MLDS HW3 Report

R04943151 梁可擎

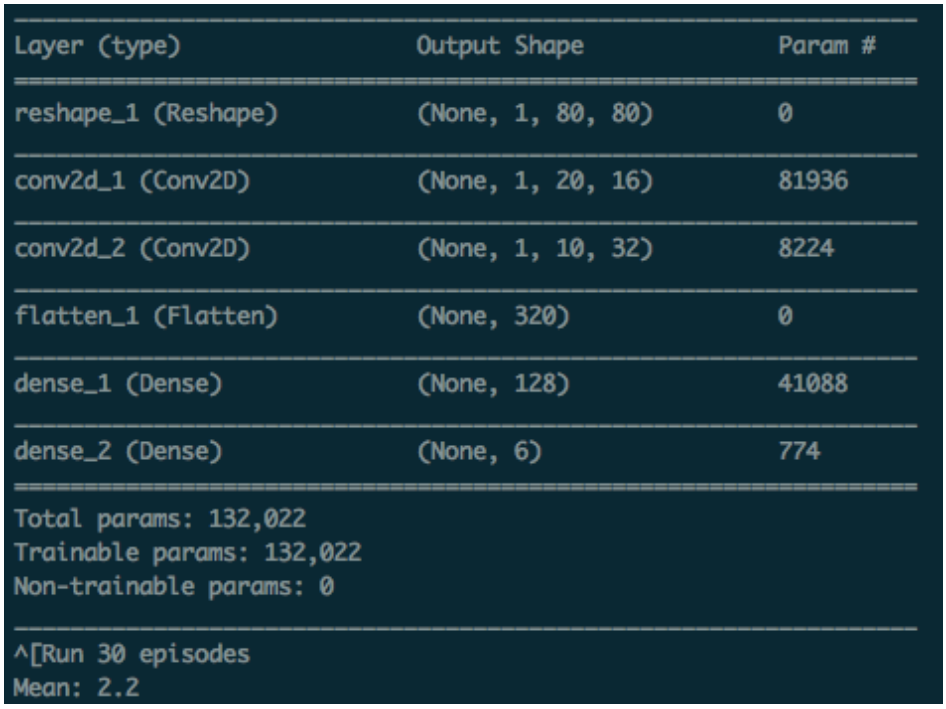
Fig. 1. Me when training deep reinforcement learning.

Model Description

Policy Gradient

這次作業我嘗試了兩種 NN model，一種是一層 Conv2D 加上三層 Dense，詳細如 Fig. 2。這個模型在 10000 episodes 之後仍然沒過 baseline。另一種是後來助教提示的兩層 Conv2D 加上一層 Dense 的模型。Conv2D 跟第一層 Dense 都有過 Relu，最後一層 Dense 則是過 softmax 以選擇 action。Optimizer 的部分使用 RMSprop，因為用 Keras 實作模型本體，RMSprop 的 learning rate 選擇參考助教及官方建議值 1e-4，而 Keras 的 decay 公式為

```
lr = self.lr * (1. / (1. + self.decay * self.iterations))
```



Layer (type)	Output Shape	Param #
reshape_1 (Reshape)	(None, 1, 80, 80)	0
conv2d_1 (Conv2D)	(None, 1, 20, 16)	81936
conv2d_2 (Conv2D)	(None, 1, 10, 32)	8224
flatten_1 (Flatten)	(None, 320)	0
dense_1 (Dense)	(None, 128)	41088
dense_2 (Dense)	(None, 6)	774

