Deep Learning for Computer Vision

Homework4

R06942052 電信碩一 鍾勝隆

Problem 1: VAE

1. Describe the architecture & implementation details of your model

I used 7 layers of Convolution for encoder and decoder in VAE respectively. Map the image (64*64*3) into mean vector and std vector(1024*2). The following is the pytorch output of the architecture.

```
VAE(
  (encoder): Encoder(
    (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (conv4): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (bn4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (conv5): Conv2d(512, 1024, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (bn5): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
    (conv6): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (bn6): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
    (convm): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
    (bnm): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
    (convv): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
    (bnv): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True))

  (decoder): Decoder(
    (convtrans1): ConvTranspose2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
    (convtrans2): ConvTranspose2d(1024, 1024, kernel_size=(2, 2), stride=(1, 1))
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
    (convtrans3): ConvTranspose2d(1024, 512, kernel_size=(3, 3), stride=(1, 1))
    (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
```

(convtrans4): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))

(bn4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)

(convtrans5): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))

(bn5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)

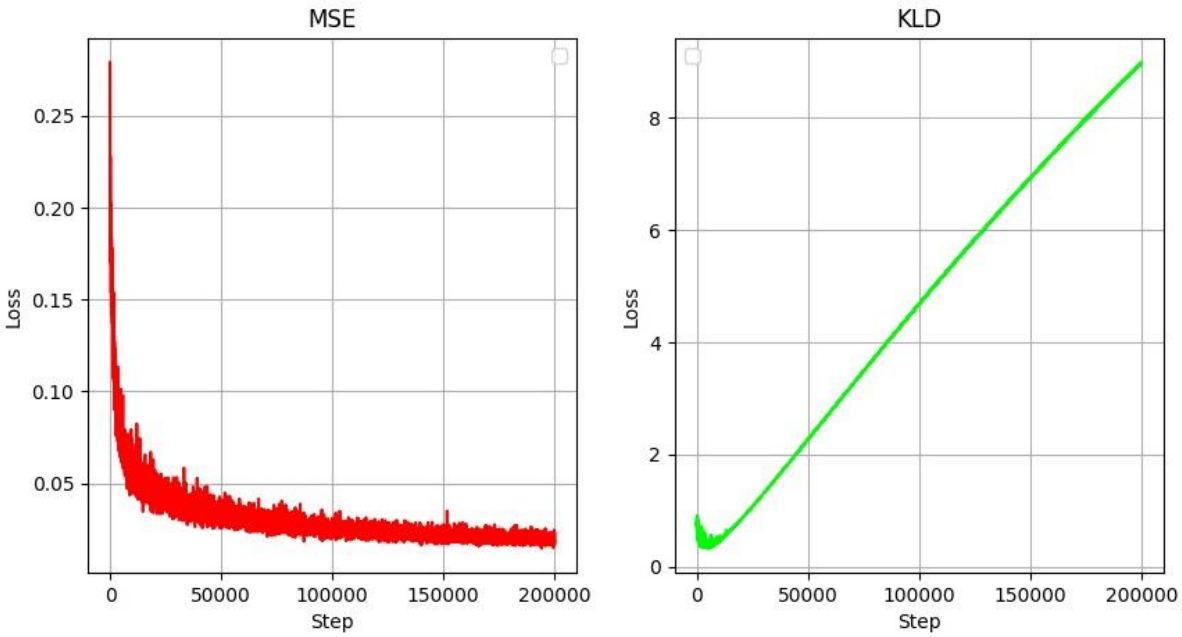
(convtrans6): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))

(bn6): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)

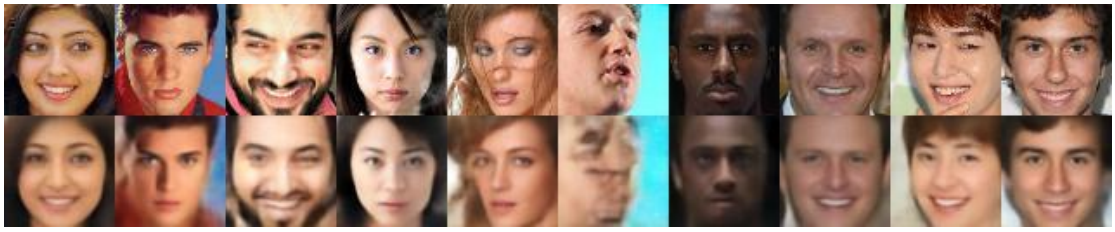
(convtrans7): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))

(bn7): BatchNorm2d(3, eps=1e-05, momentum=0.1, affine=True)))

