DLCV HW4

B03901009 電機四 郭笛萱

Problem1

1 Describe the architecture & implementation details of your model

Encoder 主要是由四層的 conv2D 所組成，每經過一層 conv2D，輸出的深度變深一倍，大小變 1/4，最後輸出 512 **x 4 x 4 的矩陣**。中間的 activation function 都是 leaky relu, 避免負數的資訊遺漏。

Latent space 的大小選用 512

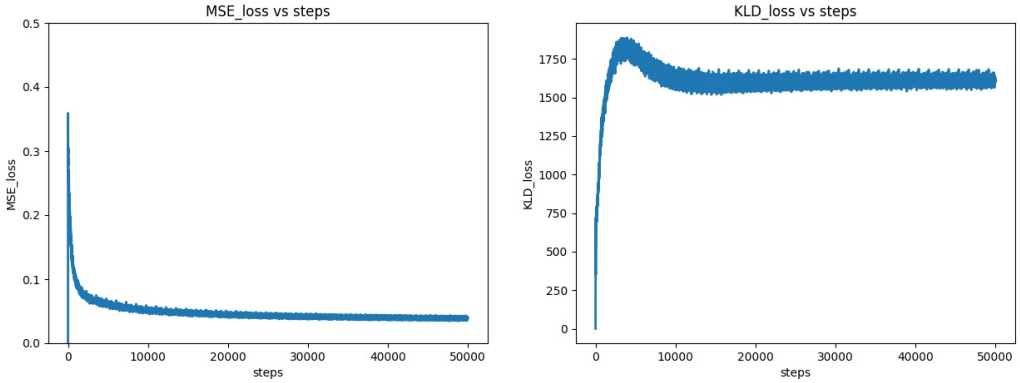
Decoder 主要是由四層的 convTranspose2D 所組成，每經過一層 transpose, 深度變淺一半，大小變大 4 倍，最後一層再將 channel 個數變為 3。

經過測試，發現用$\lambda_{KL} = 1e - 5$ 的效果最好，testing 圖片不會太模糊，reconstruct 出來的影像也看起來像個人臉

training epoch = 40, batch_size = 32

autoencoder(
 (encoder): Sequential(
  (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (1): LeakyReLU(0.2, inplace)
  (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (4): LeakyReLU(0.2, inplace)
  (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (6): LeakyReLU(0.2, inplace)
  (7): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (8): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
  (9): LeakyReLU(0.2, inplace)
)
 (decoder): Sequential(
  (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (2): ReLU(inplace)
  (3): ConvTranspose2d(256, 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): ReLU(inplace)
  (6): ConvTranspose2d(128, 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): ReLU(inplace)
  (9): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (10): Tanh()
)
 (fcn11): Linear(in_features=8192, out_features=512, bias=True)
 (fcn12): Linear(in_features=8192, out_features=512, bias=True)
 (fcn2): Linear(in_features=512, out_features=8192, bias=True)
)

