機器學習及其深度化與結構化 **HW4**

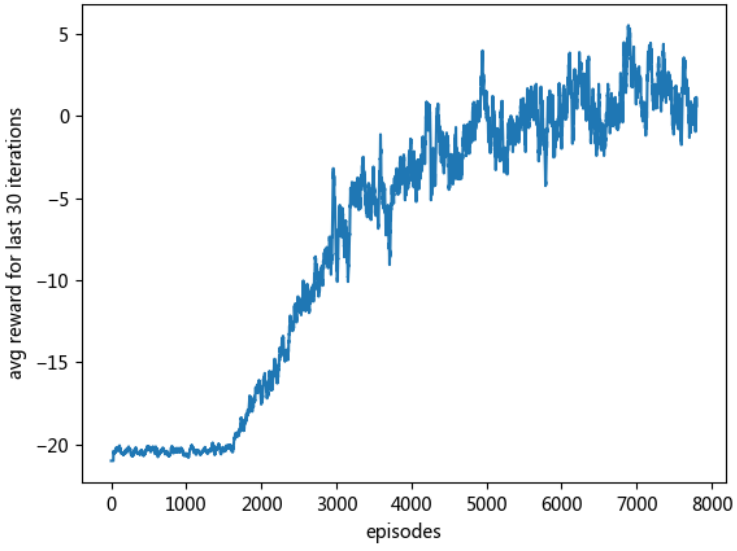
吳政軒 R06922118 4-1, 4-3, report	黃敬庭 R06944049 4-3, report	馬欣婕 R06946010 4-2 report

4-1

- Describe your Policy Gradient model (1%)

  policy gradiet 的更新模式基本上和投影片裡面講的一樣。對讀進來的圖片預處理是將上面的記分板的部份直接cut掉，然後將圖片轉為 80 * 80 * 1，然後將非背景的地方社為黑色，其他定為白色（參考openai實做方法） 。用來訓練的圖片是取每一個timestep與上個timestep圖片的差。DNN的架構為2層DNN，中間的hidden layer維度為256，最後一層過sigmoid，作為向上的機率。使用的optimizer為adam並將lr rate設為0.003。

- Plot the learning curve to show the performance of your Policy Gradient on Pong (1%)
  - X-axis: number of time steps
  - Y-axis: average reward in last 30 episodes

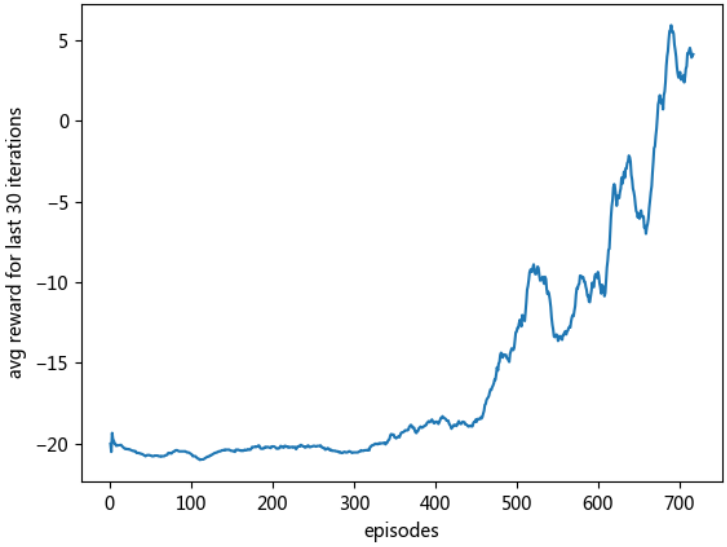


  可以看到大約到5000多iterations的時候可以超過baseline，而我們又讓他繼續training之後會慢慢上升但還是會有蠻大的振幅。在最後我們把它停止住之後，在testing的環境下可以超過baseline。

- Implement 1 improvement method on page 8
  - Describe your tips for improvement (1%)

我們使用的ppo在架構上一樣為2層DNN，並且在維度上都與第一小題的設定相同，並且參考莫凡的教學利用4個agent一起去更新，各自收集數據之後再丟到global的network去，train到穩定超過baseline為止。ppo的方式選擇的是用clip objective 的，epsilon設為0.2。使用的optimizer為adam並將lr rate設為0.001。

- Learning curve (1%)



- Compare to the vallina policy gradient (1%)

大約在600多個episodes的時候就可以穩定超過baseline。與原本最簡易的policy gradient相比，ppo的訓練時間明顯少很多。

4-2

- Describe your DQN model (1%)

