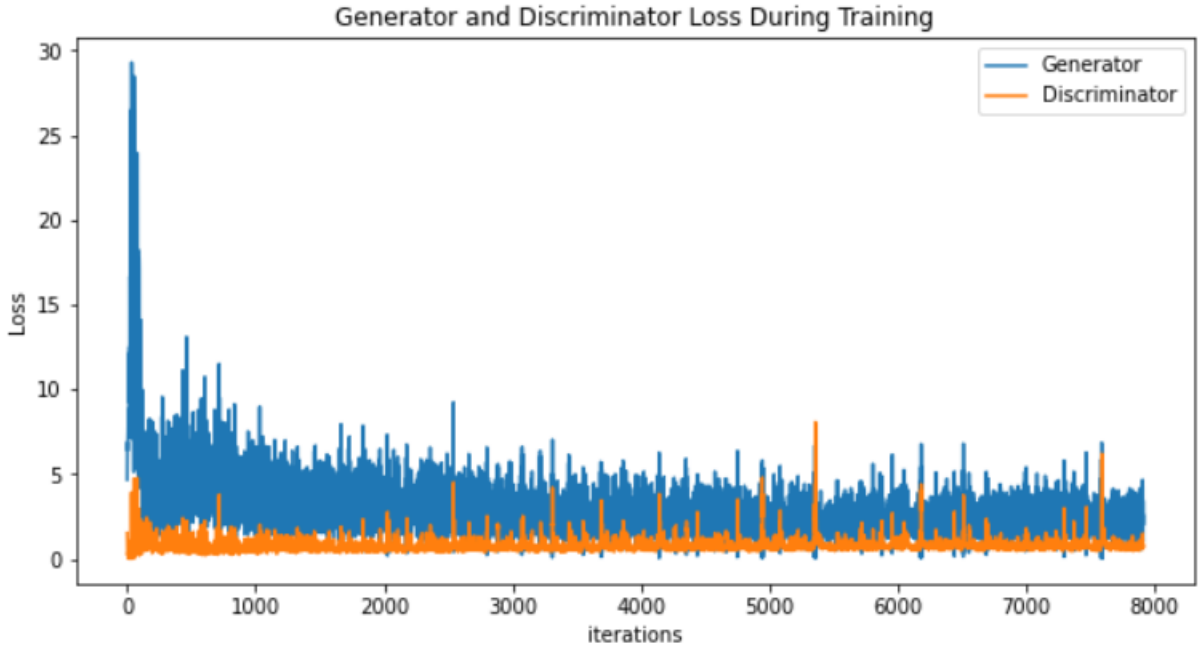
一、 Generative adversarial network (GAN)

1. 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.
   i. 使用 pytorch 中的 dset.ImageFolder 做圖像的前處理
   ii. 首先將圖片都壓成同樣的像素量(resize)
   iii. 再來我們將圖片以中心點出發以同樣長寬裁切(CenterCrop)
   iv. 最後再做 Normalize，將 RGB 每個數值的量從 0~255 壓到平均值 0.5、標準差 0.5
2. Construct a DCGAN with vanilla GAN objective, plot the learning curves for both generator and discriminator, and draw some samples generated from your model.



3. Implementation details are addressed as follows
   (a) skip
   (b) skip
   (c) In main.py, you have to complete three functions. main(), train(), and visualizaion.
   • main() : you have to set up dataset (dataloader), models, optimizers, and the criterion. After preparation, the train function is called to start the training procedure.
   • train() : In every iteration, you have to perform the following jobs

– send true data into the discriminator, and update the discriminator.

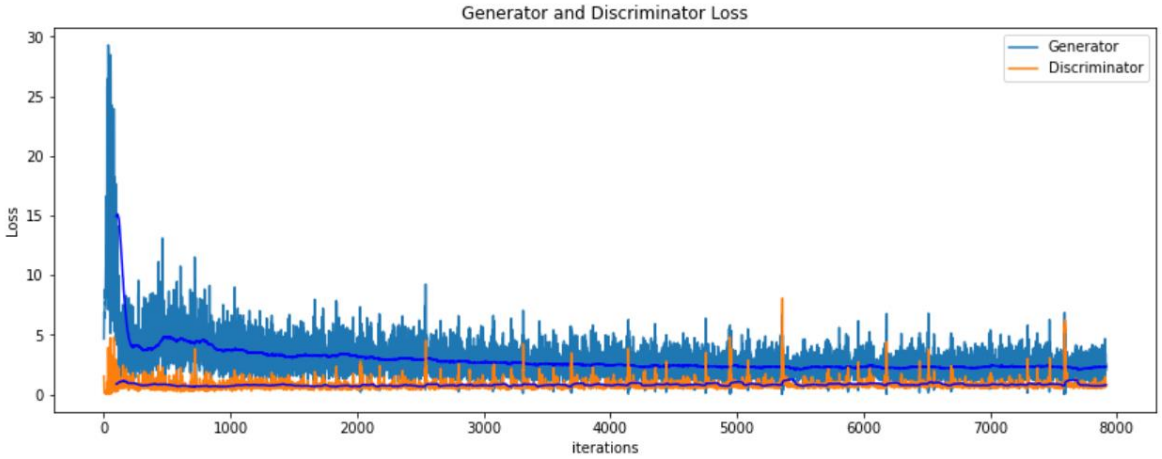
– use generator to create fake data, and send them to the discriminator. Calculate loss for both models and update them.

– record the loss in every iteration and draw some samples by the current generator after the fixed number of iterations.

