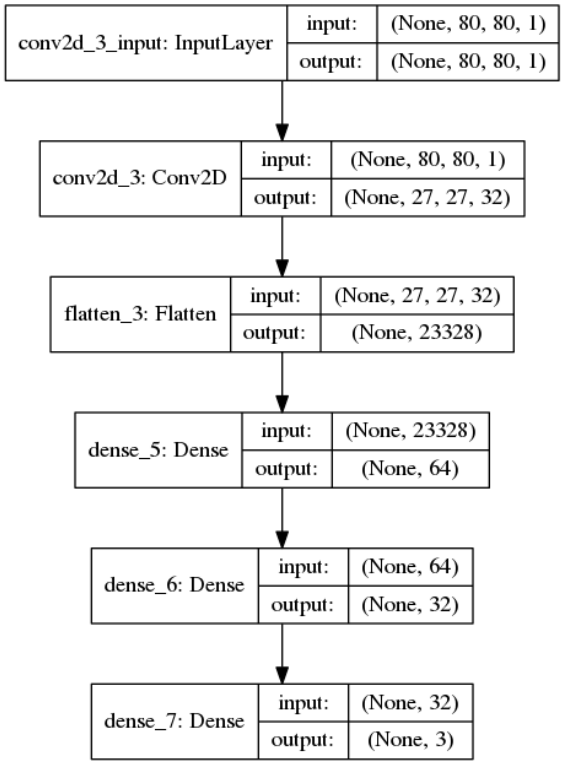
ADLxMLDS HW3 B03705006 侯舜元

Model description Pong (Policy Gradient) (1%)

資料前處理：

將 input 的 frame 刪除上下不重要的部份，且將 input 的 frame 轉成黑白畫面，最後再 resize 成(80,80,1)。由於原本的 action space 為 6，但其實只有三種動作，故我們將 output 的 action space 改成 3。

模型架構如下圖:



Training

在每一個 episode 中，會紀錄下他的 state 的變化、他所的做的 action 和他得到的 reward。在每一個 episode 結束時，會 train_on_batch 重新 fit 一次 model。

Model input: 經過 preprocess 的 state 變化 （state 和上一個 state 相減）

Model output: (預測的 action 機率分佈) + (Learning Rate) * (gradient) * (discount_reward)

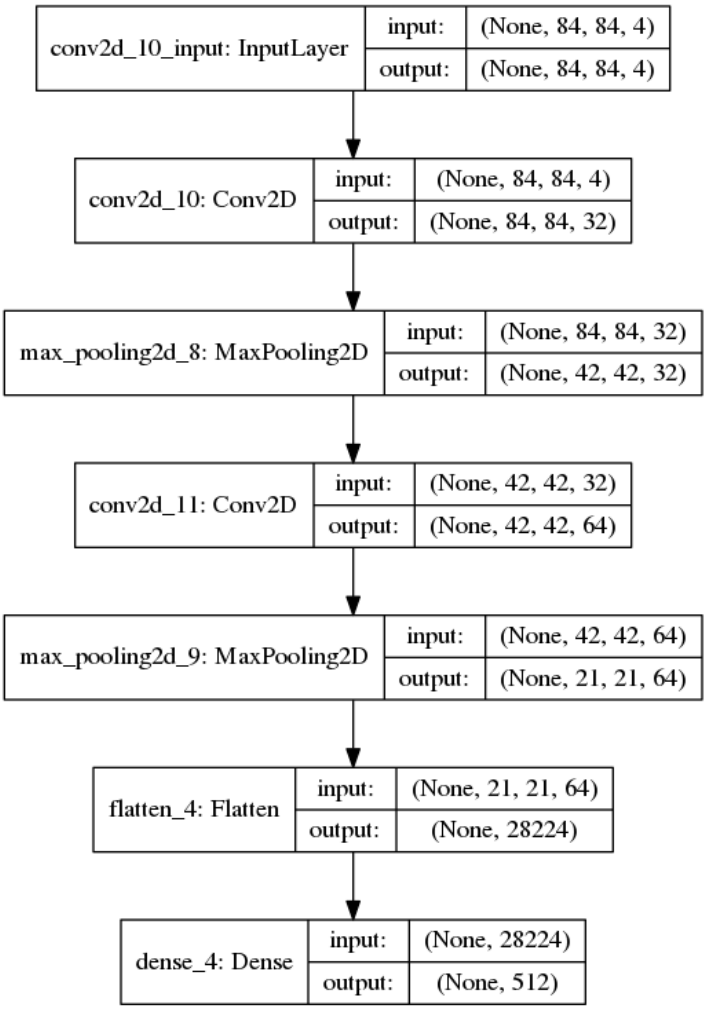
gradient 為這次動作向量(Ex: [0,0,1])與原先預測的 action 機率分佈相減

