ID & Name：R06946009林庭宇、R06946006李筑真、R06946015黃永翰

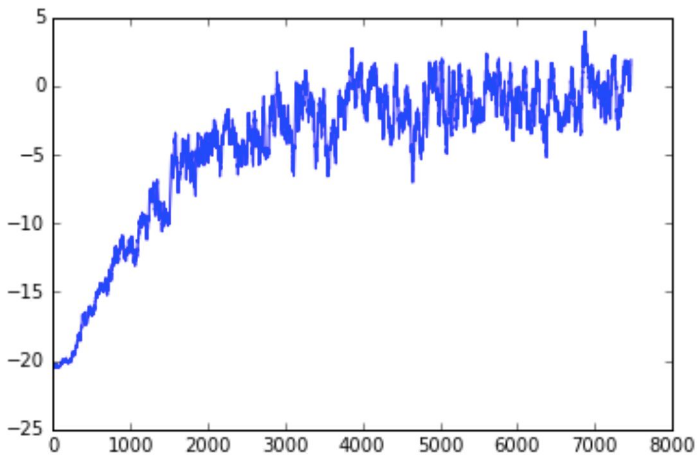
HW4-1 Policy Gradient

1. Describe your Policy Gradient model (1%)

我們將兩張相鄰observation的差作為模型的input(flatten成6400維)，目標是預測action(up or down)，model的結構為兩層fully connected layer(activation:Relu)後接output layer(activation:sigmoid)，neuron數依序為256 -> 128 -> 1．

model每和環境互動一次，即使用這些收集到的data進行training，更新一次參數，以reward作為weight，使在該observation下能產生高reward的action出現的機率增加，接著使用更新後的model繼續與環境互動，週而復始的training．

2. Plot the learning curve to show the performance of your Policy Gradient on Pong (1%)



3. Implement 1 improvement method：

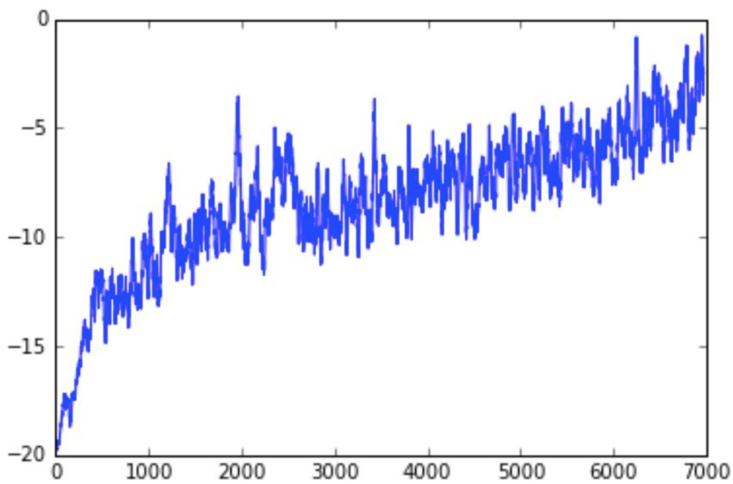
a. Describe your tips for improvement (1%)

我們選擇實作ppo2作為improvement的方式，利用important sampling的技術使我們的network與環境互動後所產生的data可以重複使用，對參數進行多次更新後，再讓模型與環境進行互動．

實際的loss如右：

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}A^{\theta^k}(s_t,a_t), clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right)A^{\theta^k}(s_t,a_t)\right)$$

b. Learning curve (1%)



c. Compare to the vallina policy gradient (1%)

由於時間有限，我們只train了7000個episode，和一般的policy gradient比，performence其實還略差，不過整體而言看起來上升的趨勢較為明顯(斜率較大)，說不定繼續train下去會有好結果．

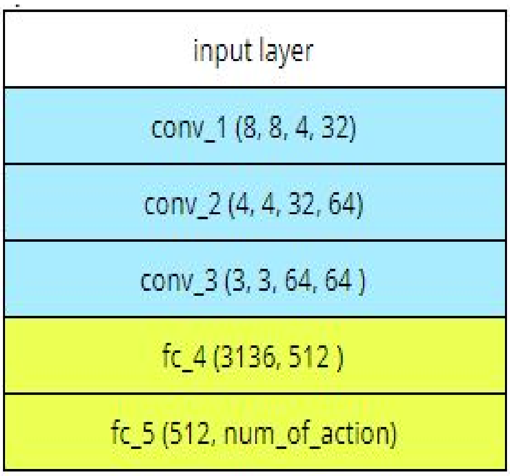
HW4-2 Deep Q Learning

1. Describe your DQN model (1%)

Parameters:

Replay Memory Size : 10000
Perform Update Current Network Step : 4
Perform Update Target Network Step : 10000
Learning Rate : 0.00025
Batch Size : 32, Gamma : 0.95
Final epsilon : 0.1, Initial epsilon : 1.0
Q Network Structure : 左圖

DQN algoritm : [https://arxiv.org/abs/1312.5602]

