4-1

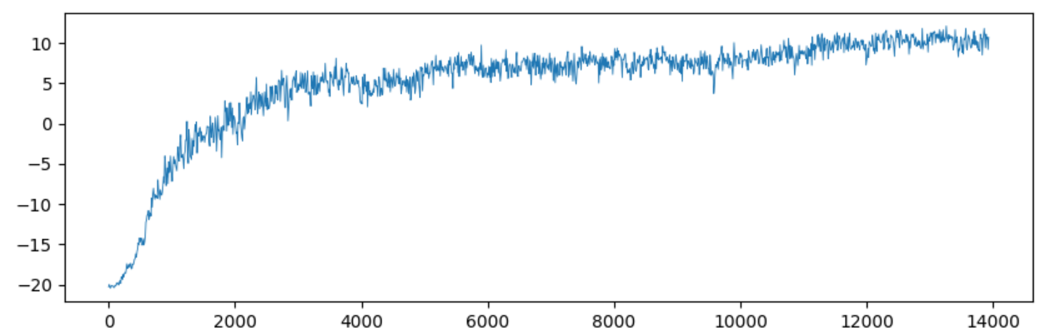
Policy Gradient model:

```
Policy(
  (l1): Linear(in_features=6400, out_features=256, bias=True)
  (l2): Linear(in_features=256, out_features=2, bias=True)
)
```

Learning Curve:

X-axis: number of time steps

Y-axis: average reward in last 30 episodes



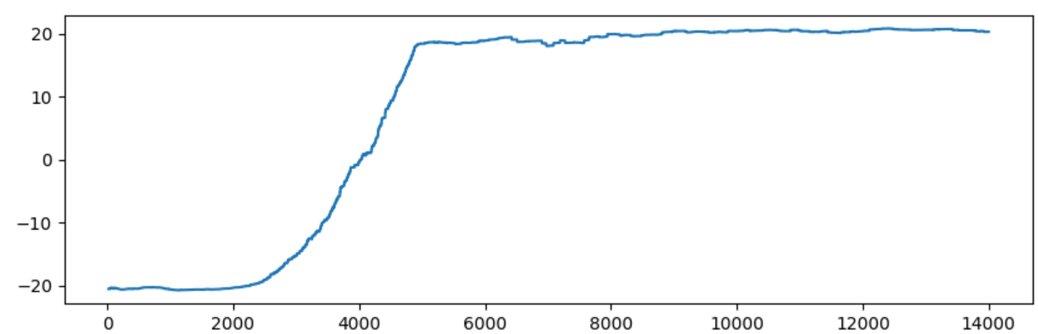
Implement 1 improvement method on page 8

Describe your tips for improvement

Learning curve

Compare to the vanilla policy gradient

Proximal Policy Optimization: 進步非常明顯，收斂的分數較原本的 PG 好很多



4-2

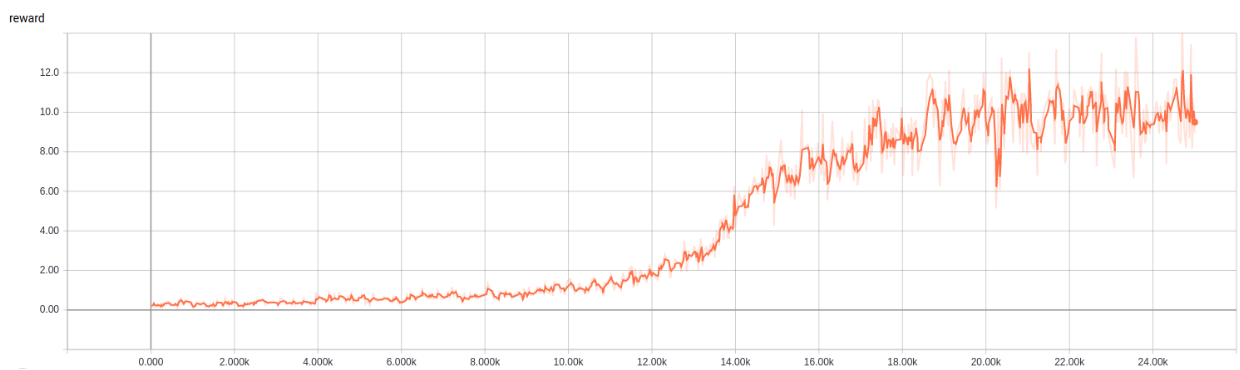
DQN model:

```
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))
  (flatten): Flatten(
  )
  (dense): Linear(in_features=3136, out_features=512, bias=True)
  (output): Linear(in_features=512, out_features=4, bias=True)
)
```

