DLCV HW4 Report

電機三 張景程 b04901138

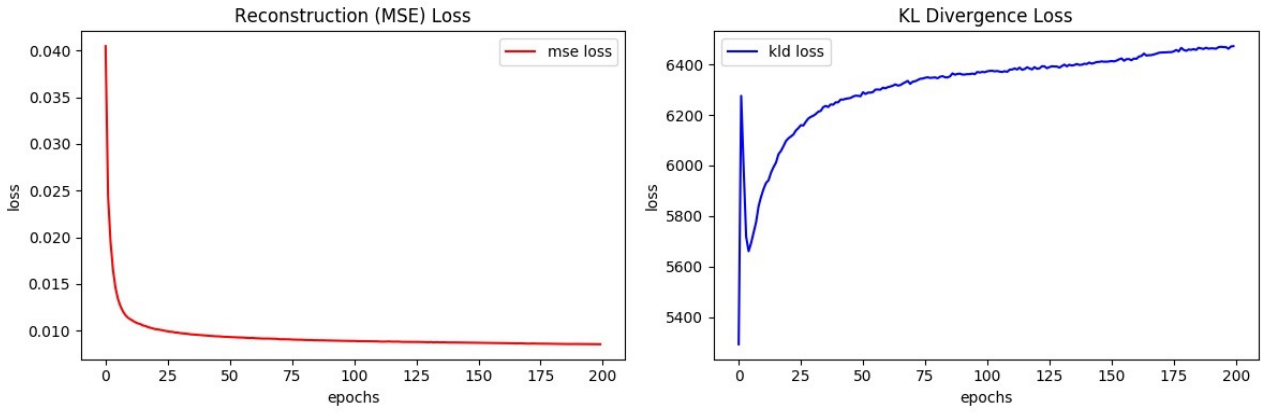
Problem 1. VAE

1. Describe the architecture & implementation details of your model.

   I choose 512 as the latent dimension, and use tanh as the activation function in the last layer. After tanh, the output will be divided by 2 and plus 0.5 so that it'll be normalized to (0,1). Besides, for each pixel of the training images, I divide it by 255 so the input range is also (0,1).

```
VAE(
  (encoder): Sequential(
    (0): Conv2d (3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.005, inplace)
    (3): Conv2d (32, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (5): LeakyReLU(0.005, inplace)
    (6): Conv2d (32, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (8): LeakyReLU(0.005, inplace)
    (9): Conv2d (64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (11): LeakyReLU(0.005, inplace)
  )
  (z_mean): Linear(in_features=2048, out_features=512)
  (z_logvar): Linear(in_features=2048, out_features=512)
  (z_decode): Linear(in_features=512, out_features=2048)
  (decoder): Sequential(
    (0): ConvTranspose2d (128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.01, inplace)
    (3): ConvTranspose2d (64, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (5): LeakyReLU(0.01, inplace)
    (6): ConvTranspose2d (32, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (8): LeakyReLU(0.01, inplace)
    (9): ConvTranspose2d (32, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(3, eps=1e-05, momentum=0.1, affine=True)
  )
  (tanh): Tanh()
)
```

