 action_size，loss 使用 MSE，optimizer 使用 rmsprop(rho=0.99)，將 rho 從 0.9 調到 0.99 有顯著進步。其餘部分按照 paper 實作，將每步的 state, action, reward, next_state 存到 memory，每次 sample 做學習。

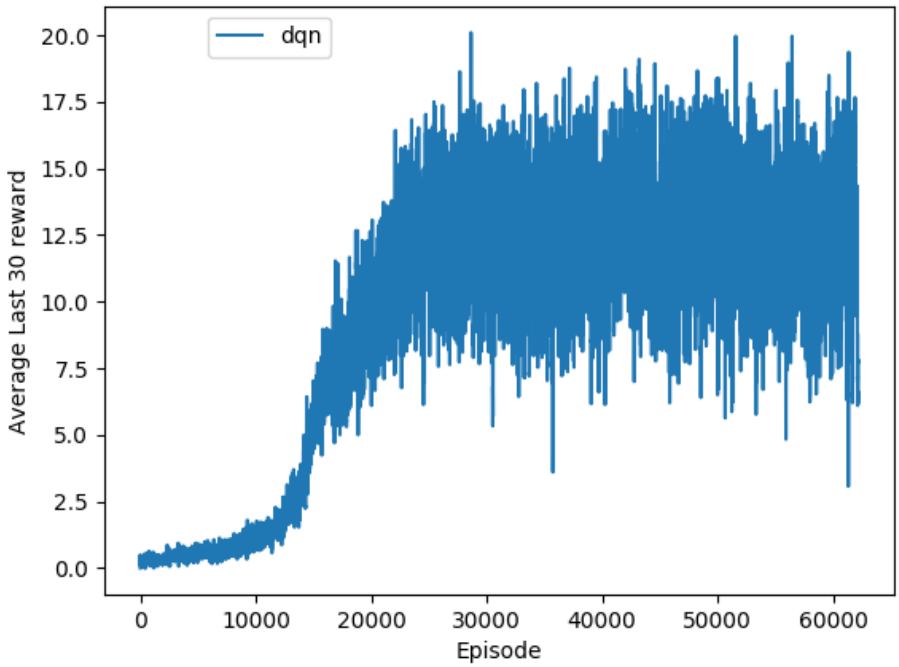
Plot the learning curve of Policy Gradient on Pong (2%):

Score: ~7



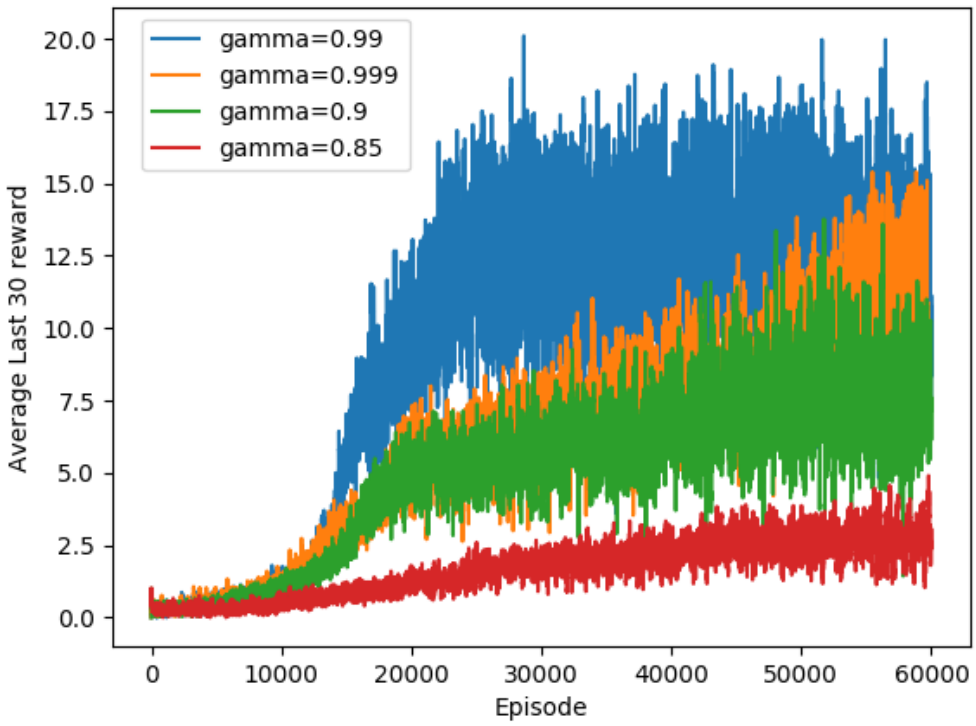
Plot the learning curve of DQN on Breakout (2%):