Total params: 132,022
Trainable params: 132,022
Non-trainable params: 0

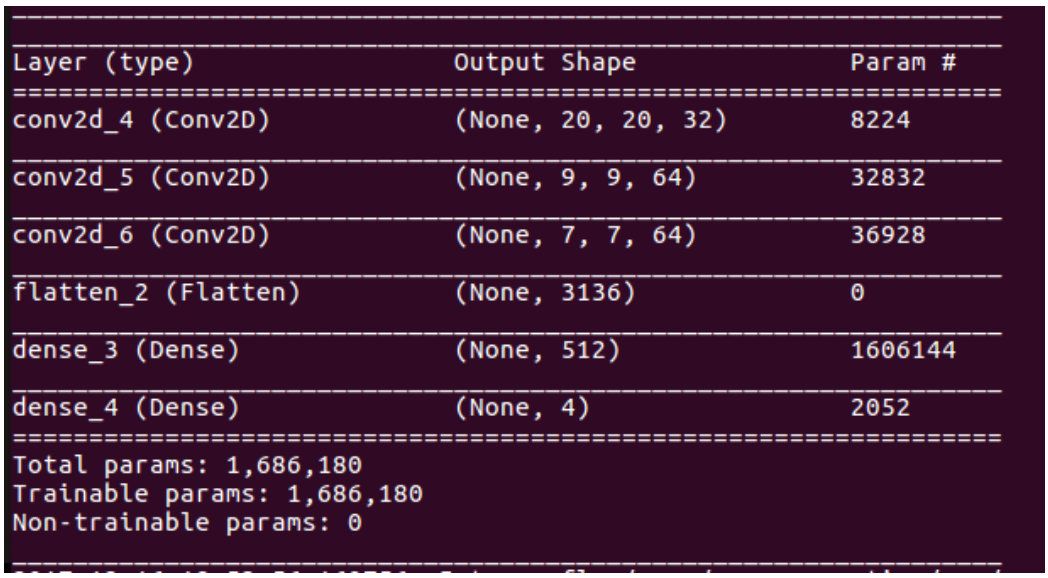
^[Run 30 episodes
Mean: 2.2

Fig. 2. Model summary of policy gradient.

所以設定成 0.99 是錯的，除了 learning rate 其他保持不動得到的收斂效果最好。另外計算 gradients 的部分，discount rewards 的 gamma 值設定為 0.99。

DQN

DQN 因為一開始很難 train 起來，後來是參考助教提供的模型架構，也就是三層 Conv2D 加上一層 Flatten 和兩層 Dense，如 Fig. 3。



Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 20, 20, 32)	8224
conv2d_5 (Conv2D)	(None, 9, 9, 64)	32832
conv2d_6 (Conv2D)	(None, 7, 7, 64)	36928
flatten_2 (Flatten)	(None, 3136)	0
dense_3 (Dense)	(None, 512)	1606144
dense_4 (Dense)	(None, 4)	2052

Total params: 1,686,180
Trainable params: 1,686,180
Non-trainable params: 0

Fig. 3 Model summary of DQN.

前三層 Conv2D 和第一層 Dense 都使用 Relu 當激化函數，exploration stetps 設為 1000000，每 10000 步會更新一次 model weights。大約在 500000 個 global steps 之後若平均 Q 值有上升代表模型有在學到東西。

Experiments

Policy Gradient

這次的訓練過程中發現前處理對模型的收斂有很大影響，一開始使用 Andrej Karpathy 的文章 [1] 裡面使用的前處理方法，直接取 RGB 圖片的 x 維，因為 Karpathy 的教學使用 stochastic gradient descent 當作 model，他將 states 也就是每個畫格壓成 (6400,) 的一維陣列來做 policy backward。我依照這個流程訓練了 10000 發現平均分數雖然看起來有提升，但收斂的效果很不明顯，即到了後期一個 episode 可能得到 15 分，也可能下一場就變 -15 分，後來改用助教 rgb2gray 的前處理方法在壓成 1-d 和 2-d 兩種 states 做訓練，發現使用 rgb2gray 的收斂速度明顯變快，大約 800 個 episodes 時 reward 就會有顯著上升，但跟 2-d array 相比還是很容易有一次輸到 -15 分以下導致 running_mean（計算方式為 0.99 前面的 reward_sum + 0.01 的 current_reward）無法穩定上升。在 Pong 的遊戲上，訓練經驗上 5000 episodes 後會稍微減緩分數增加，但不會再繼續出現 -21、-20 這樣的分數。表 1 為比較 1-d array 和 2-d array 在 5000 episodes 時的 running_mean。Fig. 4 和 Fig. 5 則為比較兩者的 learning curves，可以明顯看到到了 8000 episode 之後 2-d 的模型開始穩定收斂，並且開始穩定快速增加 reward，而 1-d 在 10000 episode 之後仍然一直在震盪。對於這樣的現象，蠻直覺的解釋是因為 2D Convolution 輸入圖片資訊壓成一維並不會有比較好的效果。