2. Plot the learning curve of the model.

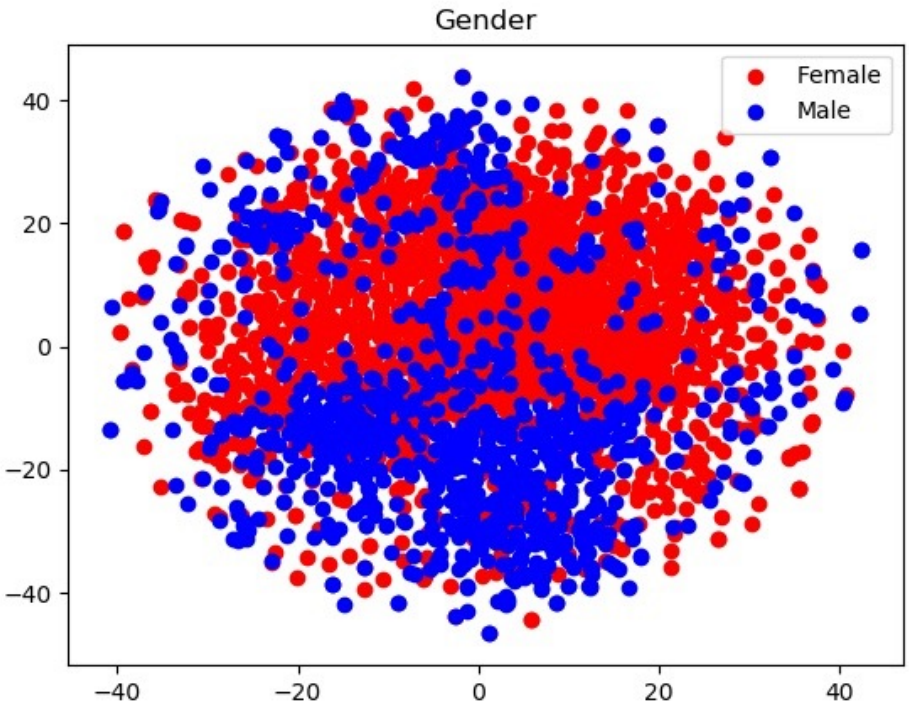


3. Plot 10 testing testing images and their reconstructed results and report your testing MSE of the entire test set.

testing MSE (pixel-wise average) : 0.008632

4. Plot 32 random generated images.

5. Visualize the latent space by mapping test images to 2D space (with tSNE) and color them with respect to an attribute of your choice.
Choose Attribute: Male，我們可以從圖中看出兩種性別大致已分為上下兩群。



6. Discuss what you've observed and learned from implementing VAE.
   (a) 參數lambda對整個實驗結果影響非常深遠，若過小或過大都會使得reconstruction和random generation其中一項的結果壞掉，必須透過條參數的方式取得平衡。
   (b) 由於VAE是先透過一個encoder將input image投影到latent space上，再藉由decoder還原出圖片，可能是因為latent space只能保留局部的重要資訊，導致output的影像都較為模糊，不過人臉的輪廓和五官大致上都還算清晰。

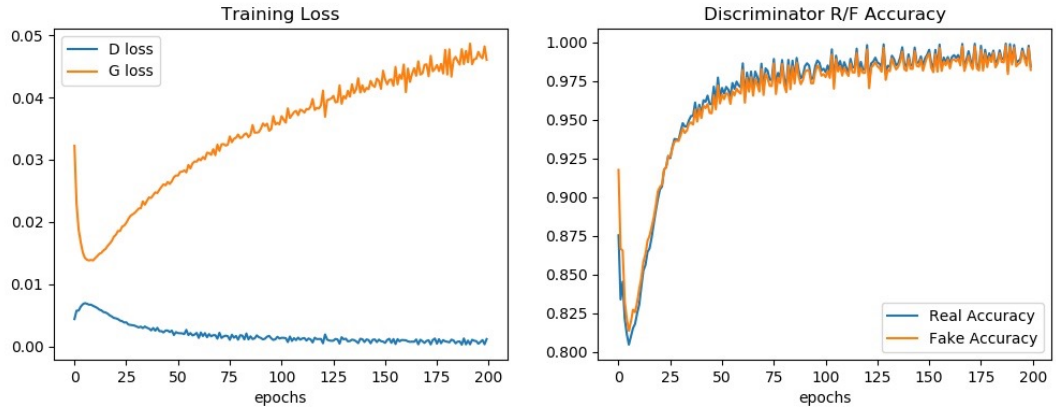
Problem 2. GAN

1. Describe the architecture & implementation details of your model.

   I choose 100 as the latent dimension, and use tanh as the activation function in the last layer. After tanh, the output will be divided by 2 and plus 0.5 so that it'll be normalized to (0,1). Besides, for each pixel of the training images, I divide it by 255 so the input range is also between (0,1).

```
Generator(
  (decoder): Sequential(
    (0): ConvTranspose2d (100, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
    (3): ConvTranspose2d (512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (5): ReLU(inplace)
    (6): ConvTranspose2d (256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (8): ReLU(inplace)
    (9): ConvTranspose2d (128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (11): ReLU(inplace)
  )
  (output): Sequential(
    (0): ConvTranspose2d (64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): Tanh()
  )
)
Discriminator(
  (main): Sequential(
    (0): Conv2d (3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
    (3): Conv2d (64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (5): LeakyReLU(0.2, inplace)
    (6): Conv2d (128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (8): LeakyReLU(0.2, inplace)
    (9): Conv2d (256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (11): LeakyReLU(0.2, inplace)
  )
  (output): Sequential(
    (0): Conv2d (512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
)
```

2. Plot the learning curve of the model and briefly explain what you think it represents.



大概在第75個epoch之後，Discriminator不管是在判斷real的影像還是判斷fake的影像在準確率上都能有不錯的表現，而這兩者仍持續震盪，顯示出Discriminator和Generator持續相互對抗的過程。在loss方面，Discriminator較為穩定，而Generator的loss則震盪的很激烈，且並不是一直持續往下降，推測是到後期Discriminator訓練得太強了。