discount_reward 為經過時間 discount 過後的 reward

模型設計 Reference: https://github.com/keon/policy-gradient

Model description Breakout (DQN Model) (1%)

模型架構如下圖：



DQN Parameters

learning rate: 0.002
online network update frequency: 1/episode
target netwrok update frequency: 100/episode
epsilion: 0.0001
epsilion decay: 0.995
epsilion min: 0.00001
gamma: 0.99
replay batch size: 64
Max steps per episode: 10000
Max memory Size = 50000

Training

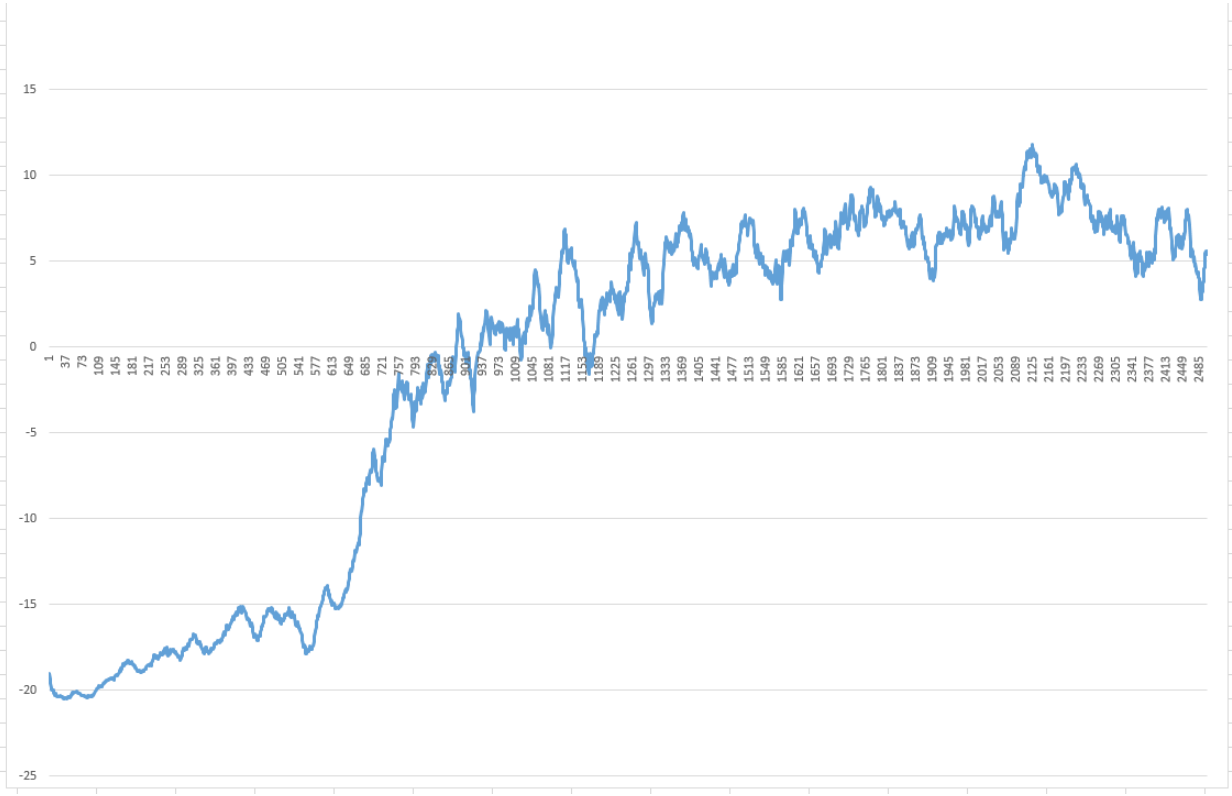
在每一個 episode 中，會紀錄下他的 state 的變化、他所的做的 action 和他得到的 reward。我們按照 online network update frequency 從 history 中 sample 出 minibatch 來 train 我們的 agent。

