Machine Learning HW11 Report

學號：B06901045 系級：電機三 姓名：曹林熹

1. (2.5%) 訓練一個 model。

a. (1%) 請描述你使用的 model（可以是 baseline model）。包含 generator 和 discriminator 的 model architecture、loss function、使用的 dataset、optimizer 參數、以及訓練 step 數（或是 epoch 數）。

ANS：

- model architecture

  在這邊我使用的是助教的 sample code DCGAN，下面的圖片給出了我們的 Generator 與 Discriminator 架構。

```
Generator(
  (l1): Sequential(
    (0): Linear(in_features=100, out_features=8192, bias=False)
    (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (l2_5): Sequential(
    (0): Sequential(
      (0): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (1): Sequential(
      (0): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (2): Sequential(
      (0): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (3): ConvTranspose2d(64, 3, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (4): Tanh()
  )
)
```

```
Discriminator(
  (ls): Sequential(
    (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Sequential(
      (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (3): Sequential(
      (0): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (4): Sequential(
      (0): Conv2d(256, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
    (6): Sigmoid()
  )
)
```

- loss function
  這裡我使用的是 BCELoss()，分別用來計算 Generator 與 Discriminator 的 loss 並且把他們加起來得到最終的 loss function。

- dataset
  這裡我使用的是 Crypko Beta 助教給的動漫 dataset，沒有額外使用 data。

- optimizer 參數
  這裡我使用的是 Adam。

- 訓練 step 數（或是 epoch 數）
  我的 epoch 數量為 10 次。

b. (1.5%) 請畫出至少 16 張 model 生成的圖片。

ANS：

我總共生成了 20 張圖片，為了能夠還原回去，因此我存了 tensor 資料下來。

可以看到我們的 model 產生出的圖片大部份是好的，我認為最好的圖片是由左上數過來第三張圖，眼睛與頭髮位置都很合理；而我認為最糟糕的圖是由左下數過來第三張圖，可以看到整個臉部崎嶇，眼睛甚至還翻轉。

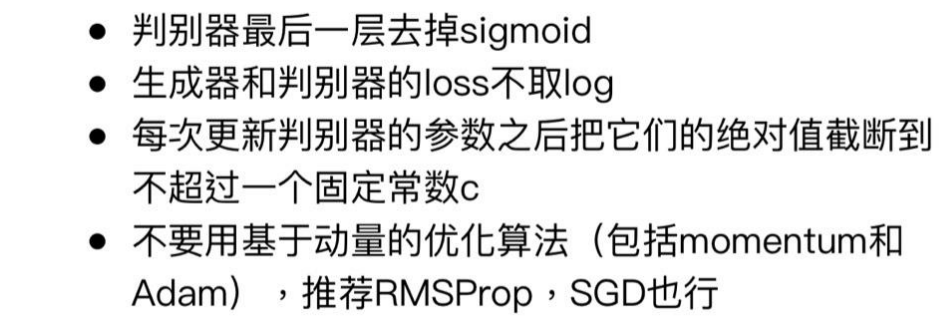
2. (3.5%) 請選擇下列其中一種 model： WGAN, WGAN-GP, LSGAN, SNGAN（不要和使用的 model 一樣，至少 architecture 或是 loss function 要不同）

a. (1%) 同 1.a ，請描述你選擇的 model，包含 generator 和 discriminator 的 model architecture、loss function、使用的 dataset、optimizer 參數、及訓練 step 數（或是 epoch 數）。

ANS：

- model architecture

  在這邊我使用的是 WGAN，實作 WGAN 其實並沒有很困難，可以由　　下面這張圖告訴我們要改的演算法。



下面的圖片給出了我們的 Generator 與 Discriminator 架構。

```
Generator(
  (l1): Sequential(
    (0): Linear(in_features=100, out_features=8192, bias=False)
    (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (l2_5): Sequential(
    (0): Sequential(
      (0): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (1): Sequential(
      (0): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (2): Sequential(
      (0): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1), bias=False)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
    )
    (3): ConvTranspose2d(64, 3, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (4): Tanh()
  )
```

```
Discriminator(
  (ls): Sequential(
    (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Sequential(
      (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (3): Sequential(
      (0): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (4): Sequential(
      (0): Conv2d(256, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (5): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
    (6): LeakyReLU(negative_slope=0.2)
  )
)
```

- loss function

  這裡我使用的不再是原來的 BCELoss()，因為 BCELoss 會有 log 。這邊的 Discriminator 與 Generator loss 改成

  (1) loss_D = -torch.mean(r_logit) + torch.mean(f_logit)

  (2) loss_G = -torch.mean(f_logit)

