[HW3 report]

- Basic Performance (6%)

1. Describe Policy Gradient model (1%)

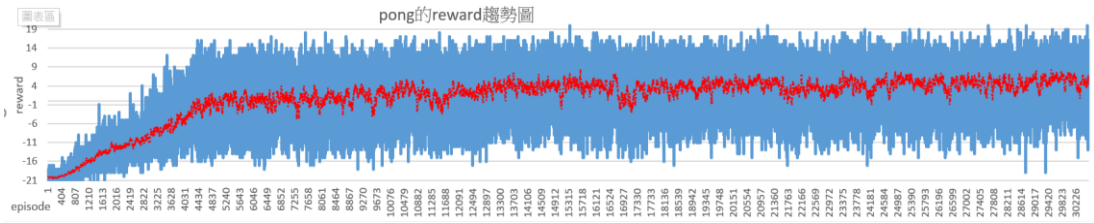
hidden layer neurons : 200 / batch size = 10
optimizer: RMSProp(learning rate = 1e-3, decay = 0.99)
gamma = 0.99(discount factor for reward)
使用兩層 nn 模型，皆利用 truncated_normal_initializer 來初始化參數為平均 0，標準差為 0.1 的 Truncated Distribution，第一層輸入為前處理成(80*80)的 image，接著用 relu 做 activation function，經過第二層後再把結果做 softmax 輸出，然後利用計算出的機率隨機選擇一個 action。在算 gradients 時，會先把 reward 用 gamma 的比率做 discount，並且用 tf.nn.moments 做 Batch Normalization。

2. Describe DQN model (1%)

使用助教提供的參考模型和參數設定，三層 conv2d，加上兩層 fully connected，利用 truncated_normal_initializer 來初始化參數為平均 0，標準差為 0.1 的 Truncated Distribution，bias 則全初始化為 0.1。

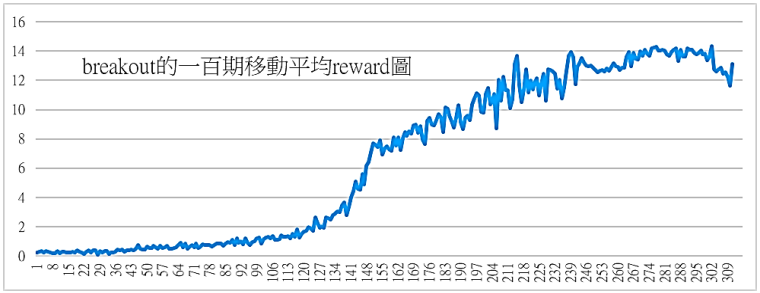
3. learning curve of Policy Gradient on Pong (2%)

藍色是每 ep 的 reward，紅色是 30 期移動平均，畫到過 baseline(>1)部分



4. learning curve of DQN on Breakout (2%)

x 軸代表每一百個 episode，總共跑了三萬多個 episode

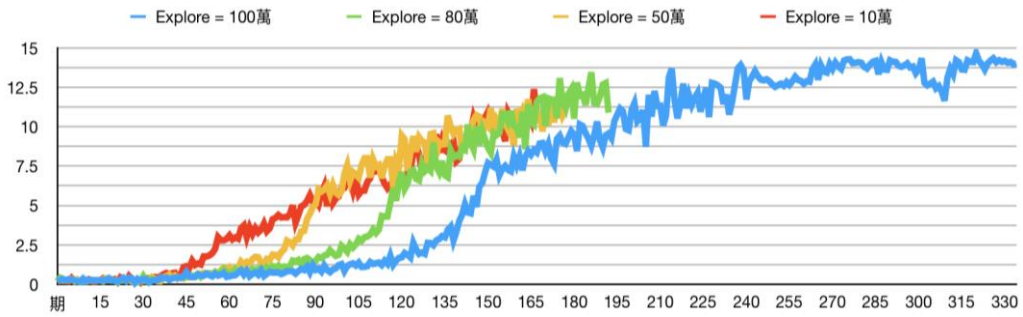


- Experimenting with DQN hyperparameters (4%)

1. Explore: 1000,000 / 800,000 / 500,000 / 100,000

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

x 軸代表每一百個 episode，藍色跑了三萬多個 episode，其他則跑了一六～一萬九 episode 之間



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

覺得 Exploration and Exploitation 是一個很有趣的問題，以前的作業是有 label ，所以訓練時就是去最小化 loss 就好，但在 RL，一切都要 agent 自己去學，多了需要去探索環境的問題，和以往有很大不同，因此這次作業感覺 random 佔了很大的比重，所以想試試看調 explore 的數值，看探索的 epsilon 下降很快的話對結果會是好或不好。

結果如上圖的學習曲線，四種參數都訓練的起來，且訓練到最後的結果其實相差不多，差別在於何時開始從平緩到穩定上升，explore 越小發生的時間點就越早，表示探索機率高真的會因為常試到不好的動作而讓平均 reward 升不上去，而一旦 epsilon 降到最低，reward 就會開始上升。另一個差別是開始上升時的斜率不同，紅色線上升的趨勢明顯比較平緩，表示探索的比較少，對學習效率是會有影響的，因為學到的環境資訊比較少，大概一萬三 ep 時就會被比較晚開始上升的黃色和綠色線追上，但除了紅色以外，其他三個顏色上升的斜率似乎都差不多，我比較意外的是，用前一萬九 ep 來比的話，藍色線其實是學得最差的，和其他三條線都沒有重疊，也許表示這個 breakout 的環境其實不用探索那麼久，可以早一點開始多用已知訊息來訓練，加快學習速度，不過因為時間問題，如果紅黃綠再繼續訓練下去，不知道後續會如何，會不會無法再升上去或是會被藍色追上。