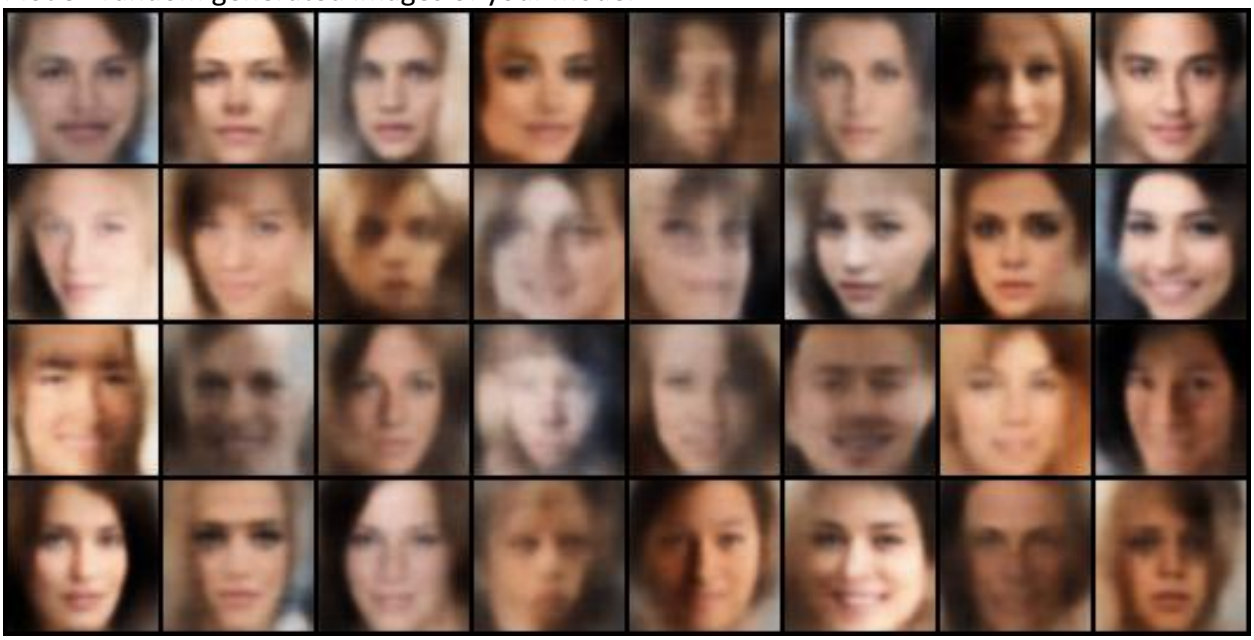
2 Plot the learning curve of your model



3 Plot 10 test images & their reconstructed result of your model and report your testing MSE of the entire test set

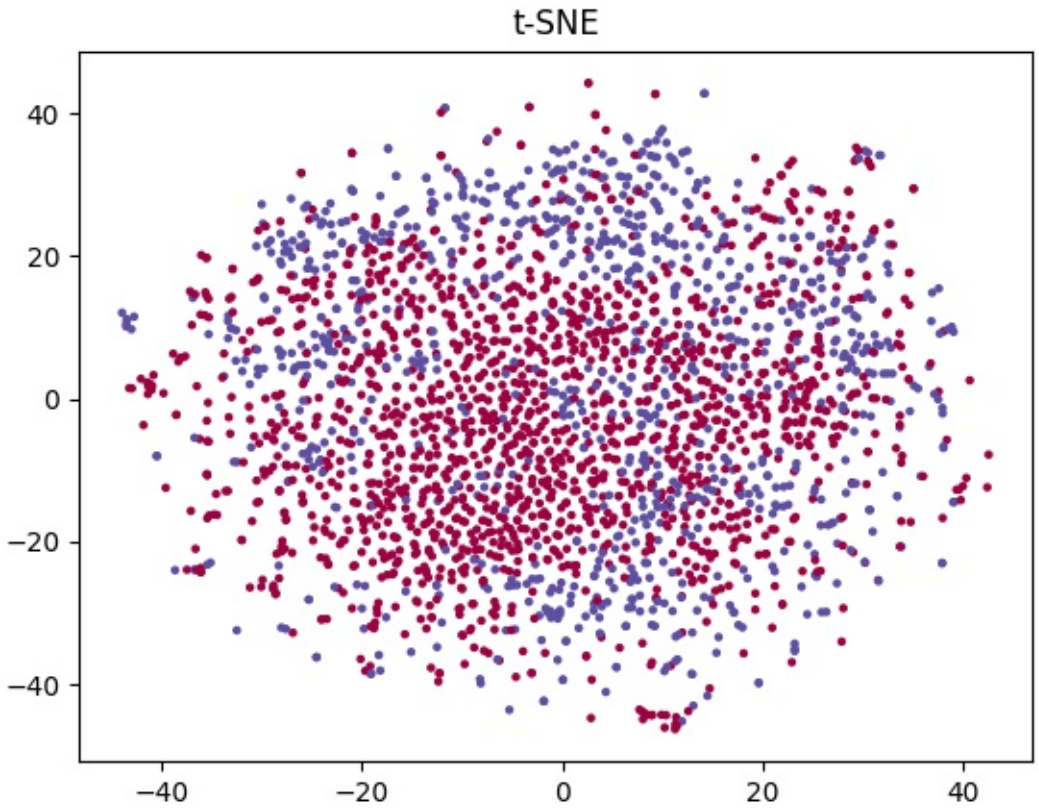
test loss of the entire test set = 0.03219