其中 r_logit 與 f_logit 為 real_imgs 與 fake_imgs 經過 Discriminator 後生成的數值。

- dataset
  這裡我一樣使用的是 Crypko Beta 助教給的動漫 dataset，沒有額外使用 data。

- optimizer 參數
  這裡我使用的是 RMSprop。

- 訓練 step 數（或是 epoch 數）
  我的 epoch 數量為 10 次。

b. (1.5%) 和 1.b 一樣，就你選擇的 model，畫出至少 16 張 model 生成的圖片。

ANS：

我總共也生成了 20 張圖片，一起與 DCGAN 比較，而為了能夠還原回去，因此我存了 tensor 資料下來。我個人認為品質有比 DCGAN 產生的好，可以看出比較沒有畸形的人物臉出現，頂多是眼睛歪歪的一點誤差。我認為最好的圖片是由右下數過來第一張圖，很像庫洛魔法使的女主角；而我認為最糟糕的圖是由右上數過來第二張圖，眼睛卡到頭髮的部份，嘴吧也很奇怪，但是與 DCGAN 最差的比已經有好很多。

c. (1%) 請簡單探討你在 1. 使用的 model 和 2. 使用的 model，他們分別有何性質，描述你觀察到的異同。

ANS：

- DCGAN
  我觀察到的部份就是深度的卷積網路與反卷積做出來的模型，並且最後有一個 Discriminator 做拮抗。

- WGAN
  這個 model 在架構上與 DCGAN 其實蠻相同的，我在前面也有提到他們兩個的差異，如果考慮直接影響到 model 影響的話，主要是在 WGAN 的 Discriminator 最後採用了 LeakyRELU 代替 Sigmoid，目的是為了防止梯度消失，如果不更換這種激活函數的話，導數很容易為 0，造成參數無法更新。另外比較特別的是使用了 Weight Clipping 的方法，給定我們的權重更新一個 threshold ，使得模型參數可以收斂。

3. **(4%) 請訓練一個會導致 mode collapse 的 model。**

a. (1%) 同 1.a ，請描述你選擇的 model，包含 generator 和 discriminator 的 model architecture、loss function、使用的 dataset、optimizer 參數、及訓練 step 數（或是 epoch 數）。

ANS：

mode collapse 就是機器會產生很多看起來很像的 data，在我們的資料集內，就可能會產生許多相同的人臉，好比 generator 在想：反正這個人臉就能騙過 discriminator，那我就都產生此圖片。以下就來實作看看會產生 mode collapse 的 model。

- model architecture –
  在這邊我使用的結構與我上面的 WGAN 相同。

- loss function
  這裡我使用的是一樣是上面的 WGAN loss function。

- dataset
  這裡我一樣使用的是 Crypko Beta 助教給的動漫 dataset，沒有額外使用 data。

- optimizer 參數
  這裡我使用的是 RMSprop。

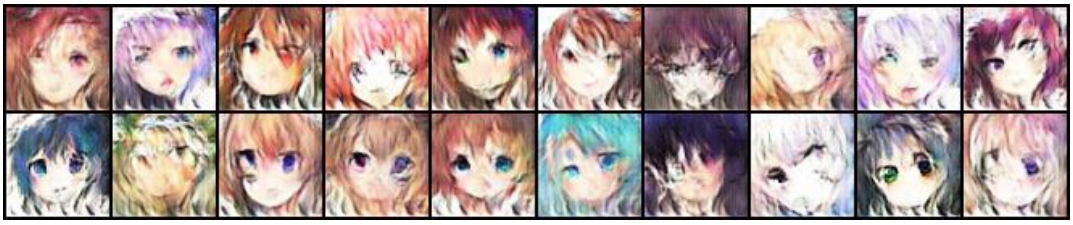
- 訓練 step 數（或是 epoch 數）
  我的 epoch 數量為 10 次。

- Weight Clipping
  這是我主要改的部份，原來的 threshold 為 0.01，我將其改為 0.1 來看看改變結果。

b. (1.5%) 請畫出至少16張 model 生成且具有 mode collapse 現象的圖片。

ANS：

我總共生成了 20 張圖片，與原本的 WGAN 比較，可以清楚看到品質比原 先的 WGAN 產生的差，頭髮的形狀根本長得很像，頂多髮色有時會不太一樣，所以機器主要是學習到了”髮型”來騙過 Discriminator。除此之外，我也發現眼睛的角度以及臉的角度也都非常類似，可以算是 mode collapse 的結果。



c. (1.5%) 在不改變 optimizer 和訓練 step 數的情況下，請嘗試使用一些方法來減緩 mode collapse。說明你嘗試了哪些方法，請至少舉出一種成功改善的方法，若有其它失敗的方法也可以記錄下來。

ANS：

如果是不改變 optimizer 和訓練 step 數來改善 mode collapse 的話，簡單紀錄以下的可行改善方法。

- 更改 Weight Clipping Threshold：
  把 0.1 改成 0.01，讓 threshold 變小。(上圖 mode collapse/ 下圖 改善)