Hyperparameters:

- Gamma: 0.99
- Batch size: 32
- Replay buffer size: 10000
- Epsilon start: 0.1
- Epsilon end: 1.0
- Epsilon decay: 1e-6
- Online net update frequency: 4 (frames)
- Target net update frequency: 1000 (frames)
- Optimizer: Adam, lr=1e-4

Learning Curve:

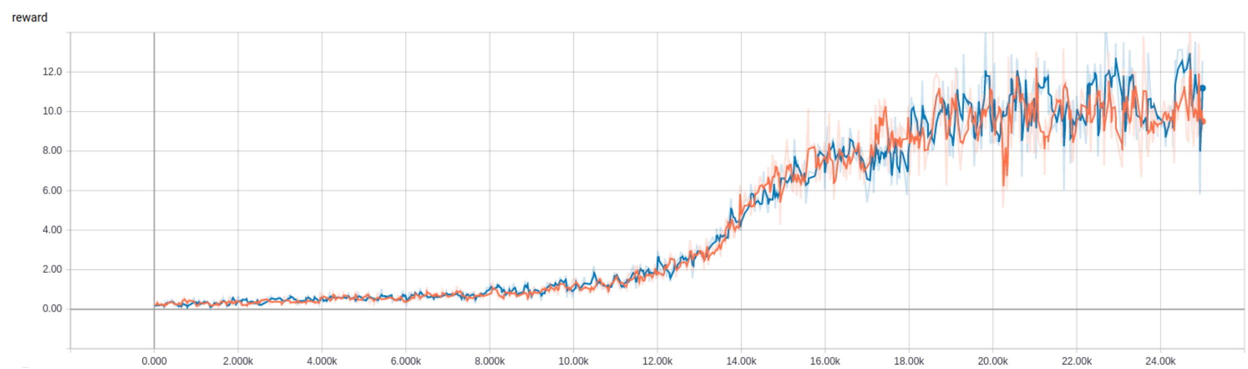


X-axis: number of frame
Y-axis: reward

Improvement tips: **Duel DQN**

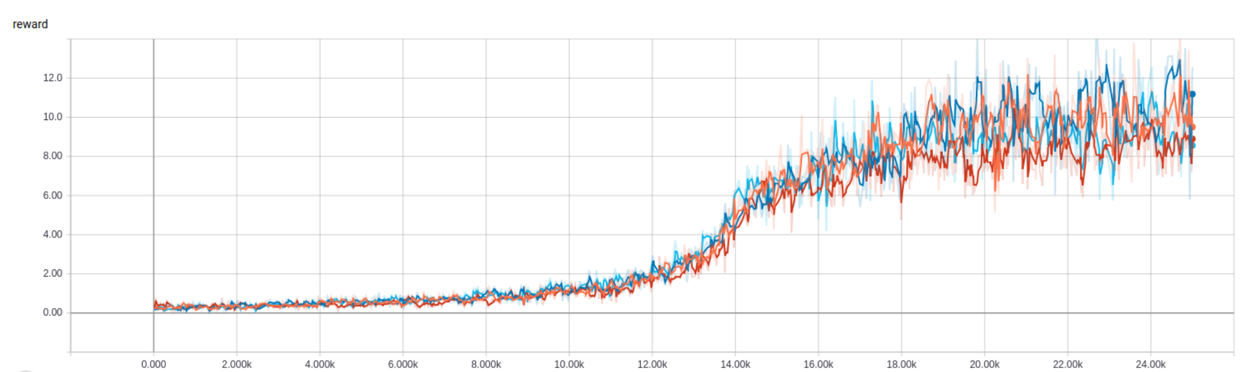
我用的方法是 Duel DQN，Duel DQN 將 Q value 分成只和 state 有關的 V(s) 和 advantage A(s, a)，可以表示為 Q(s, a) = V(s) + A(s, a)。下圖是 learning curve:

下圖是和原本的 DQN 作比較：



橘線是原本的 dqn，藍線是 duel dqn
我實做出來的 duel dqn 和原本的 dqn 的結果沒有太大的差別，learning curve 也很接近。

我又多 train 了幾次 duel dqn，換不同的參數下去試，結果如下圖



橘線是 original dqn，藍、淺藍、紅是三種不同參數的 duel dqn，結果還是都差不多。

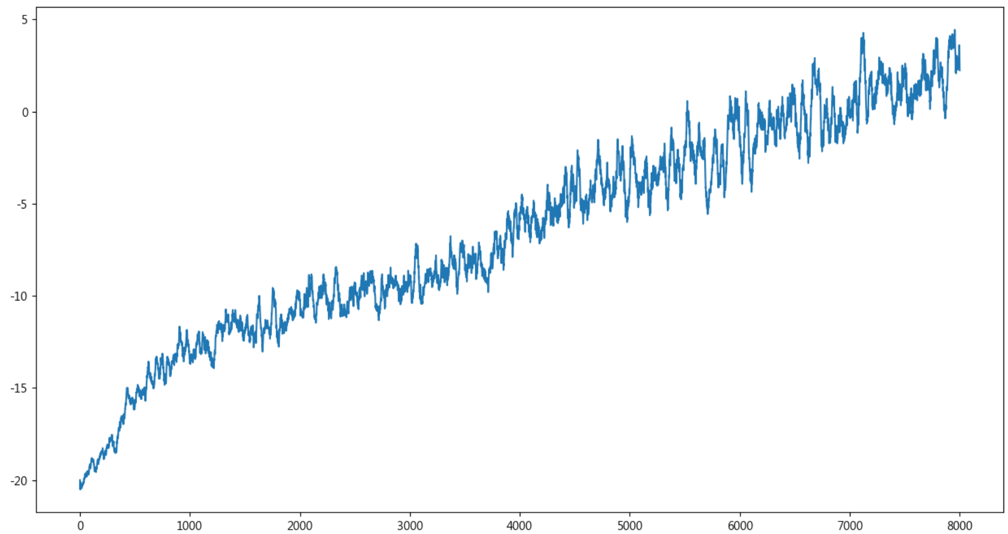
4-3

Describe your actor-critic model on Pong and Breakout
Plot the learning curve and compare with 4-1 and 4-2 to show the performance of your actor-critic model on Pong & Breakout

Actor-Critic model (Pong):