4 Plot 32 random generated images of your model



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

我選擇的分類項目是 Male = 1 or 0
紫色是 男生
紅色是 女生
可以看到 VAE 並無法將兩者分類的很好，因為我們沒有特別輸入 attribute label 進去，所以很難學到不同 label 之間的差別



6 Discuss what you've observed and learned from implementing VAE
本次作業的 VAE 有參考網路上的資料：
https://github.com/pytorch/examples/blob/master/vae/main.py

我一開始先寫了基本的 autoencoder 架構，確定能夠 work 後才改成 VAE，但是卻發現 reconstruct 出來的 image 幾乎都是雜訊。後來發現是我的 model 好像寫得不太對，encoder 應該是要把 input 變得越來越深，但是我原本寫的卻是把 input 變得越來越淺，推測是後期 filter 個數不夠多，造成學得不夠好，所以即使 autoencoder 可以 work，VAE 卻還是沒學得很好。

Problem2

1 Describe the architecture & implementation details of your model

Generator 的 input 是 100 x 1 x 1 的 noise vector，經過五層的 convTranspose2D 後，變成一張影像。

Discriminator 的 input 是一張影像，輸出成一個一維的 label，最後經過 softmax。

Loss 分為三項，分別是餵真實影像給 discriminator，判斷輸出的 label 與真實 label 的 loss，再來是餵假的影像給 discriminator，判斷輸出的 label 與假的 label 的 loss，最後是 discriminator 餵假影像後，產出的 label 與真 label 的 loss。這三個 loss 都是使用 binary cross entropy

Train epoch = 30, batch_size = 32

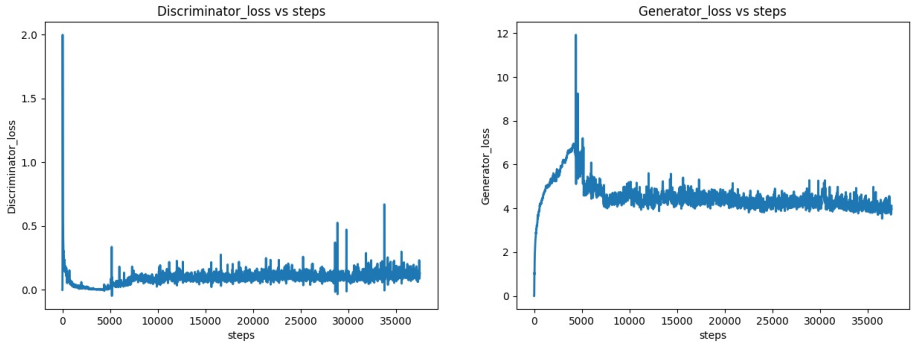
GAN_generator(

```
  (generator): 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)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```

GAN_discriminator(

