DLCV HW4 Report
b04507009 電機三 何吉瑞

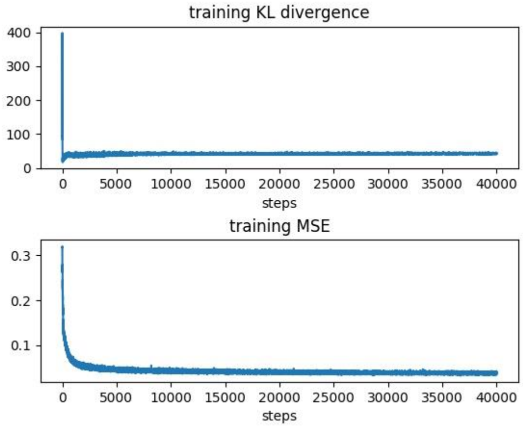
Problem1 VAE(6%)

1.

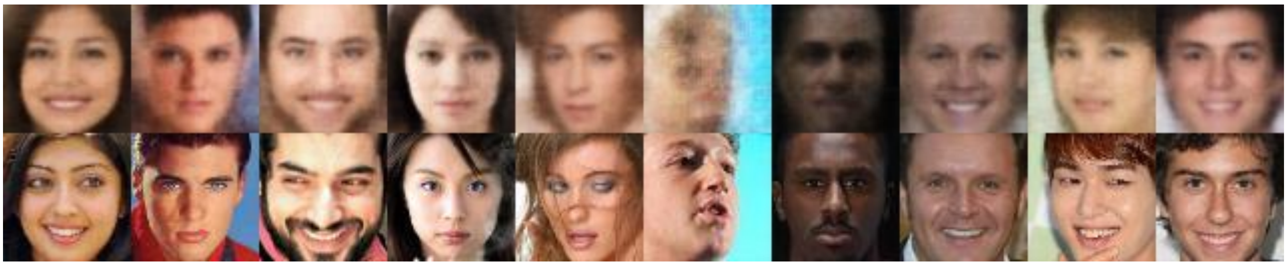
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]            1568
       BatchNorm2d-2           [-1, 32, 32, 32]              64
         LeakyReLU-3           [-1, 32, 32, 32]               0
            Conv2d-4           [-1, 64, 16, 16]           32832
       BatchNorm2d-5           [-1, 64, 16, 16]             128
         LeakyReLU-6           [-1, 64, 16, 16]               0
            Conv2d-7            [-1, 128, 8, 8]          131200
       BatchNorm2d-8            [-1, 128, 8, 8]             256
         LeakyReLU-9            [-1, 128, 8, 8]               0
           Conv2d-10            [-1, 256, 4, 4]          524544
      BatchNorm2d-11            [-1, 256, 4, 4]             512
        LeakyReLU-12            [-1, 256, 4, 4]               0
           Linear-13                  [-1, 512]         2097664
           Linear-14                  [-1, 512]         2097664
           Linear-15                 [-1, 4096]         2101248
  ConvTranspose2d-16            [-1, 128, 8, 8]          524288
      BatchNorm2d-17            [-1, 128, 8, 8]             256
        LeakyReLU-18            [-1, 128, 8, 8]               0
  ConvTranspose2d-19           [-1, 64, 16, 16]          131072
      BatchNorm2d-20           [-1, 64, 16, 16]             128
        LeakyReLU-21           [-1, 64, 16, 16]               0
  ConvTranspose2d-22           [-1, 32, 32, 32]           32768
      BatchNorm2d-23           [-1, 32, 32, 32]              64
        LeakyReLU-24           [-1, 32, 32, 32]               0
  ConvTranspose2d-25            [-1, 3, 64, 64]            1536
             Tanh-26            [-1, 3, 64, 64]               0
================================================================
Total params: 7677792
Trainable params: 7677792
Non-trainable params: 0
----------------------------------------------------------------
```

1^{st} to 12^{th} layers are encoding layers and 16^{th} to 26^{th} layers are decoding layers. The 15^{th} layer is the result of reparameterization, corresponding to $z = \mu + \varepsilon \times e^{0.5\times \mathrm{logvar}}$
I set the latent dimension as 512. I normalize the image to [-1,1] here and following cases.
Moreover, I set the batch size as 100 and ran 100 epochs.
For reconstruction, the input and output are both (64,64,3). For random generation, the input is random noise sized of (100,1,1) and output is (64,64,3).
Moreover, loss function is the combination of MSE and KL divergence, and the parameter lambda that strikes a balance between them is 1 in my model. The optimizer is Adam with lr=0.001

2.



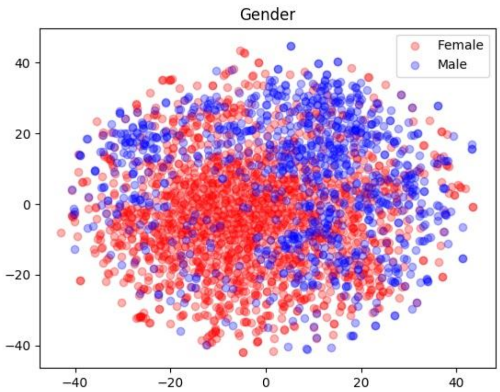
The scale is not clear in figure. The KL divergence is about 40 during the training process
3.



