1. Generative adversarial network (GAN)

A. Data augmentation can be used to enhance GAN training. Describe how you preprocess the dataset (such as resize, crop, rotate and flip) and explain why.

batch_size = 128

image_size = 64

num_epochs = 7

lr = 0.0002

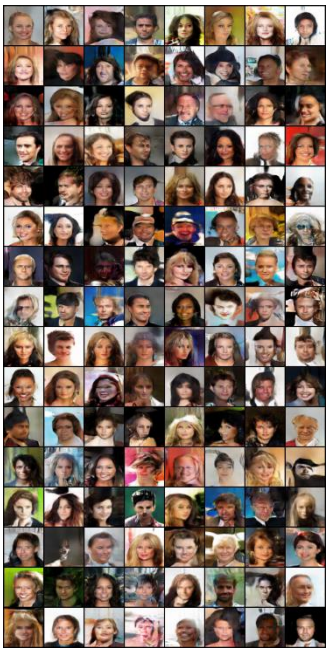
image => Resize(64,64)/ Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))

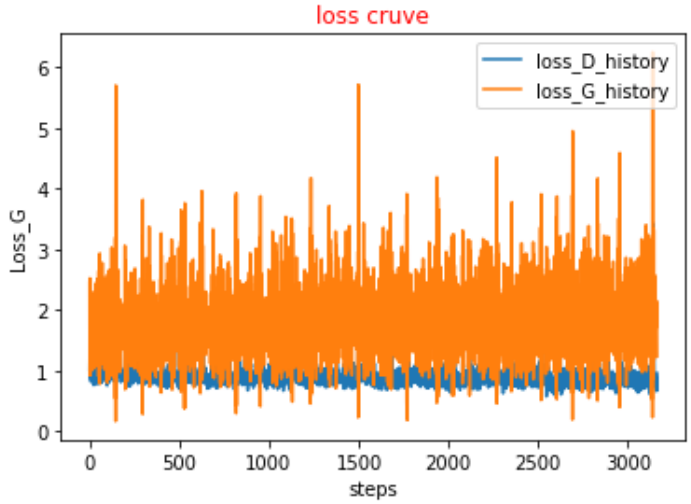
把照片都處理成一致的大小和數值都介於 0~1 之間有助於 discriminator 和 generator 的處理

B. plot the learning curves for both generator and discriminator, and draw some samples generated from your model.

epoch:0/step: 0 epoch:0/step:100 epoch:0/step:200

epoch:5/step:1500





C. Implementation details are addressed as follows

i. Models has already been designed in Model.py. Feel free to modify the generator and discriminator, and you can write down how you design your model and why. (bonus 5 points)

Yes.我有改一些 discriminator model 的內容，我在每層 layer 中都加入 Dropout(0.15)，我從 Dropout(0.5)一路試到 0.15

會加入 Dropout 的原因為我認為這個 discriminator model 太強了一點，會讓 generator 的成長幅度降低，找不到平衡，而會一路下降

dropout 是因為 dropout 太大會讓 discriminator 太弱，辨識效率差

ii. In data preprocessing, the ImageFolder and Dataloader provided by pytorch are recommended. The customized dataset (without using ImageFolder) can be implemented for extra points. (bonus 5 points)

Yes.我有自己寫一個 AlignData，而不是直接使用 ImageFolder，dataset = AlignData(path)。AlignData 裡面的功能有__init__, __len__, _get_item__，傳入 folder 的位置，ouput 即可得 dataset

iii. In main.py, you have to complete three functions. main(), train(), and visualizaion.

Yes.

iv. Visualization.

Yes.

D. Please do some discussion about your implementation. You can write down the difficulties you face in this homework, e.g. hyperparameter settings, analysis of the generated images, or anything you want to address.

我在實作的過程中有遇到 3 個比較大的問題：

1. 我的 dataset 忘記做 normalization，這讓我的 generator 產生很大的問題，因為 generator 的 model 最後是 tanh()是介於 0~1 之間，所以一開始的圖片都很差，改過後才有大幅進步。
2. 我忘記把我的 b_y(batch label) unsqueeze，這也和上面產生的影響類似，改過後就進步很多。
3. 在來這個就是 discriminator 太強的問題，如同我在(i)的敘述，我在 model 中加入 Dropout 去降低 discriminator 的強度，長是很多次才找到 0.15，也覺得結果不錯。

2. Deep Q Network (DQN)

A. Explain the purpose of the following hyperparameters: updating step α, discount factor γ, target network update period τ, and ε for ε -greedy policy.

α：learning rate.