```
  (discriminator): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(0.2, inplace)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (4): LeakyReLU(0.2, inplace)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (7): LeakyReLU(0.2, inplace)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (10): LeakyReLU(0.2, inplace)
    (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (12): Sigmoid()
  )
)
```

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



Generator 一開始的 loss 比較小，因為經過 generator 產生的圖還是雜訊，所以 discriminator 很容易判斷出這是假的照片，之後隨著 step 增加，loss 會趨於穩定 Discriminator 的 loss 因為包含了兩項所以比較難看出趨勢，大致上是呈穩定的狀態。

3 Plot 32 random generated images of your model

4 Discuss what you've observed and learned from implementing GAN

我學到 generator 跟 discriminator 兩者要同時一起變低才能夠 train 出好的 model。一開始在寫的時候，會發現這兩個沒有一起下降，而是有個降得特別快，另一個幾乎沒下降，就知道這個一定是哪裡有問題，造成一邊一直無法成功產生好的影像，另一邊才能輕鬆地分別出這是假的影像的情形。

5 Compare the difference between image generated by VAE and GAN, discuss what you've observed

VAE 生成的影像較 GAN 來得模糊，原因是因為 VAE 的 Loss 主要是由 MSE 來控制，所以造成輸出都是平均後的結果，無法產生清晰的影像。而 GAN 所產生的影像因為得騙過 discriminator，所以如果只是產生模糊的影像一下子就會被分類為假的，所以他有能力產生較為銳利的影像

Problem3

1 Describe the architecture & implementation details of your model

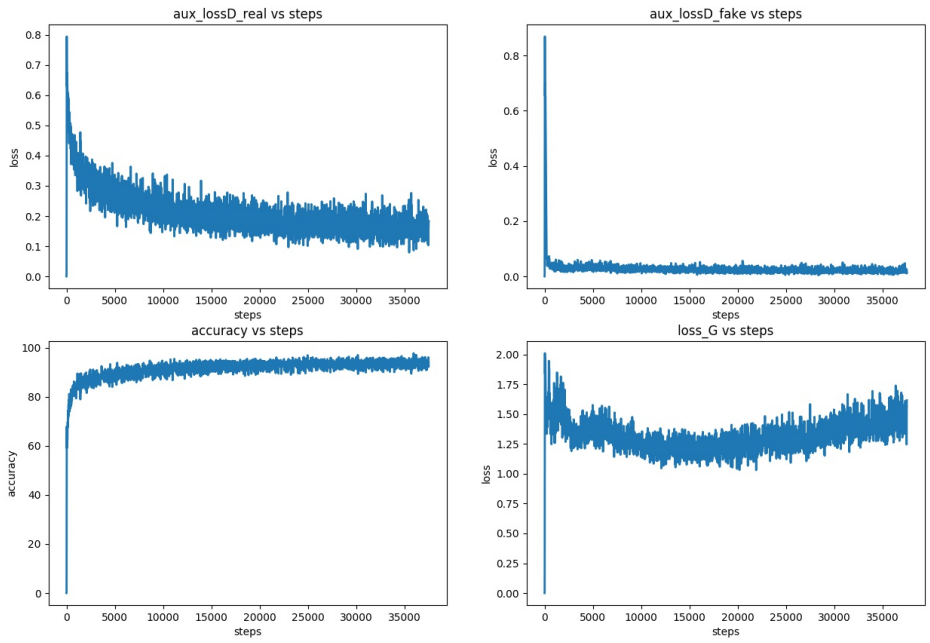
ACGAN_generator(

```
  (generator): Sequential(
    (0): ConvTranspose2d(512, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): LeakyReLU(0.2, 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): LeakyReLU(0.2, 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): LeakyReLU(0.2, 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): LeakyReLU(0.2, inplace)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
  (fc1): Linear(in_features=102, out_features=512, bias=True)
)
```

ACGAN_discriminator(

```
  (discriminator): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(0.2, inplace)
    (2): Dropout(p=0.5, 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): Dropout(p=0.5, inplace)
    (7): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (8): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (9): LeakyReLU(0.2, inplace)
    (10): Dropout(p=0.5, inplace)
    (11): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (12): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (13): LeakyReLU(0.2, inplace)
    (14): Dropout(p=0.5, inplace)
  )
  (fc1): Linear(in_features=8192, out_features=1, bias=True)
  (fc2): Linear(in_features=8192, out_features=2, bias=True)
  (sigmoid): Sigmoid()
  (softmax): Softmax()
)
```

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



- 左上角： 輸入真實影像進 discriminator，計算真實的 attribute label 與 predict 出來的 loss，可以看到 loss 會隨著時間下降
- 右上角： 輸入假影像進 discriminator，計算 random attribute label 與 predict 出來的 loss，可以看到 random label 因為是符合 normal distribution，所以似乎較真實姿繚還容易學
- 左下角：在 training 過程中，discriminator predict 出的 label 相對於真實 label 的準確度，跟左上角的 loss 的結果類似，在早期 steps 就有趨於飽和的現象。
- 右下角：反應出 generator 用 noise 產生出騙過 discriminator 影像的能力，大致上隨著時間增加而下降，產生假圖片的能力越來越高。

3 Plot 10 pair of random generated images of your model, each pair generated from the same random vector input but different attribute. This is to demonstrate your model's ability to disentangle feature of interest.

上圖選用的 attribute 是男生與女生