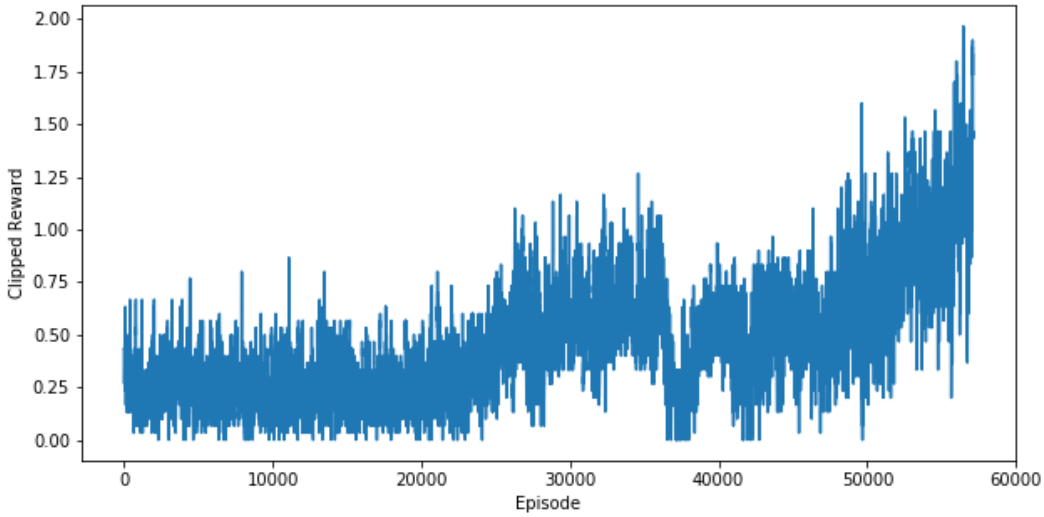
我們使用的是 CNN，總共有三層的 conv layer 跟兩層的 fc layer，activation function 使用的都是 relu。詳細如下：

```
DQN(
  (conv1): Conv2d(4, 32, kernel_size=(8, 8), stride=(4, 4))
  (conv2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
  (conv3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=3136, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=4, bias=True)
  (relu): ReLU()
)
```

訓練使用的 loss function 為從 target network 得到的 r + gamma * Q(s, a') 以及從不斷更新的 evaluation network 得到的 Q(s, a) 算出的 MSE。每1000個 step 會把 eval_net 的參數複製到 target_net 來。

我們使用的 Gamma 值為 0.99，ReplayMemory 的大小是 500,000，epsilon 在 50000個 steps 內由 1 線性降低到 0.02，總共訓練接近 60000 個 episodes。

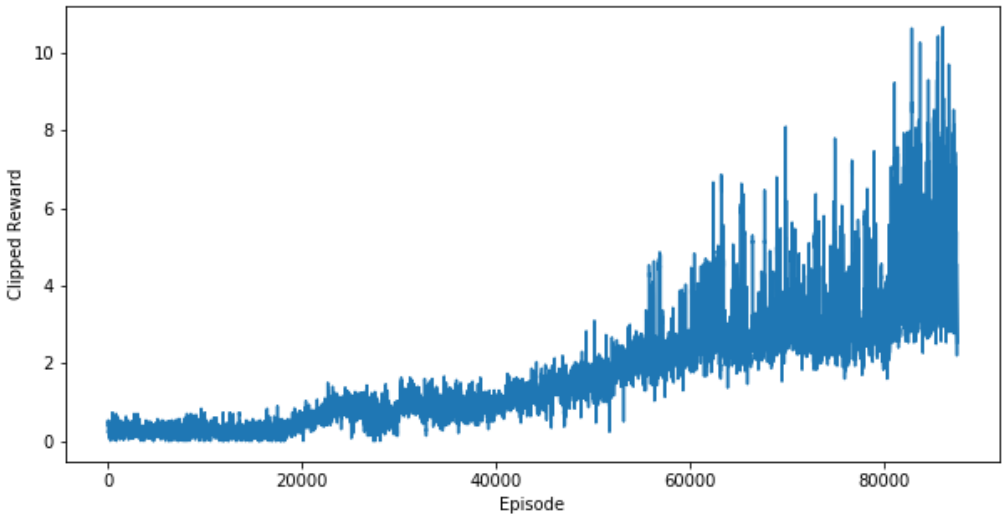
- Plot the learning curve to show the performance of your Deep Q Learning on Breakout (1%)
  - X-axis: number of time steps
  - Y-axis: average reward in last 30 episodes



- Implement 1 improvement method on page 6
  - Describe your tips for improvement (1%)