Test Score: ~60



（二）Experimenting with DQN hyperparameters (4%)

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



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

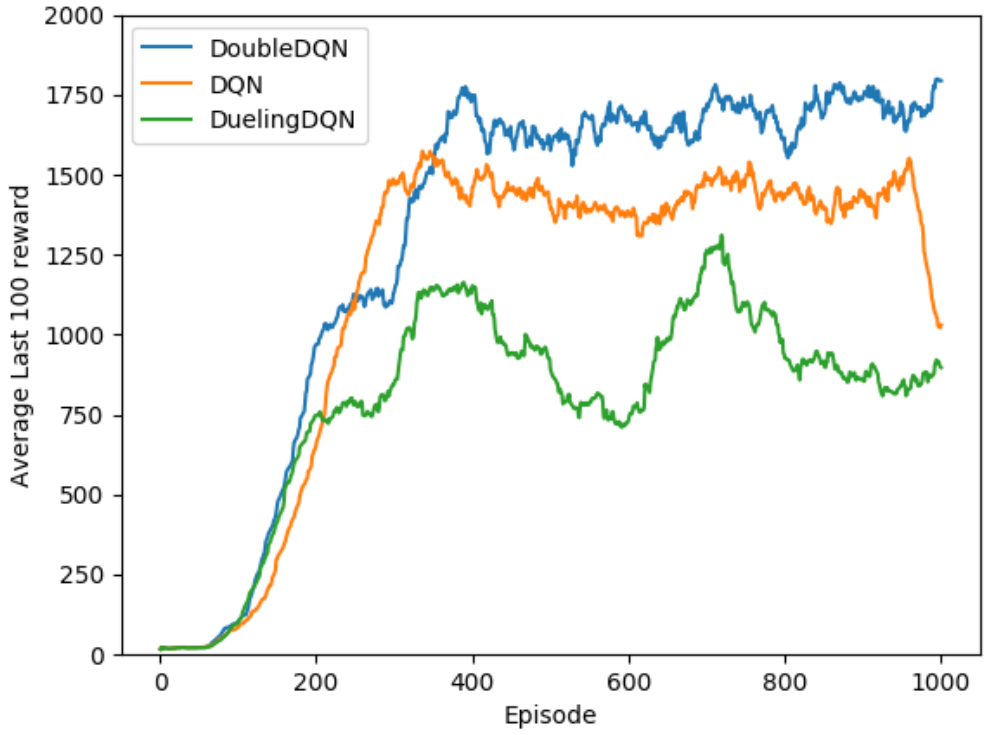
gamma 為 discounting factor，是 RL 學習中的重要參數，故選此作為實驗對象。0.99 是原先實驗的參數，可以看出他表現最好，最後平均約為 14 分。將 gamma 調大變成 0.999 會讓學習速度變慢，最後比 0.99 時表現略差。0.9 的時候表現更差，平均不到 10 分，0.85 的時候更是只有 2.5 分左右。

（三）Bonus

Bonus 的程式碼皆參考自莫煩教程 (https://morvanzhou.github.io/tutorials/machine-learning/reinforcement-learning/)

環境使用 CartPole-v0，reward 使用他建議的 reward: r1 為距離水平中心的 reward，越遠 reward 越低；r2 為桿子的垂直程度，越直分數越高，由於要讓 CartPole 高分需要同時滿足 r1 和 r2，所以訓練的 reward 使用 r1+r2，實際印出來看的 reward 使用原先的 reward。