– after finishing all epochs, you have to save the files of your models, losses, and samples. (sampling should not be more than 20 times)

詳見我的.py 檔案，在此不再貼上程式碼。

(d) 其實我不太懂這題要做甚麼…所以我就把本來畫出來的 loss 圖加上了一個 100moving average 線。



(e) Generate 出來的圖會偏白，生成出來的人臉大致上沒甚麼大問題。只是觀察(d)小題我畫出來的 loss 圖會發現：generator 的 loss 還是有下降趨勢，只是很不明顯，所以如果能夠把 learning rate 調大一點，或許能再把 loss 往下調。另外關於偏白這件事情有兩個方向可以解決：(1)治本方法：或許我們可以把 normalize 那邊的平均值往下調一點，讓整體照片變暗。(2)治標方法：或許我們在圖片生成後，使用影像處理方法把 RGB 平移到比較暗的地方以解決這個問題。

二、Deep Q Network (DQN)

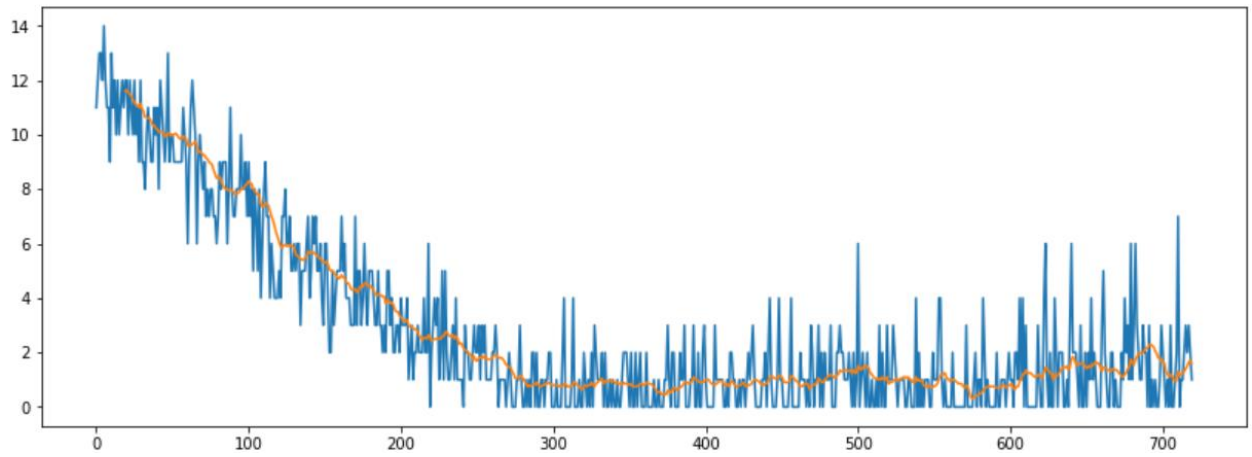
DQN 這題 debug 很久，但還是失敗了。感覺一直沒有把 model 丟進去 env 裡面，但是檢查很多遍都沒問題。

1.

2.

```
Episode:      1, interaction_steps:   2048, reward: 11, epsilon: 0.997568
[Info] Save model at '/home/jeff/mount/TimHsu_HW3/DQN/model' !
Evaluation: True, episode:      1, interaction_steps:   2048, evaluate reward:  0
Episode:      2, interaction_steps:   4096, reward: 12, epsilon: 0.995136
Episode:      3, interaction_steps:   6144, reward: 12, epsilon: 0.992704
Episode:      4, interaction_steps:   8192, reward: 13, epsilon: 0.990272
Episode:      5, interaction_steps:  10240, reward: 13, epsilon: 0.987840
Episode:      6, interaction_steps:  12288, reward: 12, epsilon: 0.985408
Episode:      7, interaction_steps:  14336, reward: 14, epsilon: 0.982976
Episode:      8, interaction_steps:  16384, reward: 12, epsilon: 0.980544
Episode:      9, interaction_steps:  18432, reward: 11, epsilon: 0.978112
Episode:     10, interaction_steps:  20480, reward: 11, epsilon: 0.975680
Episode:     11, interaction_steps:  22528, reward:  9, epsilon: 0.973248
Evaluation: True, episode:     11, interaction_steps:  22528, evaluate reward:  4
```

3. 照理來說應該要在約 400 的時候 reward 上來的，可惜事與願違，猜測是沒有把 model 丟進去 env 裡面，但是檢查很多遍都沒問題。



4. 我怎麼覺得這個是照一開始給的 UP Noop Down 機率亂給，然後訓練出來的分數。

```
test()
```

```
Episode:      0, interaction_steps:      0, reward: 10, epsilon: 1.000000
Episode:      1, interaction_steps:      0, reward: 10, epsilon: 1.000000
Episode:      2, interaction_steps:      0, reward: 15, epsilon: 1.000000
Episode:      3, interaction_steps:      0, reward:  8, epsilon: 1.000000
Episode:      4, interaction_steps:      0, reward: 12, epsilon: 1.000000
Episode:      5, interaction_steps:      0, reward: 12, epsilon: 1.000000
Episode:      6, interaction_steps:      0, reward: 13, epsilon: 1.000000
Episode:      7, interaction_steps:      0, reward: 10, epsilon: 1.000000
Episode:      8, interaction_steps:      0, reward: 12, epsilon: 1.000000
Episode:      9, interaction_steps:      0, reward:  9, epsilon: 1.000000
```