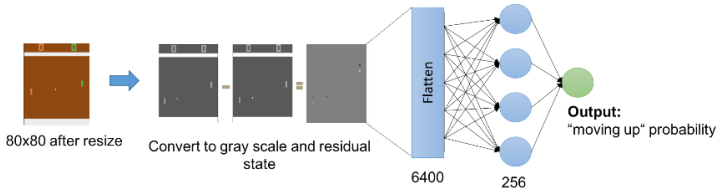
MLDS HW4 Report

楊其昇 B03901101 林宣竹 B03901065 郭恆成 B03901145

4-1

1. Describe your Policy Gradient model

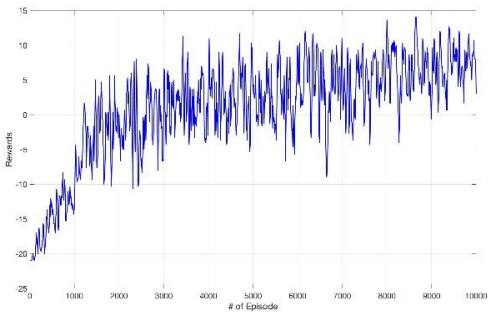
   由於是要教機器玩 Pong 這個遊戲，而這個遊戲的動作就只有"往上"或"往下"（捨棄維持現狀的選項），因此如下圖所示，模型本身架構十分簡單：輸入的遊戲畫面首先被 resize 成 80x80，再轉成灰階，並最後以 Residual state 的方式輸入模型，經過 flatten 後，直接送入只有一層 256 neurons 的 hidden layer (activation: Relu)，再加上一層只有 1 個 neuron 的 output layer (activation: sigmoid)， 得到"往上"的機率 P ("往下"的機率即為 1-P）。



   訓練過程中，使用 Adam optimizer，其中 learning rate=1e-4，更新參數的時機為一場 21 分的比賽結束時（定義為一個 episode）。另外有對每個 episode 的 reward 增加 discount factor 的考量後，進行 normalization。在測試時，則一律以輸出機率最高的選項作為機器決定下一刻要做的動作。

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

   以下為訓練 10000 個 episodes 的過程：



3. Implement 1 improvement method

   選擇 Proximal Policy Optimization 中的 clip 版本 (PPO2)，作為實驗對象。

   a. Describe your tips for improvement

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

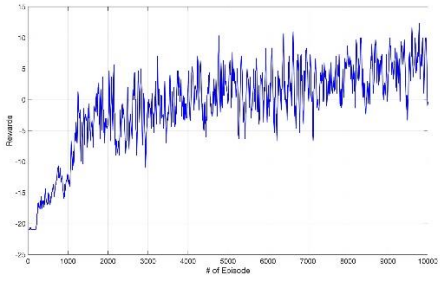
end for

參考作業說明投影片，PPO2 的演算法如上，batch size = 4，K = 8。
考慮實作上通常為 minimize loss 以及原始 paper 的內容，因此實際上

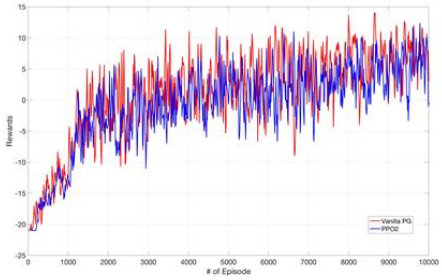
$$\text{Loss} = \mathrm{E}_{(s_t,\, a_t)\sim\pi_k}[\max\left(-\hat{\mathrm{A}}_t^{\pi_k} \times \frac{p_\theta(a_t|s_t)}{p_{\theta_k}(a_t|s_t)}, -\hat{\mathrm{A}}_t^{\pi_k} \times clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\right] - c_1 \times \textbf{cross entorpy}$$

其中 c1 = 0.01、ϵ = 0.2 皆為原始 paper 中所提到的參數，cross entropy 中的 label 為先前 episode 中，θ k 隨機 sample 出的 action，prediction 則為 θ 所預測相對應的機率。
由於原始 paper 中的 model 除了決定每個 action 的機率外，還有預測 state value（一種 variance reduction 的技巧），但此次作業中，我們並沒有實作此功能，因此 loss function 和原始 paper 相比，變少了一項 state value function 的 loss。另外，實作時的模型架構皆和第一題所述相同。

b. Learning curve



c. Compare to the vanilla policy gradient
下圖為標準的 Policy Gradient 和 PPO2 之間 learning curve 的比較：