Model Input : 遊戲的 state
Model output: reward + gamma * (Max target_network 預測的結果)
而我們會按照 target network update 的週期同步更新我們的 target network。

模型設計 Reference:
https://github.com/coreylynch/async-rl/blob/master/async_dqn.py
https://keon.io/deep-q-learning/

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

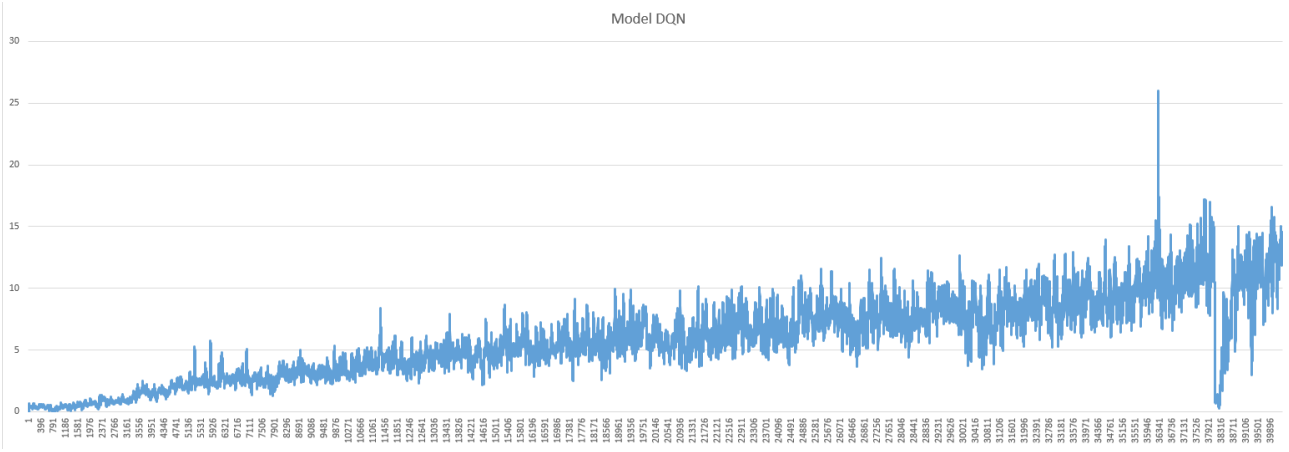
Baseline model



(x-axis : episode, y-axis : 30 episodes moving average)