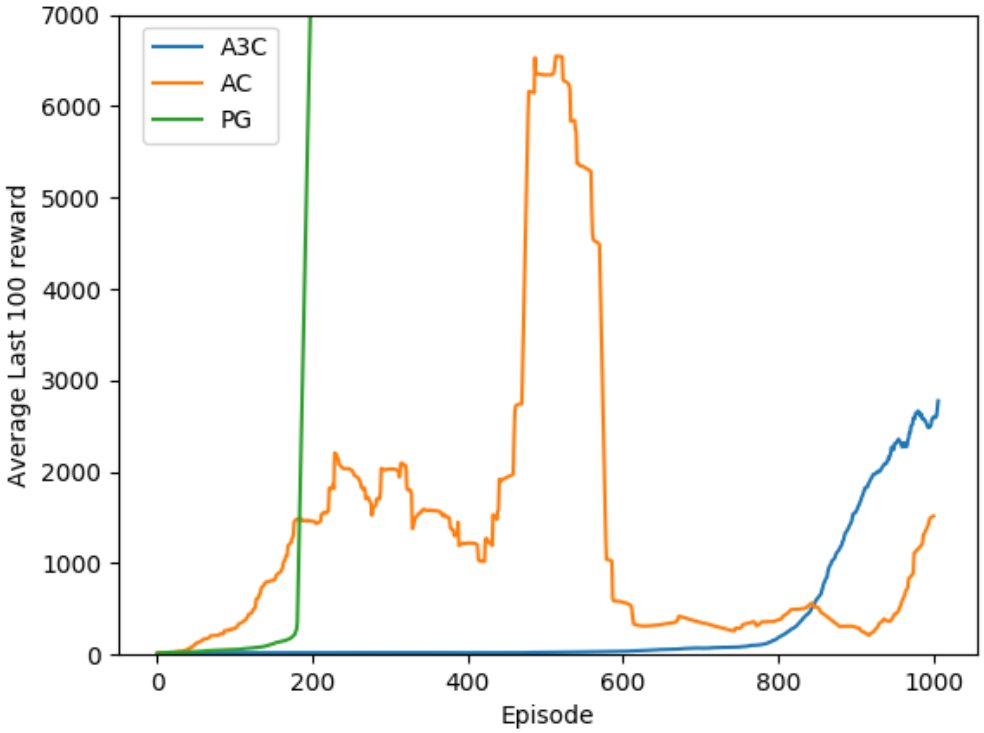
1. DQN, Double_DQN, Dueling_DQN



三個 model 的參數都是一層 dense，64 neurons，memory size: 3000，lr=0.005，gamma=0.9。結果如圖，Double DQN 比一般 DQN 略好，原先預期會最好的 Dueling DQN 表現最差，我想這跟環境本身比較簡單有關，又或者

是此組參數適合 DQN 不適合 Dueling。此外有一個想法是用 Dueling 那篇 paper 舉的例子：賽車遊戲，當路上沒有障礙物的時候，無論哪個 action 都不會有特別的 advantage，當障礙物出現時，advantage 的重要性出現，因為錯誤的 action 會造成很大的 reward 損失。但是在 CartPole 的情況中，使用自定義的、具有連續性質的 reward，錯誤的 action 帶來的 reward 損失很小，所以就像是賽車例子中前半部分，advantage 沒有發揮功效。

2. PG, AC, A3C



PG 為一層 Dense，大小為 10 neurons，lr=0.02，gamma=0.99。AC 和 A3C 參數較多在此省略。看圖明顯看出 PG 的效果極好，而且這是我設定 step 在 40000 以內尚未結束就停止，所以最高分是 40000，圖片顯示的是 window size=100 的移動平均，若是讓 PG 繼續 train 下去會在更早將 performance 帶上去。AC 中間過程表現不錯，但跟聽說到的 AC 的毛病一樣，會不穩定且不收斂，訓練過程中間幾個高的地方 reward 達到 40000 以上，但是很快的掉下來，不像 PG 一樣收斂。A3C 前面成長緩慢，到後面穩定成長，超過 AC 的表現。如同 DQN 那裡說的，可能是環境較為簡單，無法展現出 A3C 強大的地方，唯一能感受出的是我訓練時開 12 個 threads，A3C 十分鐘之內可以跑完，AC 等了好幾個小時才結束，雖然與中間 reward 飆升導致時間較長也有關係，但仍然可以感受到 A3C 平行與穩定的力量。