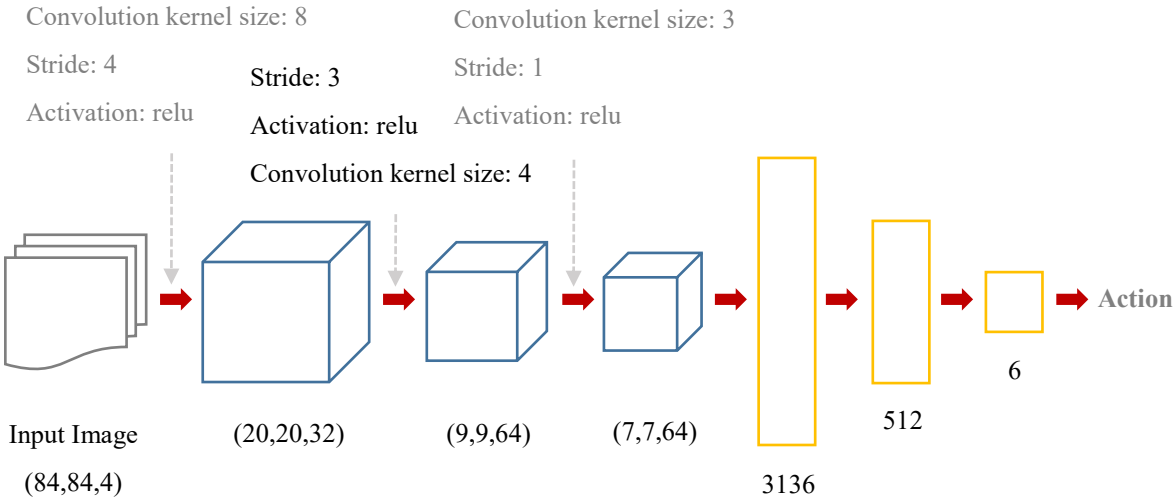
可以發現 PPO2 的學習過程比標準的 Policy Gradient 要穩定得多，推測或許是因為更新參數時多考慮了幾個 episodes 再去更新。另外在實作時發現 PPO2 對於 learning rate 等參數相較於標準的 Policy Gradient 較敏感，learning rate 不能太高，也不能太低否則會學很慢，參考了 OpenAI 的實作後，最後 PPO2 使用 Adam optimizer，learning rate = 2e-5，epsilon = 1e-5，此外更新參數時有做 Gradient norm clipping（上限為 0.5）。雖然使用 PPO2 效果未必會比標準的 Policy Gradient 來得好，但 PPO2 卻給了一個保證比較穩定的訓練過程。雖然 PPO2 有比較穩定的訓練過程，但

實作中其實學習速度的快慢還滿靠運氣的，推測或許是我們實作的 PPO2 相較於原始 paper 少了一項和 state-value 有關的 loss 才會導致無法有效降低 variance，使得每次訓練時都有一點運氣的成份在裡面。

4-2

1. Describe my DQN model

   本次作業希望透過 Deep Q Learning 教機器學會玩 Breakout 這個遊戲。我們建立的模型架構如下：

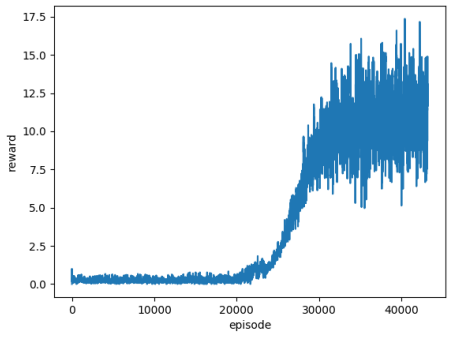


   將大小 84*84 且 4 個 channel 的遊戲畫面作為 input，通過三層 convolution layer，kernel size 依次為 8, 4, 3，stride 則為 4, 3, 1，然後將其結果 flatten，再通過兩層 dense layer，unit 數分別為 512 和 4，最後選擇 Q value 最大者做為我們要執行的 action 選項。

   訓練過程中，使用 RMSProp Optimizer，momentum, epsilon 和 decay 的值分別設 0, 10^{-6} 和 0.99，learning rate 則為 0.00025，另外，做了 gradient clipping 的動作，將其限制在±1之間。

   此外，運用了遞減的ε值來達到增加隨機性的效果。在最初 50000 個步中，或是當一隨機值小於ε時，模型會隨機執行 action，而這個ε值會從訓練執行 50000 步後，遞減至 0.1（過程中經過 10^6 步），此過程為 exploration 階段。