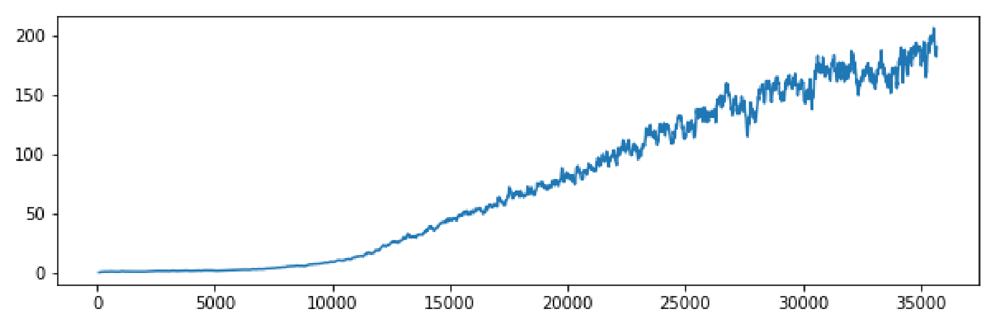
```
Policy(
  (affine1): Linear(in_features=6400, out_features=128, bias=True)
  (action_head): Linear(in_features=128, out_features=6, bias=True)
  (value_head): Linear(in_features=128, out_features=1, bias=True)
)
```

Actor 以及 Critic 共用前面的 Linear layer，後面分別 output action 以及 state score。
Gamma = 0.99, Optimizer = Adam(lr = 1e-4)



X-axis: number of episode
Y-axis: average reward in last 30 episodes
發現 reward 提升的比原本的 PG 更為穩定，提升幅度比較低的原因我猜想是因為我為了降低訓練時間而把中間 hidden layer 的 unit 從 256 降成 128 導致。

<u>Actor-Critic (Breakout) :</u>



X-axis: number of episode
Y-axis: average reward in last 100 episodes

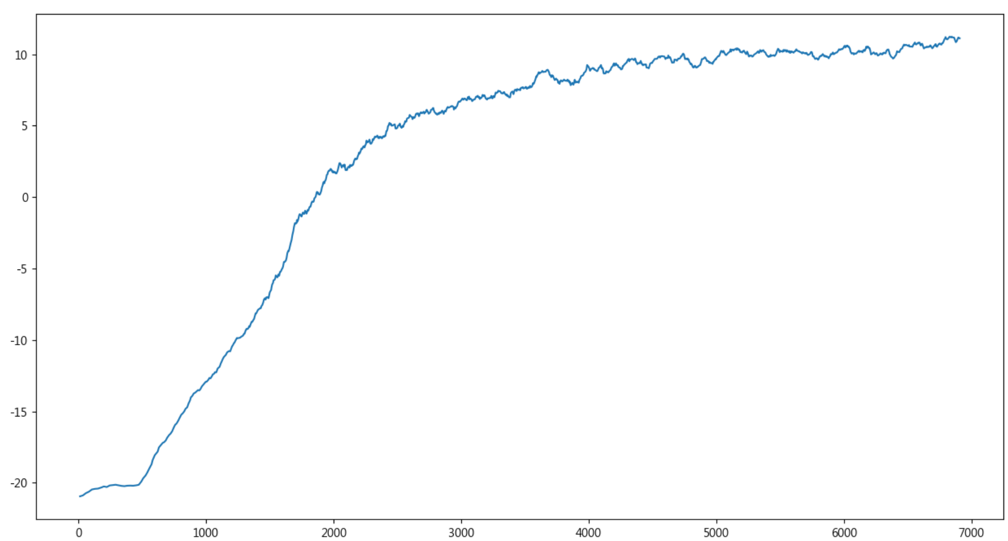
Improvement tips: A3C

A3C 是根據 Actor-Critic 所提出的一種算法，可以解決 Actor-Critic 不收斂的問題。主要是透過平行的訓練，使各個agent share同一個model結構，藉此讓相鄰兩次的更新比較沒有關聯性，進而提高其收斂性。

Pong:

```
NNPolicy(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv4): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (gru): GRUCell(800, 256)
  (critic_linear): Linear(in_features=256, out_features=1, bias=True)
  (actor_linear): Linear(in_features=256, out_features=4, bias=True)
)
```

Learning Curve:



整條曲線有smooth過，採取train時間一分鐘採取一個點。

X-axis: number of episode

Y-axis: average reward in last 100 episodes

A3C 前面reward提升的速率比較快，大約在2500 episodes的時候可以達到baseline