γ：代表越是未來所給的 reward 影響是越小的，當下的 reward 是最大的。

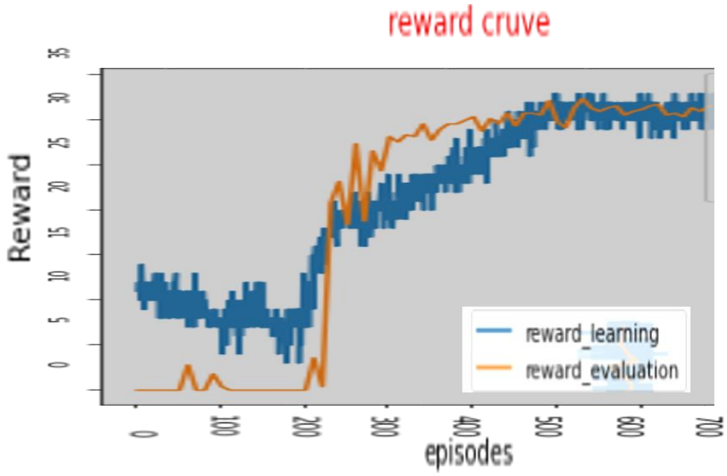
τ：source code 裡面是 10000，代表 10000 次後才會去更新 target Q，這個限制是為了讓 policy Q 可以有時間去縮短和 target Q 之間的距離，τ 太大的話會讓 policy Q 停滯太久，而 τ 太小的話則會讓 policy Q 不夠去訓練到 target Q 就更新下一個。

ε：source code 中會從 0.99 一路下降，目的是為了要增加一開始訓練時會亂數選擇要使用哪個 action，而不會一開始就直接從上一次經驗中做選擇。

B. To speed up the training process, you can simply change the probability of random agent: [NOOP (0.3), UP (0.6), DOWN (0.1)]. Please show the total reward of sample episodes for this configuration.

```
[Info] make directory './model'
Episode:      0, interaction_steps:   2048, reward: 12, epsilon: 0.998157
[Info] Save model at './model' !
Evaluation: True, Episode:      0, Interaction_steps:   2048, evaluate rewar
Episode:      1, interaction_steps:   4096, reward: 11, epsilon: 0.996314
Episode:      2, interaction_steps:   6144, reward: 10, epsilon: 0.994470
Episode:      3, interaction_steps:   8192, reward: 12, epsilon: 0.992627
Episode:      4, interaction_steps:  10240, reward:  8, epsilon: 0.990784
Episode:      5, interaction_steps:  12288, reward: 11, epsilon: 0.988941
Episode:      6, interaction_steps:  14336, reward: 11, epsilon: 0.987098
Episode:      7, interaction_steps:  16384, reward: 12, epsilon: 0.985254
Episode:      8, interaction_steps:  18432, reward: 10, epsilon: 0.983411
Episode:      9, interaction_steps:  20480, reward: 10, epsilon: 0.981568
Episode:     10, interaction_steps:  22528, reward:  9, epsilon: 0.979725
```

C. Use the modified random agent in the -greedy and keep training. Show your configuration and discuss what you find in training phase.



我發現兩條 curve 都有不穩定的小幅度浮動，雖然整體是持續上升，但細部觀察是有不穩定性的，然後兩條 curve 最終都收斂在 30 左右。

D. After training, you will obtain the model parameters for the agent. Show total reward in some episodes for deep Q-network agent.

```
[Info] Restore model from './model/q_target_checkpoint_1538048.pth' !
Episode:      0, interaction_steps:      0, reward: 32, epsilon: 0.100000
Episode:      1, interaction_steps:      0, reward: 31, epsilon: 0.100000
Episode:      2, interaction_steps:      0, reward: 33, epsilon: 0.100000
Episode:      3, interaction_steps:      0, reward: 31, epsilon: 0.100000
Episode:      4, interaction_steps:      0, reward: 32, epsilon: 0.100000
Episode:      5, interaction_steps:      0, reward: 32, epsilon: 0.100000
Episode:      6, interaction_steps:      0, reward: 32, epsilon: 0.100000
Episode:      7, interaction_steps:      0, reward: 31, epsilon: 0.100000
Episode:      8, interaction_steps:      0, reward: 32, epsilon: 0.100000
Episode:      9, interaction_steps:      0, reward: 33, epsilon: 0.100000
```

E. Sample some states, show the Q values for each action, analyze the results, and answer

i. Is DQN decision in the game the same as yours? Any good or bad

move?