2. Plot the learning curve to show the performance of my Deep Q Learning on Breakout



Average reward 從 20000 個 episode 之後開始上升，一直成長到 12 左右停止。Unclipped reward 的測試結果為，一百次遊戲中平均得到分數 74.65。

3. Implement 1 improvement method on page 6

(1) Describe your tips for improvement

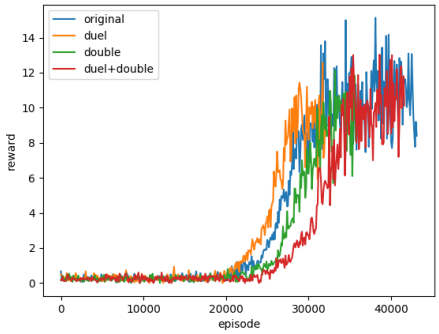
A. Double DQN

原本的 DQN 會有 overestimate 的問題，double DQN 利用 DQN 中的兩個神經網路 eval_network 和 target_network，消除這誤差的影響。實作上，利用 eval_network 估計 target_network 中的 $Q_{max}(s', a')$，然後用此估計出來最大值的動作來選擇 target_network 中的 Q(s')。

B. Dueling Network

Dueling DQN 比起原本的方法，更能判斷各個動作對下一步的影響，進一步得到更好的結果。實作上，在最後一層 layer 分成兩部分，專門分析 state 的 value (V) 與分析每種動作的 advantage (A)，兩者套用 Q = V(s) + A(s, a)的公式，得到最後的 Q 值。

(2) Learning curve



四條曲線分別是原本的 DQN、duel DQN、double DQN 和 duel+double DQN，可以看出四者趨勢相近，average reward 最後同樣在 10~12 之間擺盪，但 duel DQN 比起其他三者，在更短的時間內學會如何玩遊戲。

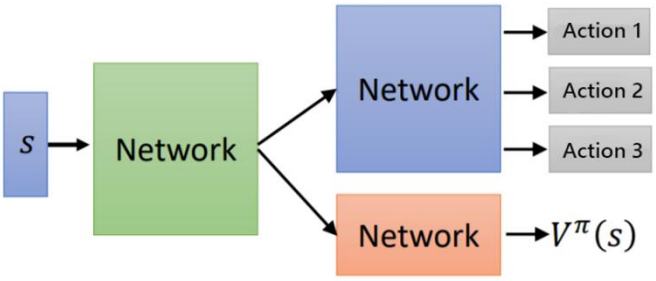
(3) Compare to origin Deep Q Learning

original	duel	double	duel+double
74.65 (episode:43000)	75.79 (episode:31000)	66.92 (episode:34500)	55.42 (episode:38000)

上表中的 reward 是選附近 episode 中結果最好的，很意外地只有 duel DQN 的分數大於原本的 DQN，double 和 duel+double DQN 的分數反而比較低，可能是因為 train 的時間不夠久，但由於我們 CPU 不夠好，thread 開的不夠多，無法訓練出夠好的模型，所以我們最後採用 duel DQN 在 31000 episode 時的 model 作為本次實驗的最終檔案。

4-3

1. Describe your actor-critic model on Pong and Breakout

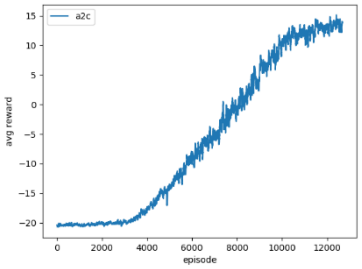


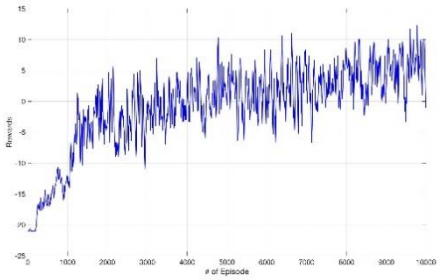
基本的 actor-critic model 採用 advantage actor-critic (A2C) 的架構。首先進來的 image 會先經過除以 255.的 scaling，在通過 num_fliter 分別為 32, 64, 64, 512 的 CNN，抽取 image 的 feature。最後分別經過 unit 別為 512, 1 的 DNN 當作 value function。以及相似架構的的 actor 不過 output layer 的 unit 變成兩個 case 對應的 action 數目。訓練過程使用 RMSProp optimizer、Temporal-difference (TD) 的方法以及 discount factor=0.99。