- Bonus

1. Improvements to DQN (2%)

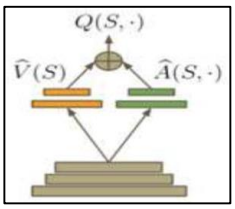
i. Implement and describe why they can improve the performance (1%)

我選擇在 Breakout-v0 上實作 Double Q-Learning 和 Dueling Network。Double Q-Learning 目的是減少取 max Q 值計算方式帶來的計算偏差，因為 Q 值中包含了 noise，只取最大的就容易受到 noise 影響，所以改用兩個 Q-learning 來折中原本的 Q-learning 與 target network，用當前的 Q 函數來選擇 action，用 target network 來計算 Q 值，而這樣能減少估計偏差是因為，如左下圖式子所示，把計算 Q 值函數叫做 Q1，如果用另一個同樣有 noise 問題的 Q2 來選擇動作，帶入到 Q1 中去，由於這裡的 Q1 並沒有取 max，所以得到的結果可正可負，平均下來就會接近於 0，得到一個相對無偏的結果。

Double DQN: Remove upward bias caused by $\max_a Q(s, a, \mathbf{w})$

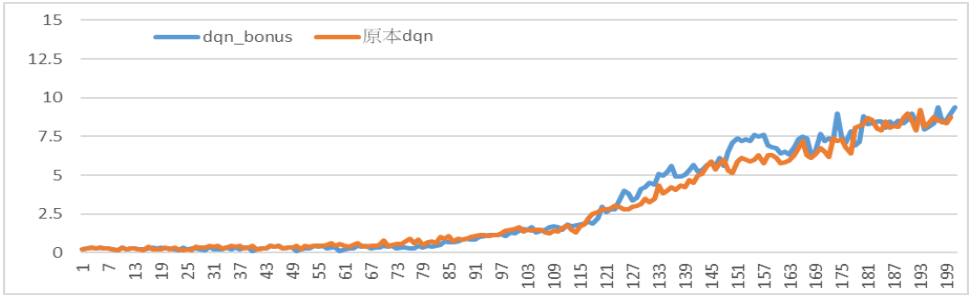
- Current Q-network $\mathbf{w}$ is used to select actions
- Older Q-network $\mathbf{w}^-$ is used to evaluate actions

$$l = \left(r + \gamma Q(s', \underset{a'}{\operatorname{argmax}}\, Q(s', a', \mathbf{w}), \mathbf{w}^-) - Q(s, a, \mathbf{w})\right)^2$$



Dueling Network 則是在選擇 action 時多加入了優勢的概念，如右上圖所示，conv2d 攤平後的神經元會被分成兩邊，左邊為 value network，右邊為 advantage network ，也就是選擇動作時把當下狀態的 state value 和 action value 都考慮在內，就像人的大腦一樣，不同的區域注意到不同部分，最後再綜合資訊來做判斷，因此可以加快學習效率，但在論文中也有做實驗，顯示當 action 數越多時，影響才會越顯著。

ii. compare and analyze the results with and without the improvements (1%)



實驗結果如上圖，發現加了 Double Q-Learning 和 Dueling Network 的影響好像不太顯著，不太知道是真的進步了還是只是 random 不同的差別，我想可能是因為兩者都是在選擇動作上進行改進，breakout 的 action 數只有 4，因此如 Dueling Network 論文所說，影響的效果可能就沒有那麼明顯，而 Double Q-Learning 的論文也有提到，這樣雖然能有效降低過度估計的問題，但可能有時也會導致 underestimation，也許會因此影響到學習效率，或是 Dueling Network 對於 state value 和 action value 的估計。

2. Implement advanced RL method, describe what it is and why it is better (2%)

我選擇在 pong 上實作 a3c(Asynchronous Advantage Actor-Critic)，a3c 是由 Google DeepMind 所提出，用來解決 Actor-Critic 不收斂的問題。主要方法是創建多個並行的環境，讓多個副 agent 同時來玩遊戲，再用一個中央 agent 來接收各副 agent 的參數更新，由於這些副 agent 玩遊戲時是彼此獨立的，所以中央 agent 接收到的參數更新就會有不連續性的干擾，每次更新的相關性被降低，收斂性便會提高，這是因為採用多個不同訓練環境採集的樣本，會使樣本的分佈更加平均，有利於模型訓練，平行訓練也提升訓練效率，中央 agent 可以綜合副 agent 的經驗來學習，如此也不需要 dqn 的記憶庫，可以節省儲存空間。

另外，a3c 也在選擇動作時加入優勢的概念，即某個動作 a 在某個狀態 s 下相對其他動作的優勢，透過計算當下狀態 s 的價值 V，來和每個動作 a 在當下狀態 s 的價值 Q 做比較，只有能增加當下狀態價值 V 的動作才有優勢，即應該要增加出現機率，否則即使動作價值 Q 為正，出現機率也應降低，透過這種方式計算損失函數也會讓 agent 學習更有效率。

以下是前 181 個 episode 兩者的 reward 趨勢圖，紅色是 30 期移動平均，可以看出 a3c 學習曲線成長快很多，在 ep23 就開始出現正的 5.0，一般 rl 則到 ep3423 才出現正的 2.0，且 a3c 約 40 個 ep 時移動平均就開始為正，雖然每一 ep 平均約 70.53 秒，比一般 rl 約 15 秒左右慢不少，但就 reward 開始為正的時間來看，a3c 半小時內就達到，一般 rl 則花了半天，所以學習效率真的高很多。（a3c 採用兩層 cnn，其他參數、環境和一般 rl 相同）