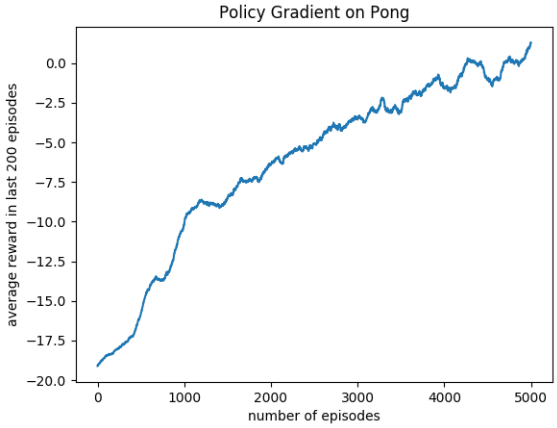
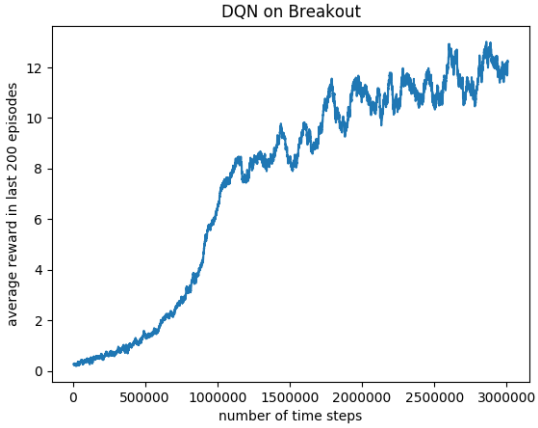
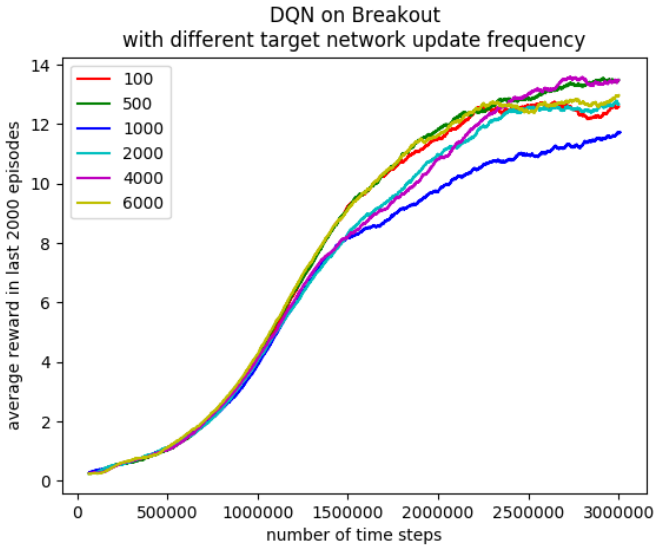
# HW3 Report b05902052 劉家維

- Basic Performance (6%)
  - Describe your Policy Gradient & DQN model (1% + 1%)
    - Policy Gradient:
      - Input: (batch_size, 1, 80, 80) 的影像
      - 二維卷積層：濾鏡數=16，濾鏡大小=(8, 8)，卷積步長=4，activation=ReLU
      - 二維卷積層：濾鏡數=32，濾鏡大小=(4, 4)，卷積步長=2，activation=ReLU
      - 平坦層
      - 全連接層：輸出維度=128，activation=ReLU
      - 全連接層：輸出維度=3，activation=Softmax
      - 優化器：RMSprop (學習率=1e-4， 衰減率=0.99)
      - batch_size=一整個 episode，gamma=0.99
    - DQN model:
      - Input: (batch_size, 4, 84, 84) 的影像
      - 二維卷積層：濾鏡數=32，濾鏡大小=(8, 8)，卷積步長=4，activation=ReLU
      - 二維卷積層：濾鏡數=64，濾鏡大小=(4, 4)，卷積步長=2，activation=ReLU
      - 二維卷積層：濾鏡數=64，濾鏡大小=(3, 3)，卷積步長=1，activation=ReLU
      - 平坦層
      - 全連接層：輸出維度=512，activation=LeakyReLU
      - 全連接層：輸出維度=4
      - 優化器：RMSprop (學習率=1e-4, 衰減率=0.99, loss_function=MSELoss)
      - batch_size=32，gamma=0.99， online network 更新頻率=4，target network 更新頻率=1000，歷史回放大小=10000 states，探索率=前 100 萬步由 1 線性降至 0.05、之後保持 0.05
  - Plot the learning curve to show the performance of your Policy Gradient on Pong (2%)
  - Plot the learning curve to show the performance of your DQN on Breakout (2%)



- Experimenting with DQN hyperparameters (4%)
  - Plot all four learning curves in the same graph (2%)




  - Explain why you choose this hyperparameter and how it effect the results (2%)
    - 因為 DQN 在訓練時有兩個網路，若 target network 更新過快會造成很難訓練（訓練的目標快速變動），更新過慢可能會造成訓練速度較慢。為了證實這個想法，因此我針對 target network update frequency 進行訓練。但是實驗結果卻出乎意料，反而更新速度比 1000 快的 (100 或 500) 和比 1000 慢的 (2000, 4000, 6000) 都還比 1000 好。跟其他人討論過後，發現他的 1000 和 10000 也差不多，這部份可能是實驗誤差。而 100 和 500 較 1000 好的原因我猜測是因為這樣能夠讓訓練速度加快（觀察 100 和 500 的學習曲線即可得知），這次實驗沒有挑到 update frequency 頻繁到訓練不起來的數值（跟其他人討論有人 frequency = 4，結果完全訓練不起來）。不過為什麼 6000 中途也學的跟 100 和 500 一樣快我無從得知。
- Improvements to DQN (2%)
  - Implement at least two improvements to DQN and describe why they can improve the performance (1%)
    - Dueling Network
      - 原本的 DQN 在訓練時，我們會認真計算每組 Q(s, a) 。也就是說對於每個 state 我們都會去計算當下每個 action 的 Q value。但是這其實並不是非常有必要，因為對於某些 state ，每個 action 之間其實差距不大。因此 Dueling Network 計算兩個值，V(s) 和 A(s, a) ，分別是 state 的價值和當下做了 action 之後能夠得到的 advantage，並且定義 Q(s, a) = V(s) + A(s, a)。值得注意的是實作時必須將 A(s, a) 的平均調整為 0，避免神經網路直接學到整個 Q value。
    - Double DQN

    - 原本的 DQN 在訓練時，我們用 target network 預測的值 Qmax 會有誤差，而我們每次更新卻又是拿那個有誤差的 Qmax 繼續更新 online network 的 Q 值，導致高估 Q value。因此，Double DQN 只是把 target network 中 Q 值最大的 action 挑出來，而他的 Qmax 卻是拿 online network 自己的 Q 值去計算（target network 只負責挑出最好的 action ，至於最好的 action 真正的 Q 值是多少則是用 online network 去計算）。
- Plot a graph to compare and analyze the results with and without the improvements (1%)