2. Plot the learning curve of your model



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

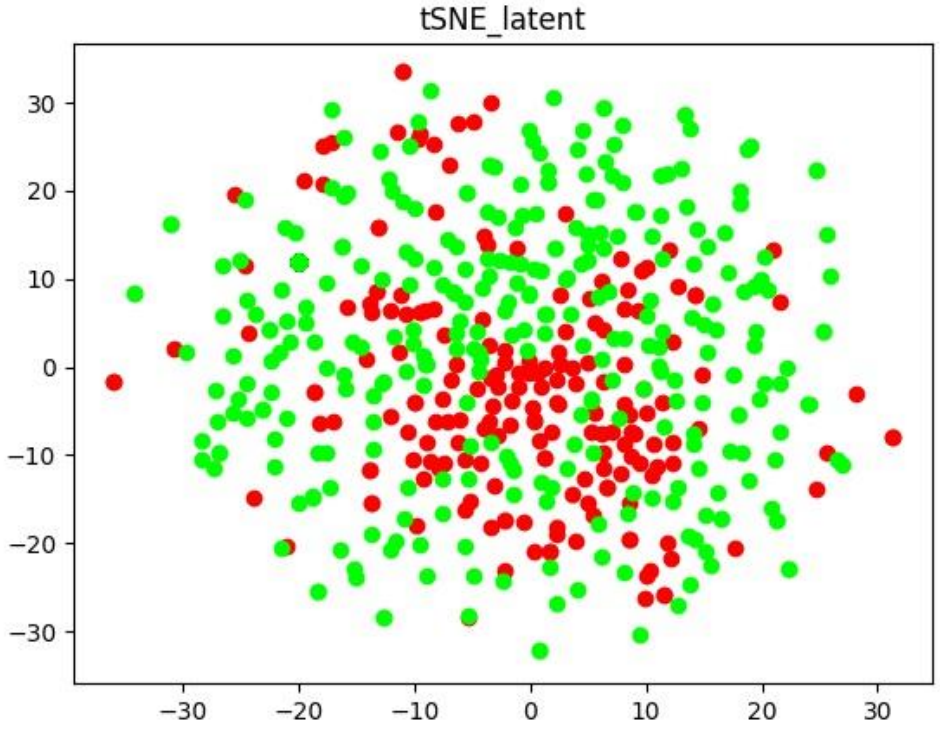


Average MSE loss = 292.5680

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

   The attribute I chose is ‘Smiling’, red dots represent those smiling faces, and green ones represent those are not smiling.



6. Discuss what you’ve observed and learned from implementing VAE.

   The images tend to be blurred whether they are reconstructed or

random generated. It's affected by the MSE loss which makes image have less high frequency detail.

Problem 2: GAN

1. Describe the architecture & implementation details of your model

I used 5 layers of Convolution for generator and discriminator in GAN and used Binary Cross Entropy as the loss function of real and fake image. Moreover, I rescaled the image value to -1~1 and use tanh as the output layer of the generator. I also implement soft labeling which makes the image label (0.7~1.0/0.0~0.3) instead of (1/0).

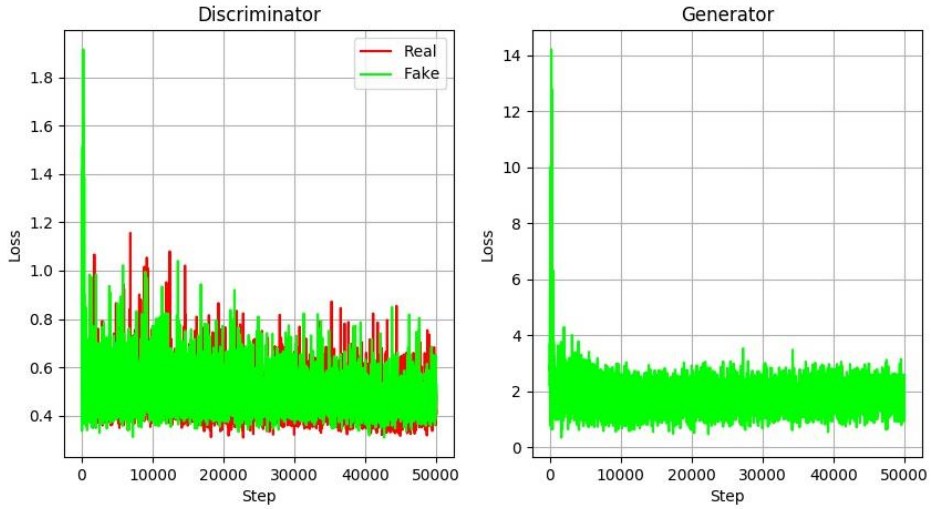
```
Generator(
  (convtrans1): ConvTranspose2d(100, 1024, kernel_size=(4, 4), stride=(1, 1))
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
  (convtrans2): ConvTranspose2d(1024, 512, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1))
  (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
  (convtrans3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1))
  (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (convtrans4): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1))
  (bn4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (drop1): Dropout(p=0.5)
  (convtrans5): ConvTranspose2d(128, 3, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1))
  (drop2): Dropout(p=0.5))
```

```
Discriminator(
  (conv1): Conv2d(3, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
  (conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (bn4): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
```