DQN

Fig. 6. 為 DQN 的 learning curve。

	running_mean on 5000 episodes
1-d array states	-13.0
2-d array states	-11.0

Table 1. 比較 1-d array states 和 2-d array states 在 5000 episodes 之後的 running_mean.

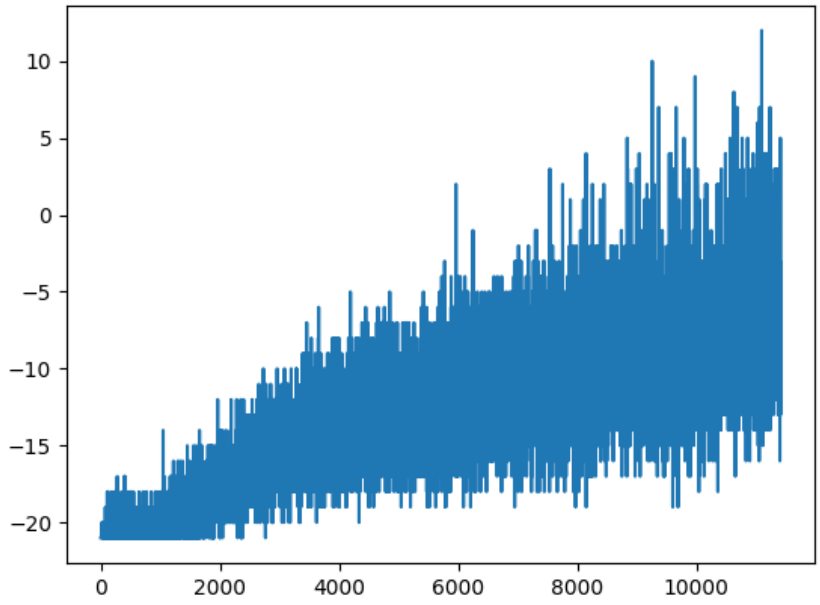


Fig. 4. Learning curve of 1-d array states.

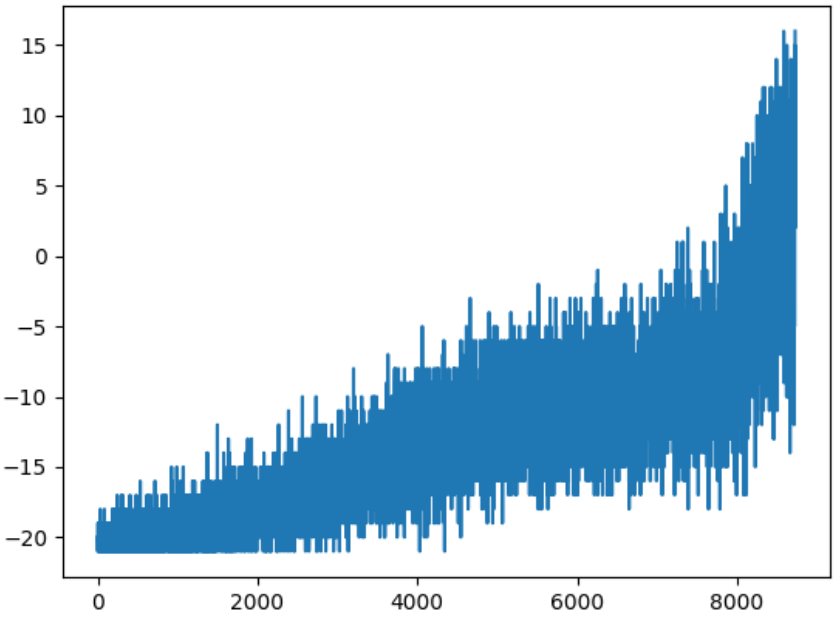


Fig. 5. Learning curve of 2-d array states.

