MLDS Hw4- Reinforcement Learning

R05229014 鄒適文

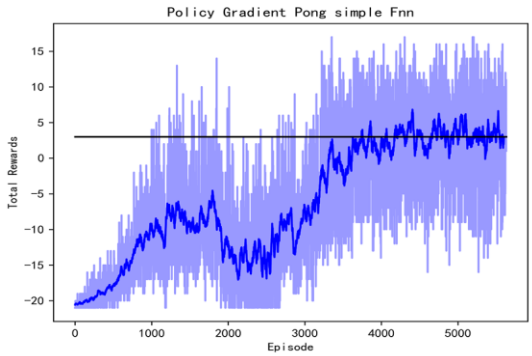
R05229016 羅章碩

HW4-1:

1. Describe your Policy Gradient model (1%)

我使用的 model 是兩層全連接層的 model，其 units 分別為 512、256，最後接 softmax 並輸出三個動作的機率(分別是上、下、不動)，使用的 optimizer 為 Adam 剛開始的學習率設為 0.001 之後訓練到兩千多個 Episode 時，將學習率調成 1e-4，選 action 的方式剛開始是依據每個動作的機率來進行 sample，訓練到後期改成輸出最大機率的動作。在訓練初期時使用了使用自己設計的 reward，只要橫桿在球來的時候如果有移動到球的附近接球，就會得到一些分數，增加訓練初期的速度。

2. Plot the learning curve. (1%)



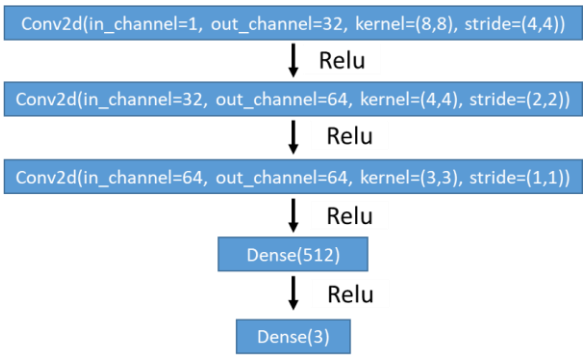
此圖中淺藍色為每一個 Episode 的分數，而深藍色則為 30 個 Episode 的平均分數。而圖中黑色水平線為 baseline 分數。

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

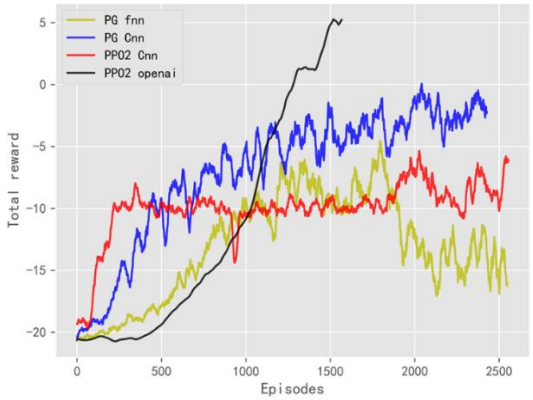
這邊我使用的是 PPO2 的技巧，將目標函數改為下列式子：

$$L^{CLIP}(\theta) = \hat{E}[\min(r_t(\theta)\widehat{A_t}, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A_t}]$$

使得原本 on policy 的訓練方式轉變成 off policy，讓收集完一次 Episode 的資料後可以被用來訓練 model 很多次(在這邊是 10 次~20 次)。在這邊是使用 cnn 的模型架構來比較，訓練一個用一般 policy gradient 的模型和一個用 ppo2 的模型來加以比較，其 cnn 架構是以 openai 中的架構為基礎(可以簡單地和 openai 做個小比較)。



4. Learning curve (1%)



上圖為每 30 個 episode 平均的結果，紅線為 PG cnn、藍線 PPO2 cnn、黃線 PG FNN，而黑色線是用 openai baseline 中的 ppo2 來訓練的結果，其代表每 update 一次參數後的結果而不是一個 episode 的結果。