(conv5): Conv2d(1024, 1, kernel_size=(4, 4), stride=(1, 1)))

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

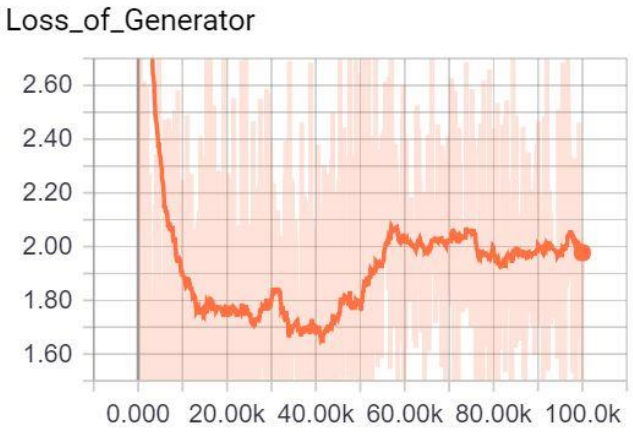
   The learning dropped quickly but stayed noisy as the step increase. I think it is showing the generator and discriminator are competing to each other. Thus, the learning curve cannot be smooth as normal CNN method.



3. Plot 32 random generated images of your model

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

   If I training GAN with more steps, mode collapse will occur. The learning curve of generator (around 50k steps) climb up suddenly and the random generated faces will only be one to two types, shown in following figs.



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

   Those faces from VAE is blurrier the those from GAN. The edge of image from CelebA dataset is various because of the variety of the face poses. Thus, VAE tends to make the region with more variety to much blurrier with its MSE loss function. On the other hand, GAN is not constrained by the MSE, so it won't tend to produce blurry images. However, GAN might generate some faces won't be considered as faces (like the face with red frame in Fig.2_3). The stability of the learning curve also shows the tendency of two methods.

Problem 3: ACGAN

1. Describe the architecture & implementation details of your model

   I mainly copied the structure in GAN. The 'Smiling' class information is encoded into one-hot format (ex: [1,0] for smiling, [0,1] for not smiling) and concatenated to the random generated vector. Generator takes the concatenated vector as input to produce faces. The loss function I used for classification is negative log likelihood loss function.

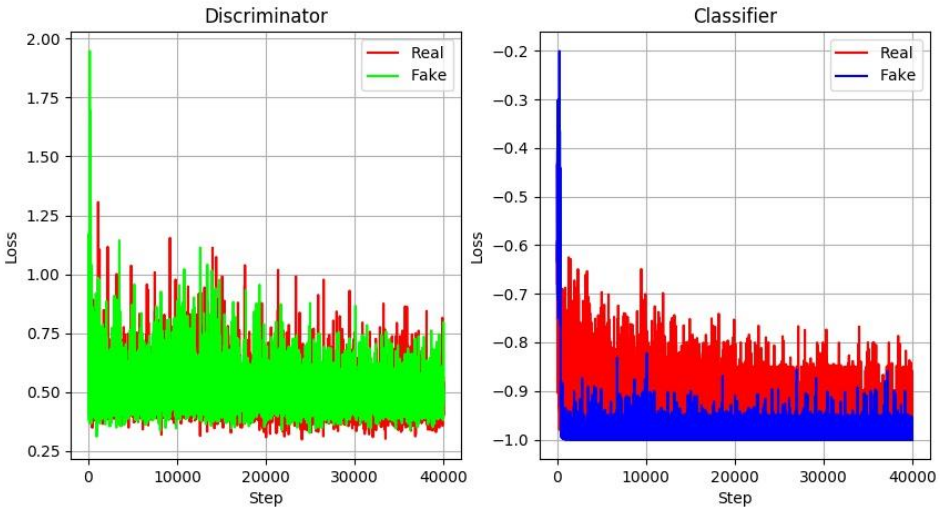
```
Generator(
  (convtrans1): ConvTranspose2d(100, 1024, kernel_size=(4, 4), stride=(1, 1))
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
  (convtrans2): ConvTranspose2d(1024, 512, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1))
```

(bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
(convtrans3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
(convtrans4): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(bn4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
(drop1): Dropout(p=0.5)
(convtrans5): ConvTranspose2d(128, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(drop2): Dropout(p=0.5))

Discriminator(
(conv1): Conv2d(3, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
(conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
(conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
(conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
(bn4): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True)
(conv5): Conv2d(1024, 1, kernel_size=(4, 4), stride=(1, 1)))

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

The discriminator is like the one in GAN case. real and fake data are



fooling the generator which cause the noisy loss. The classifier works better in the fake data in the beginning of the training steps. I think the reason is because the initial input is random but with distinct attribute for ‘smiling’. Thus, the classifier has less loss in those fake face generate by ACGAN.

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

   The upper row is for not smiling, the lower one is for smiling.