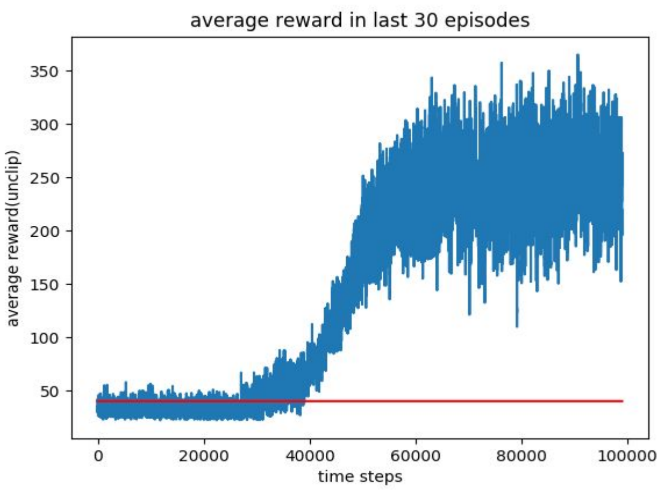


Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

每episode每個time step有epsilon的機率是隨機選擇action，另外1-epsilon的機率是選擇從Q network的輸出找值最大的action。然後收集執行這個action得到的 $(\phi_t, a_t, r_t, \phi_{t+1})$ 存入memory中。然後從memory取出minibatch，丟進Q network計算每個狀態的目標值，接著每4個time steps更新Q network，每10000個time step更新一次target Q network(copy weights from Q network)。

2. Plot the learning curve to show the performance of your Deep Q Learning on Breakout (1%)(unclip)



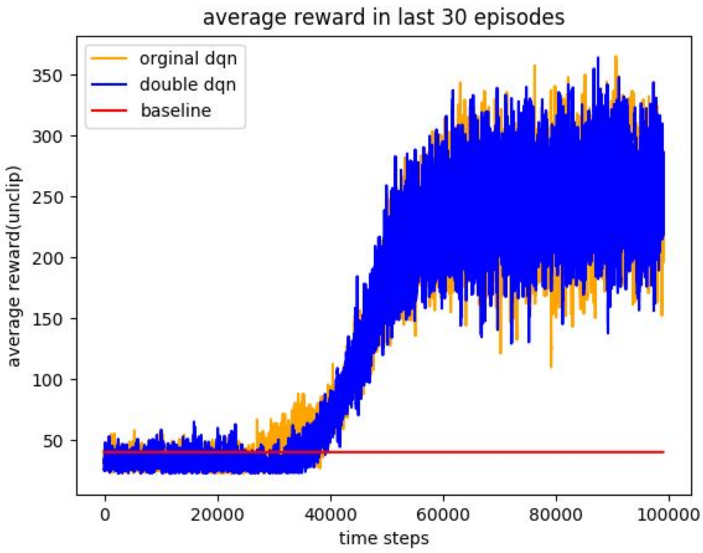
3. Implement 1 improvement method :

  a. Describe your tips for improvement (1%)

我選擇Double DQN，它可以避免每次都選到被高估的action。

執行方法只要將 $r_t + \max_a Q(s_{t+1}, a)$ 改成 $r_t + Q'\left(s_{t+1}, arg \max_a Q(s_{t+1}, a)\right)$。

b. Learning curve (1%) (unclip)



c. Compare to origin Deep Q Learning(1%)

從b.可以看到基本上使用double dqn(藍色)看起來與dqn(橘色)相差無幾，不會使表現增益得較快，但還是可觀察到它的穩定性比dqn好些，learning curve的震盪情形相對於dqn小一些。從老師slides第37頁可以看到扣除double dqn並不太影響其(彩虹模型)表現，故可以理解double dqn在performance上並沒有明顯助益。

HW4-3 Actor-Critic

1. Describe your actor-critic model on Pong and Breakout (2%)

pong

Actor

dense(200)

dense(2)

Critic

dense(200)

dense(1)

breakout

Actor

conv_1(8,8,4,32)

conv_2(4,4,32,64)

conv_3(3,3,64,64)

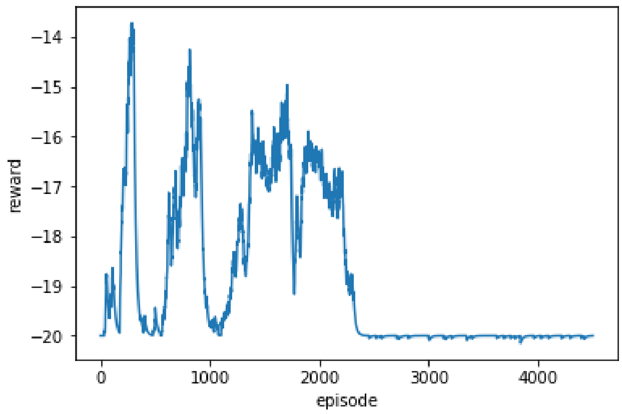
fc_4(3136,512)
fc_5(512,num_of_action)