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

A2C 4-1

Pong



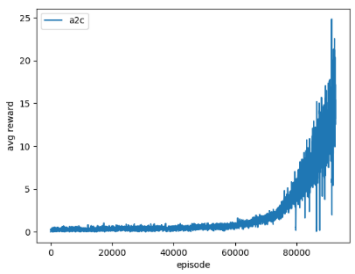


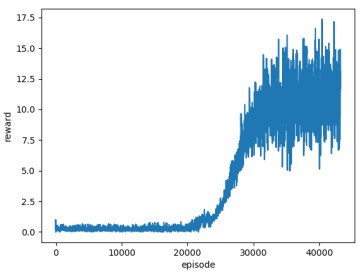
在 Pong 的 case 中可以看到比起 4-1，A2C 的方法很明顯的 reward 震盪幅度較小，30 個 episode 間的 reward 都較平均。雖然在 10000 個 episode 時兩者看起來都約落在 7-9 之間，A2C 僅是略高於 policy gradient，但 a2c 的 training curve 看起來還在穩定上升，policy gradient 則否。

另外 A2C 在訓練一開始時上升的較慢，前 3000 個 episode 都在-19 ~ -21 之間。不過這不排除在 4-1 的時候僅用了往上、往下兩個 action，而訓練 A2C 的時候則保留了”不動”這個 action，所以可能在初期較不容易訓練。

A2C 4-2

breakout

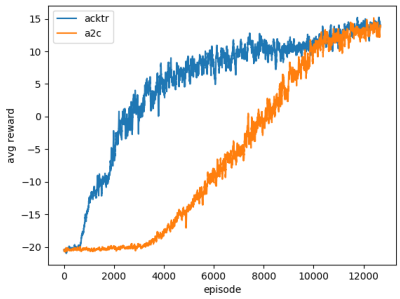




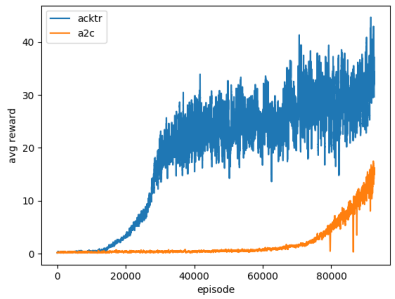
在 Breakout 的 case 中也能觀察到類似的狀況，A2C 在訓練過程中，震盪的幅度比 DQN 小，但是在訓練一開始的時候上升的較慢，但因為 DQN 在訓練的時候有使用 reward clip 的方法，A2C 則否，所以兩者在訓練過程上，無法看出兩者 performance 差異。但在 4-2 Unclipped reward 的測試結果中，一百次遊戲中平均得到分數 74.65，遠比 A2C 在訓練過程中得到的平均還高，所以可以猜測 A2C 的方法訓練較慢，在 4-3 的實作中還未收斂。

3. Plot the learning curve and compare with 4-1, 4-2 and 4-3 to show the performance of your improvement.
   a. Describe the method
      使用 Actor Critic using Kronecker-factored Trust Region (acktr) 當作 improvement method。Acktr 實作中，基本的 actor-critic model 使用 4-3.1 的 a2c model，加上 trust region optimization (TROP) 來保證穩定度，同時加上 Kronecker factorization 提升 sample 效率。
   b. Plot the learning curve and compare with 4-1, 4-2 and 4-3 to show the performance of your improvement.

pong



breakout



在 Pong 的部分可以看出來 A2C 與 ACKTR 的方法訓練到最後的 reward 相近，都比 policy gradient 的方法快。在訓練過程中 ACKTR 的 reward 上升的比 A2C 快，相對好訓練，但是 ACKTR 的 training curve 震盪幅度較大，不過在 Pong 的 case 中影響不大，兩者的表現都比 4-1 好。

在 Breakout 的 case 中也看到在訓練一開始 ACKTR 進步的速度，遠快於 A2C，在 15000 個 episode 左右 reward 已經不再幾乎都是 0。但在訓練速度較快的優點放大的同時，訓練過程中震盪較大的缺點也很明顯。而與 4-2 的 DQN 比較，兩者的 training curve 滿相似的，但 ACKTR 的結果仍低於 DQN unclipped reward 的數值。另外在 reakout 這個 case 中不論是 ACKTR 還是 A2C 的方法，很明顯的將近 100000 個 episode 都還沒收斂，不能比較最後兩者 reward 的差別，有很大的進步空間。但礙於訓練 Breakout 所需的時間遠大於 Pong 難以訓練到收斂。

4. 分工表

楊其昇 B03901101: 4-1 林宣竹 B03901065: 4-2 郭恆成 B03901145: 4-3