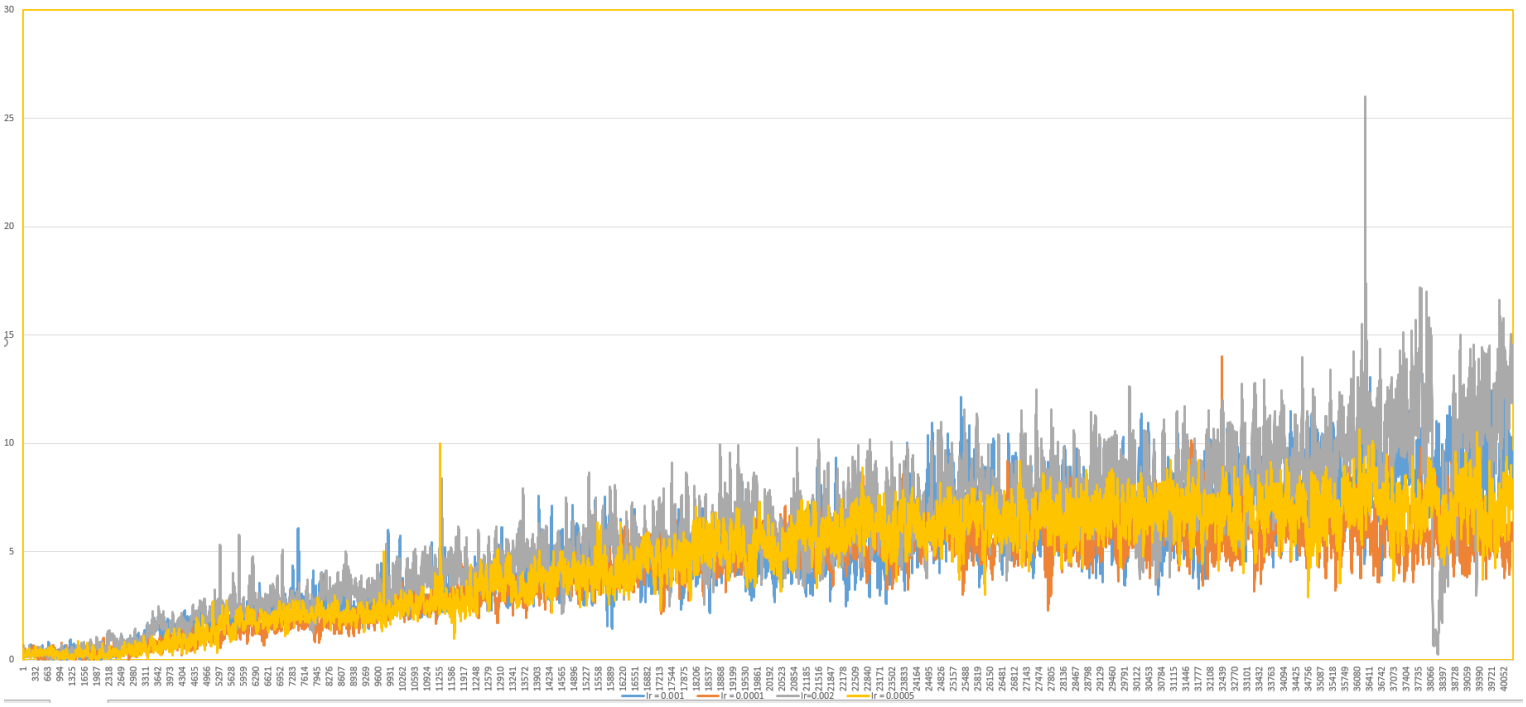
在約 1000 個 episode 後就可以贏過電腦了，最後到 2000 episode 跑 test 大概可以獲得 8 分左右。

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



Experimenting with DQN hyperparameters (4%)

想了解 learning rate 在相同訓練次數下(約 40000 episode)，對 performance 和 agent 學習的速度的影響。故我選擇了四種不同的 learning rate : 0.0001, 0.0005, 0.001 ,0.002 做實驗，實驗結果如下



(x-axis : episode, y-axis : 30 episodes moving average)

橘線：lr 0.0001
黃線： lr 0.0005
藍線： lr 0.001

灰線： lr0.002

從學習的速度來看有看出，在比較高的 learning rate 上灰線(0.002)，在初期的地方就領先了其他 learning rate，直到最後他的 performance 平均來看都在其他 learning 高。而藍線(0.001)和黃線(0.0005)比較沒有太大的差異，學習的速度對比的 performance 差不多（圖上大量重疊）。而橘線(0.0001)看的成長稍微緩和了些，在同一個時點 performance 通常都是四個中墊底的。

而從分數震盪的幅度來說，可以明顯看到灰線(0.002)有著最大的變動幅度。而大致上也有依循著較高的 learning rate 有比較大的振幅。。

在四萬 episode 下四種不同 learning rate model 去跑 test 的結果：
橘線：lr 0.0001 → 10
黃線： lr 0.0005 → 25.5
藍線： lr 0.001 → 35
灰線： lr0.002 → 59 （過 baseline)
也是符合常理的推測，較高的 learning rate 能夠比較快的學習有比較好的 performance。

另外值得一提的是，在圖中沒有出現我有將其他三個過 baseline 的 model 拿去繼續 train，黃線： lr 0.0005 和藍線： lr 0.001 在約 5,6 萬 episode 的時候通過 baseline 了，在 training 的 reward 可以達到 12 附近。
但是橘線：lr 0.0001 跑了 9 萬多 episdoe 的學習曲線使終於 7,8 附近震盪，不見起色，test 也沒有過 baseline，故太低的 learning rate 可能很難 train 起來這次的 model。