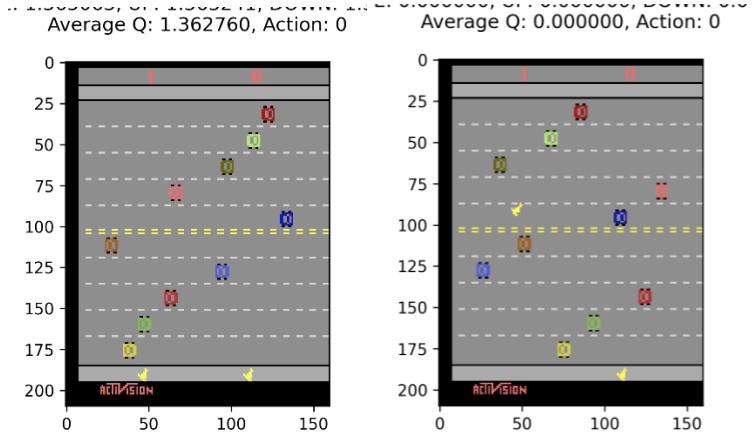
大部分一樣，但有兩點比較不一樣：

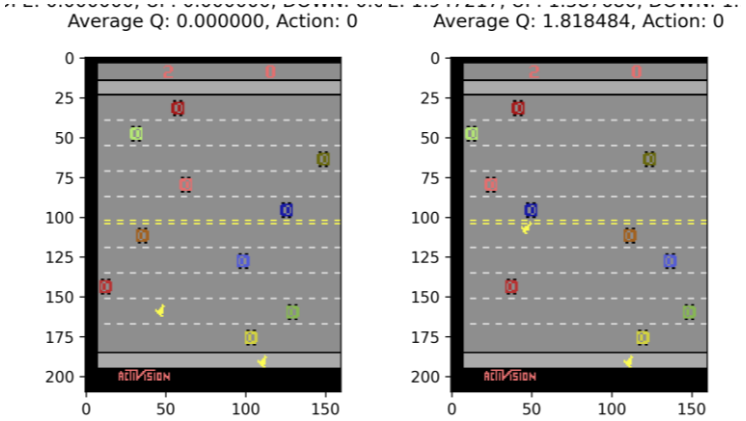
1. DOWN 的決定比較常不一樣，DQN 在遊戲一開始會一直選擇 DOWN，之後才變 NOPE，我不會做這個決定
2. DQN 選擇 NOPE 的次數太多，我不會 NOPE 那麼多次

ii. Why the averaged Q-value of three actions in some state is larger or less than those of the other states?

Q value 可以視為做某個動作的 reward，因此 average Q-value 就是這次 state 中平均可得的 reward，因此他會有時候大有時候小。

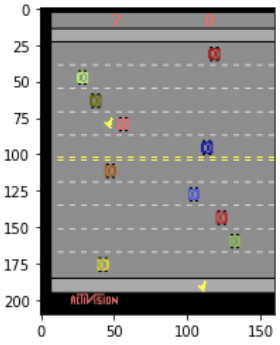
NOPE:



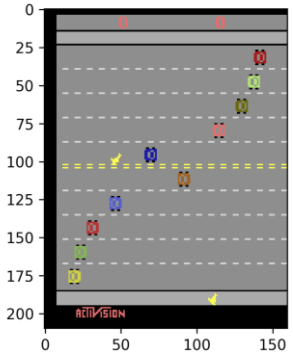


UP:

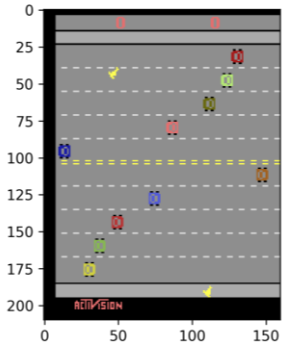
Step: 0453
NOPE: 2.105845, UP: 2.134763, DOWN: 2.079800
Average Q: 2.106803, Action: 1



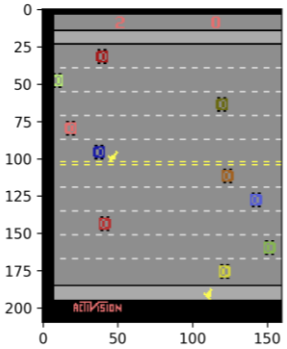
Average Q: 1.871132, Action: 1



Average Q: 2.157207, Action: 1

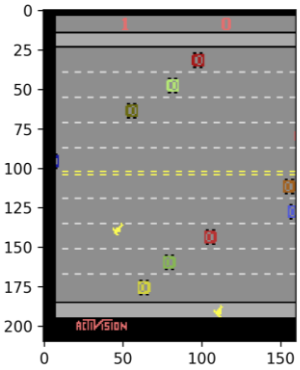


Average Q: 1.987141, Action: 1

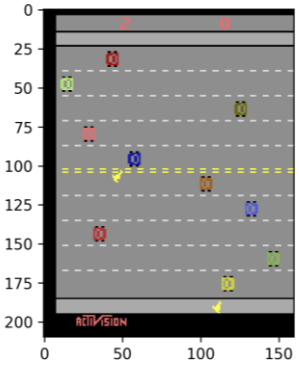


DOWN:

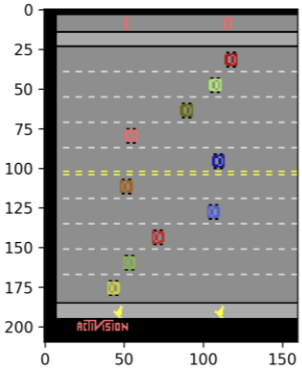
Average Q: 0.000000, Action: 2



Average Q: 1.892956, Action: 2



Average Q: 0.000000, Action: 2



Average Q: 1.451330, Action: 2