5. Compare to the vanilla policy gradient (1%)

比較紅、藍色線的結果可以發現使用 PPO2 的方法 Total reward 上升的非常快速(前 200 個 episodes)，不過玩到了 300 多 Episode 的時候分數卻上不去了，反而是單純使用 PG 的有繼續上升，後來 tune 了非常久都還是一直沒甚麼效果，分數依然卡在那，後來我看了一下 PPO2 訓練出來的 model 玩遊戲的情形，發現橫桿會接球的時候大部分都是讓對手失分的時候，如果接到球後不會讓對手失分那 model 就可能會漏接那球。看到此現象後，我嘗試改了一下 reward 的情形，使用第一小題提到的方法，並增加自己失分時的扣分數，這樣訓練下去後發現 model 似乎有變好的情形，不過因為時間的關係所以來不及將 model 訓練好並寫在 report 中。除此之外，我也有去嘗試一下 openai 中 baseline 的 ppo2，發現雖然剛開始訓練分數的上升沒有很顯著但到了後面卻上升的非常快速，跟我訓練出來的模型剛開始上升快後面趨緩的情形有差別。

確認後發現 openai 中 ppo2 的算法(如下)還有加入 entropy 跟 squared-error loss，之後或許有機會可以嘗試看看。

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

HW4-2

1. Describe your DQN model (1%)

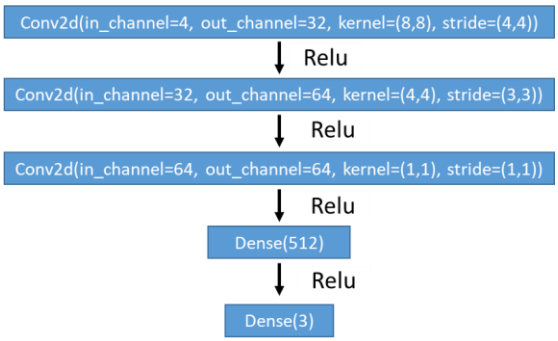
Simple DQN : 架構幾乎參考 DQN paper

Opt : RMSprop(lr=0.00025, moment:0.5)、Gamma : 0.99、

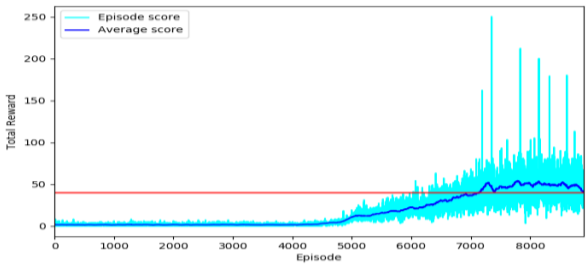
Memory size : 300000、Batch size : 32、Target model update : 10000 steps、

Policy model update : 4、

Exploration : 1 to 0.005 in 1000000 frames (after memory full)



2. Plot the learning curve to show the performance of your DQN on Breakout (1%) **REWARD IS CLIPPED**

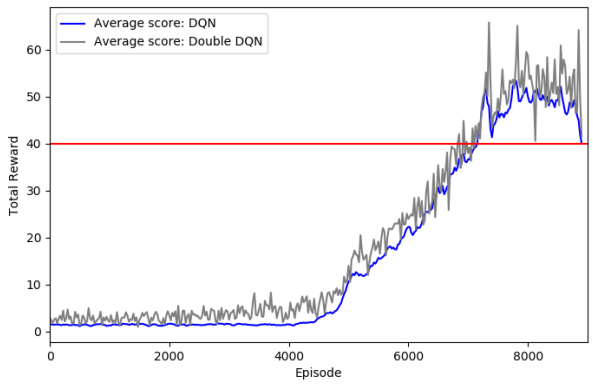


3. Implement 1 improvement method on page 5. Describe your tips for improvement (1%)

Double DQN :

原本的 DQN target model 的 output 直接選擇 Q 值最大的動作去 optimize，而 Double DQN 則是選擇使用 Training 的 output 動作決定要 optimize target model 的哪一個 output，而這好處在於，實際選擇的動作跟所獲得的 reward 比較接近，可能可以得到比較好的結果，理論上是可以解決 Q value overestimate 的問題。

4. Learning curve (1%)　　REWARD IS CLIPPED



5. Compare to origin DQN (1%)

可以看到 Double DQN 表現確實比原本的 DQN 還要好，並且同時我發現 DQN 在 train 的時候常常自己玩一玩就死掉，而這是我取訓練中最好的結果，並且 fine tune 過的結果，卻似乎簡單的被 Double DQN 超越了。

HW 4-3:

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

Pong:

Actor model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 6400)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               3277312
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 771
=================================================================
Total params: 3,409,411
Trainable params: 3,409,411
Non-trainable params: 0
_________________________________________________________________
```

Critic model:

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            (None, 6400)         0
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 512)          3277312     input_3[0][0]
__________________________________________________________________________________________________
input_4 (InputLayer)            (None, 3)            0
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 256)          131328      dense_7[0][0]
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 256)          1024        input_4[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 256)          0           dense_8[0][0]
                                                                 dense_9[0][0]
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 64)           16448       add_1[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 1)            65          dense_10[0][0]
==================================================================================================
Total params: 3,426,177
Trainable params: 3,426,177
Non-trainable params: 0
__________________________________________________________________________________________________
```

Breakout:

A3C:

Breakout : 使用的模式架構如下

Conv2d(16, (8,8), stride=4)　gamma=0.99

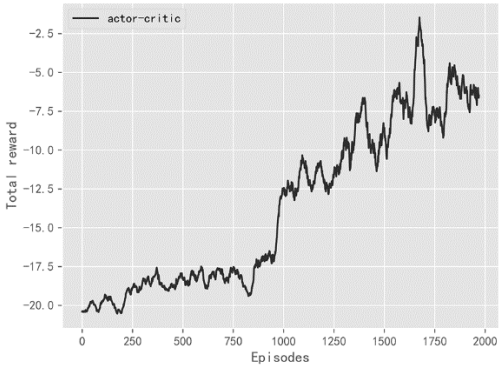
Conv2d(32, (4,4), stride=2)　actor_lr = critic_lr = 2.5e-4

Dense(256)　threads=12

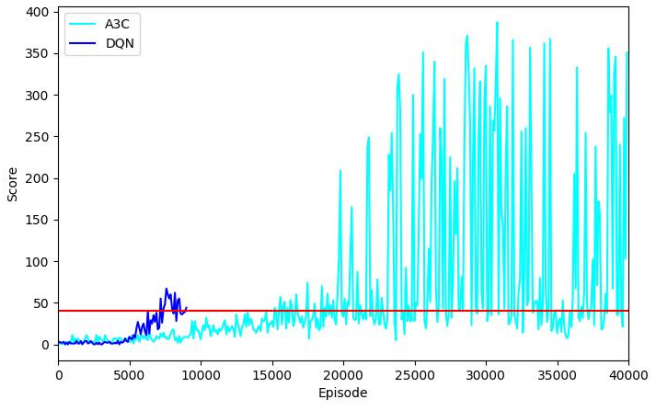
Dense(1)　opt = Adam

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:



Breakout:　**REWARD IS CLIPPED**



3. Reproduce 1 improvement method of actor-critic. Describe the method(1%)

Pong 和 Breakout 皆使用 A3c:

Actor-Critic 是將 Value based 跟 Policy based 結合的作法，與原本的 PG 在於多了一個 critic network 評判行為之得分，而 Actor 根據 Critic 的評分修改選行為的機

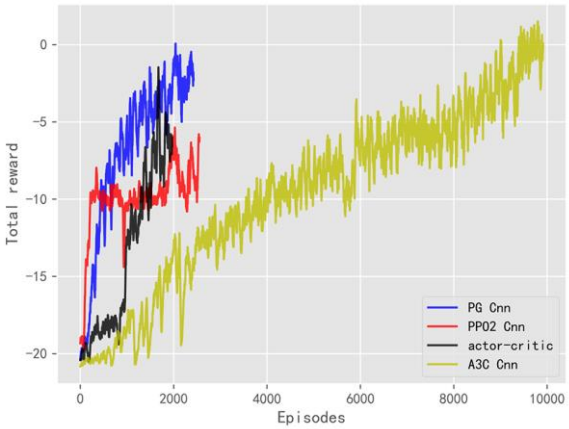
率，而 A3c，就像平行宇宙、鳴人的多重影分身一樣，使用多核的電腦頑童一個遊戲，匯集各種經到同一個本體中，也同時從其獲取最新的玩遊戲方法。

然後我覺得可能使用 DQN 或者 DDQN 或許表現也可以達到與 a3c 差不多的表現，但訓練時間實在是太久了，我使用 DQN 花了兩天半才跑到 10000 個 episode，而 A3C 僅花 12 小時就已經跑到 40000 個 episode 了，效率上實在是遠勝 DQN

Pong 使用的 A3C 模型架構與 4-1 使用的 CNN 模型類似，在訓練 Pong 的過程中同樣如上述所說，其訓練效率很高，同時能達到與使用 PG 方式訓練的模型同樣的 performance，平行訓練的方式其實也可以用在 PPO2 或 PG 的訓練上，增加訓練速度。

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

4-1 vs 4-3



4-2 DQN vs a3c **REWARD IS CLIPPED**