    我們選擇的 improvement 方法是 Dueling DQN，模型的架構跟原本的 Natural DQN 一樣，但不是輸出各個動作對應的Q值，而是分成 value 跟 advantage 兩部份，value 代表的是各個 action 共有的部份，advantage 則表示真正由該 action 貢獻的部份，為了讓差異突顯，有減去各 action 得到的 advantage 平均值。
    一樣使用 Gamma 值 0.99，ReplayMemory 500000，從1到0.02線性降低的 epsilon，訓練接近 100000 個 episodes。

  - Learning curve (1%)



- Compare to origin Deep Q Learning(1%)

  在這次作業中在 Natural DQN 的部份分數上升得很慢，而 Dueling DQN 在五萬個 episodes 之後開始有明顯上升，超過 80000 個 episodes 之後 clipped reward 可以達到 10 分。可能在 natural DQN 的部份還需要更久更久的時間。

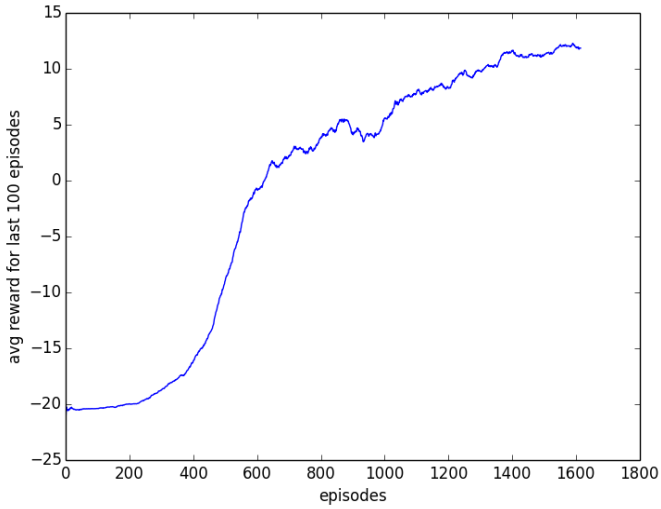
4-3

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

  訓練方式和架構就和老師投影片中的一樣，我們的share part是兩層Convolution加一層Dense，actor network是直接將share part的output通過一層Dense，取softmax決定action probability，而value network則是通過一層Dense後output一個value。Pong會先將state preprocess成[80,80]的shape，Breakout則是直接用[84,84,4]的shape。

- 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%)
  - X-axis: number of time steps
  - Y-axis: average reward in last 100 episodes

Learning curve (pong)



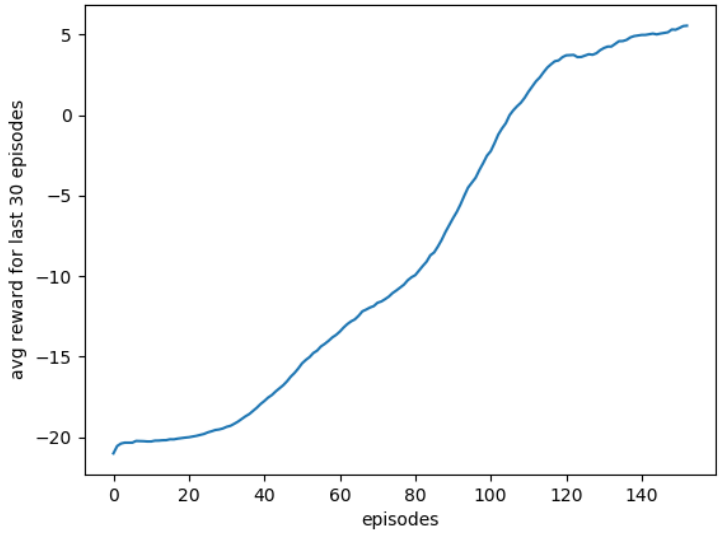
和4-1相比，訓練效果明顯提升許多，在200 episode左右就開始往上爬升，1000 episode大概就可以通過baseline了，相反的，4-1單純的policy gradient在1000個episode時根本就還沒開始往上爬。

- Reproduce 1 improvement method of actor-critic (Allow any resource)
  - Describe the method (1%)

    我們使用A3C (Asynchronous Advantage Actor-Critic) 的架構，和原始的policy gradient 相比多了advantage、value based的方法，state先經過兩層Convolution，Flatten後再通過一層Dense輸出一個256維的vector，再分別送進actor network 和 value network，然後用RMSPropOptimizer training，我們開了12個thread加速訓練，只要140 episode就超過5了。

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

    Learning curve (pong)



由圖中可以發現A3C又比原始的actor-critic快速許多，只要150 episode左右平均reward就可以超過5了。