3. Plot 32 random generated images.

4. Discuss what you've observed and learned from implementing GAN.
   (a) 大約在第五個epoch左右，生成出來的圖片就能明顯看到人臉，至於局部細節的部分則需要更多時間讓model去學習如何產生更好的圖片來騙過Discriminator。
   (b) 在training時我有試著讓每張training data都水平翻轉一次，也就是用兩倍的data數量來訓練我的model，但效果跟沒有水平flip的結果比起來並沒有太大的差異。

(c) 疊model時必須要讓Generator和Discriminator有差不多的結構，因為兩者之間必須來進行對抗，所以架構上不能相差太大。

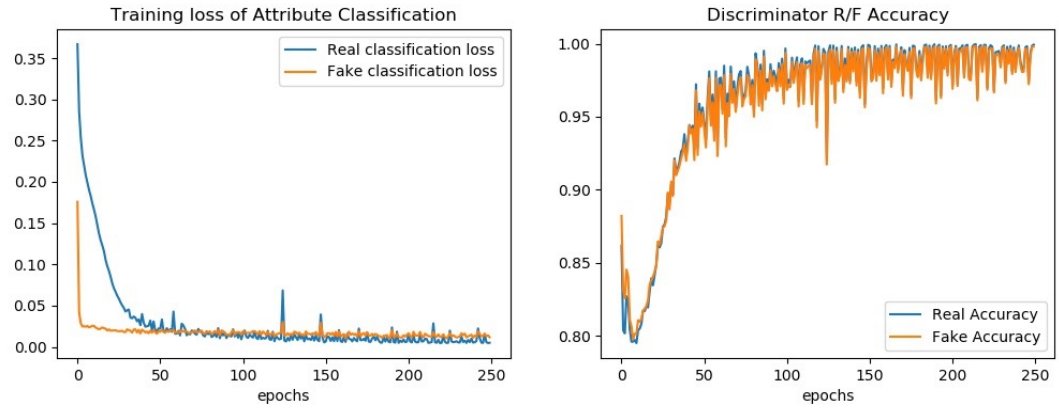
5. Compare the difference between image generated by VAE and GAN, discuss what you've observed.
   和VAE相比，GAN生成的圖片明顯較為清楚，同時生出的人臉彼此之間變化也較大，顏色、表情、形狀也更為豐富，在調seed的時候可以明顯體會到這個結果。
   此外，由於必須要騙過Discriminator，且VAE必須要在MSE和KLD兩者之間作取捨，因此在人臉細節的部分，GAN的表現要比VAE好上許多，也不會那麼模糊。

Problem 3. ACGAN

1. Describe the architecture & implementation details of your model.
   I choose 100 as the latent dimension, and use tanh as the activation function in the last layer. After tanh, the output will be divided by 2 and plus 0.5 so that it'll be normalized to (0,1). For each pixel of the training images, I divide it by 255 so the input range is also between (0,1).
   Besides, I add 1 class to the latent dimension, so the input dim of the Generator is 101.

```
ACGenerator(
  (decoder): Sequential(
    (0): ConvTranspose2d (101, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
    (3): ConvTranspose2d (512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (5): ReLU(inplace)
    (6): ConvTranspose2d (256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (8): ReLU(inplace)
    (9): ConvTranspose2d (128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (11): ReLU(inplace)
  )
  (output): Sequential(
    (0): ConvTranspose2d (64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): Tanh()
  )
)
ACDiscriminator(
  (main): Sequential(
    (0): Conv2d (3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, inplace)
    (3): Conv2d (64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (5): LeakyReLU(0.2, inplace)
    (6): Conv2d (128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (8): LeakyReLU(0.2, inplace)
    (9): Conv2d (256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (11): LeakyReLU(0.2, inplace)
  )
  (out_dis): Sequential(
    (0): Conv2d (512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
  (out_aux): Sequential(
    (0): Conv2d (512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
)
```

2. Plot the learning curve of the model and briefly explain what you think it represents.



在train ACGAN時，必須同時考量Generator對於同一個attribute的生成能力以及Discriminator的判別能力。在這裡我們也可以看到在一百的epoch之後Discriminator的判別能力已有相當水準，且loss也降得很低。在訓練初期，由於Discriminator是從real image抓取重要資訊來學習判斷圖片真偽的能力，因此loss下降的趨勢相較於fake要來得緩慢。

3. Plot 10 pair of random generated images of your model, each pair generated from the same random vector input but with different attribute.

我所取的feature是smiling，圖中上排和下排分別是沒笑和有笑的結果。