MSE on testing set: 215.99, Consider the normalization, 2.249 per pixel

4.

5.



I sampled all the images in testing set, and the result roughly shows the difference between two genders. They clustered at different location which can be discriminated.

6.

(1) The parameter lambda is important for training. If the MSE is over weighted, the reconstruction would be very similar to the original input images. However, the random generation would be very noisy. On the contrast, if the KLD is overweighted, there would be a big difference between the reconstruction images and original images.

(2) After striking a balance between reconstruction and random generation, I found out that the results are blurry compared with the result of GAN, which matches the theory.

(3) In my cases, the size of latent dimension and batch size didn't affect the performance obviously.

Problem2 GAN(5%)

1.

Generator:

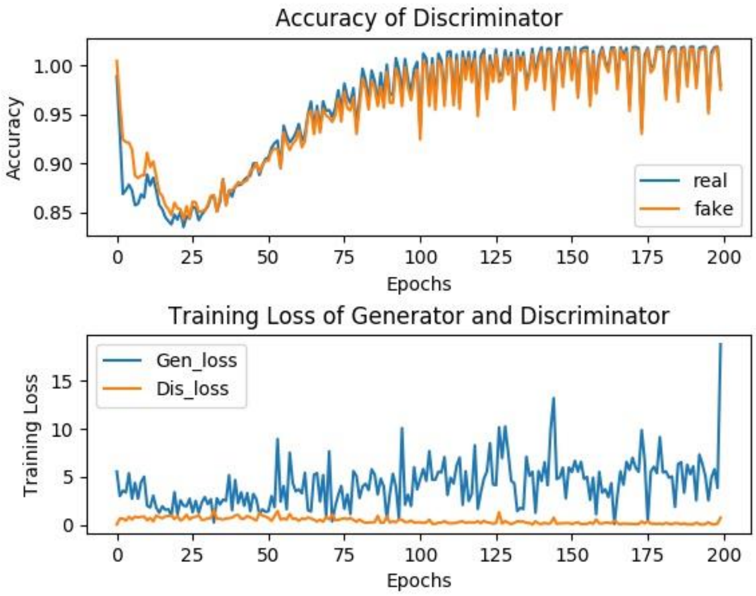
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         819200
       BatchNorm2d-2            [-1, 512, 4, 4]           1024
              ReLU-3            [-1, 512, 4, 4]              0
   ConvTranspose2d-4            [-1, 256, 8, 8]        2097152
       BatchNorm2d-5            [-1, 256, 8, 8]            512
              ReLU-6            [-1, 256, 8, 8]              0
   ConvTranspose2d-7          [-1, 128, 16, 16]         524288
       BatchNorm2d-8          [-1, 128, 16, 16]            256
              ReLU-9          [-1, 128, 16, 16]              0
  ConvTranspose2d-10           [-1, 64, 32, 32]         131072
      BatchNorm2d-11           [-1, 64, 32, 32]            128
             ReLU-12           [-1, 64, 32, 32]              0
  ConvTranspose2d-13            [-1, 3, 64, 64]           3072
             Tanh-14            [-1, 3, 64, 64]              0
================================================================
Total params: 3576704
Trainable params: 3576704
Non-trainable params: 0
----------------------------------------------------------------
```

Discriminator:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           3072
       BatchNorm2d-2           [-1, 64, 32, 32]            128
         LeakyReLU-3           [-1, 64, 32, 32]              0
            Conv2d-4          [-1, 128, 16, 16]         131072
       BatchNorm2d-5          [-1, 128, 16, 16]            256
         LeakyReLU-6          [-1, 128, 16, 16]              0
            Conv2d-7            [-1, 256, 8, 8]         524288
       BatchNorm2d-8            [-1, 256, 8, 8]            512
         LeakyReLU-9            [-1, 256, 8, 8]              0
           Conv2d-10            [-1, 512, 4, 4]        2097152
      BatchNorm2d-11            [-1, 512, 4, 4]           1024
        LeakyReLU-12            [-1, 512, 4, 4]              0
           Conv2d-13              [-1, 1, 1, 1]           8192
          Sigmoid-14              [-1, 1, 1, 1]              0
================================================================
Total params: 2765696
Trainable params: 2765696
Non-trainable params: 0
----------------------------------------------------------------
```

I set the latent dimension as 100 and the max number of filters as 512. Moreover, I set the batch size as 200 and ran 200 epochs.
For the Generator, the input is a random noise sized of (100,1,1) and the output is an image sized of (64,64,3) whose value is bounded in [-1,1]
For the Discriminator, the input is an image sized of (64,64,3) and the output is the prediction whose value is in [0,1].