Critic

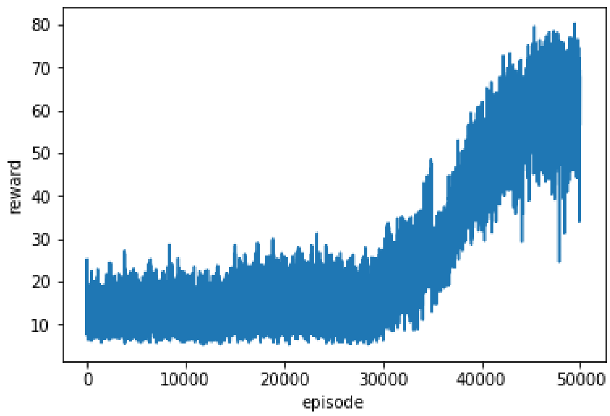
conv_1(8,8,4,32)
conv_2(4,4,32,64)
conv_3(3,3,64,64)
fc_4(3136,128)
fc_5(128,1)

2. Plot the learning curve and compare with 4-1 and 4-2 to show the performance of your actor-critic model on Pong & Breakout (2%)

pong (unclip)



breakout (unclip)



pong的部分訓練不太起來，到某個時間點後reward幾乎維持在最低分，有可能是Actor或Critic的結構沒有設好，但Actor Critic的訓練時間比較久，測了幾個都沒什麼好轉，breakout的部分則是有辦法訓練起來，但收斂速度慢且效果沒那麼好。

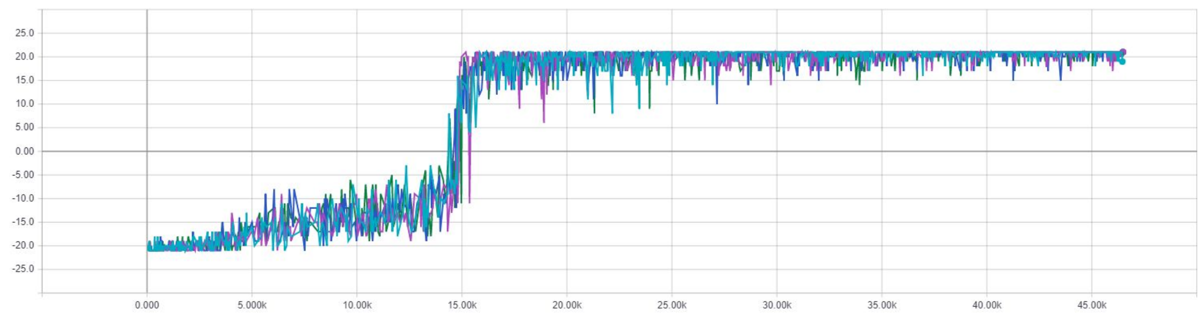
3. Implement 1 improvement method :Reproduce 1 improvement method of actor-critic (Allow any resource)

a. Describe the method (1%)

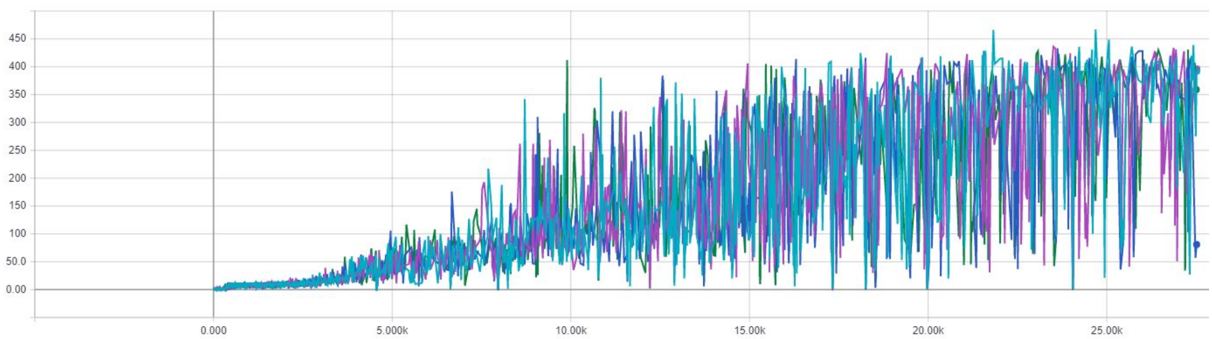
使用Synchronous Advantage Actor Critic (A2C)，不同於A3C每個Agent會自己更新全局網絡，因此在某個時間，Agent使用的權重可能與另一個Agent使用的權重不同，而在Synchronous Advantage Actor Critic中，Agent的所有更新都將被收集後再更新到全局網絡。

b. Plot the learning curve and compare with 4-1 and 4-2, 4-3 to show the performance of your improvement (1%)

pong (unclip)



breakout (unclip)



A2C的方法在pong中比4-1和最簡單的Actor Critic效果好，在第15000個episode後幾乎都能拿到接近滿分，在breakout中效果也不錯，但是收斂速度比4-2還要慢，我想應該是需要訓練兩個網路，使得對資料量的要求較大，需要玩比較多 episode，但比最簡單的Actor Critic收斂快，應該是因為有四個agent一起訓練。