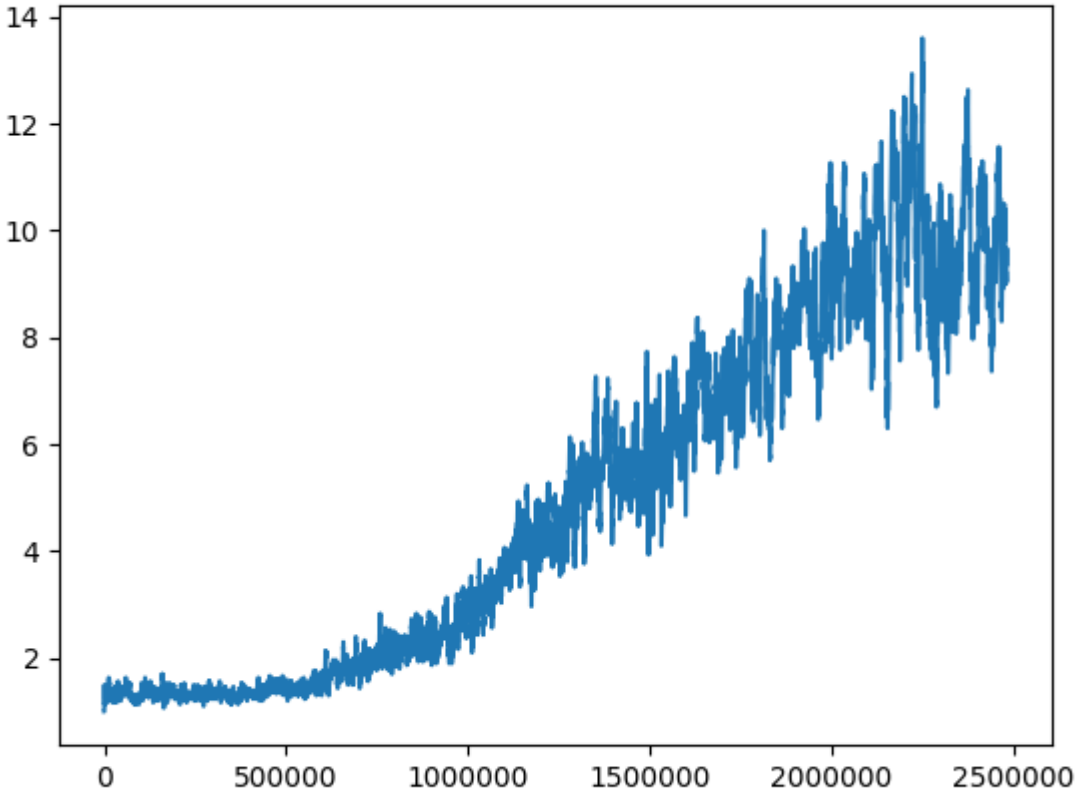


Fig. 6. Learning curve of DQN.

DQN hyperparamter 的部分，由於訓練都需花費相當多的時間，想選擇透過改變 exploration rate 來加速訓練並觀察其訓練曲線的變化。設定中，Epsilon 起始值為 1、最終值為 0.1，代表一開始盡量隨機選擇動作來探索更多狀態，訓練到後來則透過模型預測來選擇動作。實驗中發現，若 epsilon 遞減過快，則過多狀態從來沒被訓練到，訓練雖可以較早看出進步，然分數會很快在低分處飽和、無法再得到更好的結果。使 epsilon 緩慢遞減，可以探索足夠狀態後，再專注於模型的預測，雖大量增加訓練時間，卻較能保證最後得到夠好的結果。

Reference

[1] Deep Reinforcement Learning: Pong from Pixels: http://karpathy.github.io/2016/05/31/rl/