Loss function: binary cross entropy loss
Optimizer: Both use Adam with lr=0.0001 and betas=(0.5, 0.999)

2.



The accuracy of real and fake images for discriminator are close. Moreover, compared to the loss of discriminator, the loss of generator is unstable.

3.

4.

(1) Sometimes it came out that all the images looked same, which may due to mode collapse. I thus adjust some of parameters (ex: for the optimizer) to avoid this problem in the model I submit

(2) Some tips like adding dropout and using soft labels do help in the beginning steps. However, after enough many epochs, the results are similar.

(3) After about 10 epochs, it showed primary results. However, it still takes tens of epochs to learn the detail information of faces. Moreover, the difference is little between the results from after 100 epochs.

(4) Applying data augmentation by flipping the images does enhance the performance.

(5) Sometimes black faces would be generated.

5.

(1) Compared with the random generated images in VAE, those in GAN has higher image quality.

(2) When changing the random seed, I also found out that GAN has higher diversity than VAE does.

(3) In case that image is distorted, that in GAN is also more serious than in VAE.

(4) In VAE, the backpropagation is from the comparison with the real image, which is supervised learning. In GAN, parameters updating in the generator is not triggered by the data but use the loss of discriminator, which is semi-supervised learning

Problem3 ACGAN(4%)

1.

Generator:

Layer (type)	Output Shape	Param #
ConvTranspose2d-1	[-1, 512, 4, 4]	819200
BatchNorm2d-2	[-1, 512, 4, 4]	1024
ReLU-3	[-1, 512, 4, 4]	0
ConvTranspose2d-4	[-1, 256, 8, 8]	2097152
BatchNorm2d-5	[-1, 256, 8, 8]	512
ReLU-6	[-1, 256, 8, 8]	0
ConvTranspose2d-7	[-1, 128, 16, 16]	524288
BatchNorm2d-8	[-1, 128, 16, 16]	256
ReLU-9	[-1, 128, 16, 16]	0
ConvTranspose2d-10	[-1, 64, 32, 32]	131072
BatchNorm2d-11	[-1, 64, 32, 32]	128
ReLU-12	[-1, 64, 32, 32]	0
ConvTranspose2d-13	[-1, 3, 64, 64]	3072
Tanh-14	[-1, 3, 64, 64]	0

Total params: 3576704
Trainable params: 3576704
Non-trainable params: 0

Discriminator:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	3072
BatchNorm2d-2	[-1, 64, 32, 32]	128
LeakyReLU-3	[-1, 64, 32, 32]	0
Conv2d-4	[-1, 128, 16, 16]	131072
BatchNorm2d-5	[-1, 128, 16, 16]	256
LeakyReLU-6	[-1, 128, 16, 16]	0
Conv2d-7	[-1, 256, 8, 8]	524288
BatchNorm2d-8	[-1, 256, 8, 8]	512
LeakyReLU-9	[-1, 256, 8, 8]	0
Conv2d-10	[-1, 512, 4, 4]	2097152
LeakyReLU-11	[-1, 512, 4, 4]	0
Conv2d-12	[-1, 1, 1, 1]	8192
Conv2d-13	[-1, 1, 1, 1]	8192

Total params: 2772864
Trainable params: 2772864
Non-trainable params: 0

The architecture is similar to that in GAN, only adding a discriminator in 13th layer to discriminate the class. Moreover, I set the batch size as 200 and ran 200 epochs.

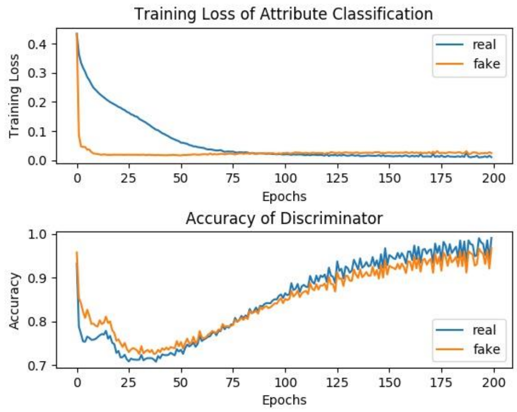
For the Generator, the input is a random noise sized of (100,1,1) and the output is an image sized of (64,64,3) whose value is bounded in [-1,1]

For the Discriminator, the input is an image sized of (64,64,3) and the output are the prediction of the class and real/fake whose values are in [0,1].

Loss function: binary cross entropy loss

Optimizer: Both use Adam with lr=0.0001 and betas=(0.5, 0.999)

2.



The loss of classification in fake images decays much faster than that in real images. Moreover, the accuracy of discriminating the image is close between real and fake images during the training process.

3.

There’s an obvious difference between the images of two rows. The second row is the smiling version of the first row. Some features like the shape of mouth, teeth, eyesight are quite obvious to show that how ACGAN works.