IMVFX Assignment 2-1 DCGAN Practice

資工科碩 312551080 紀軒宇

A.2-1 Please provide a brief introduction about your experiments, including details such as setting of hyperparameter, data augmentation techniques used, network structure, etc. (5%)

2-1-1 Setting of hyperparameter

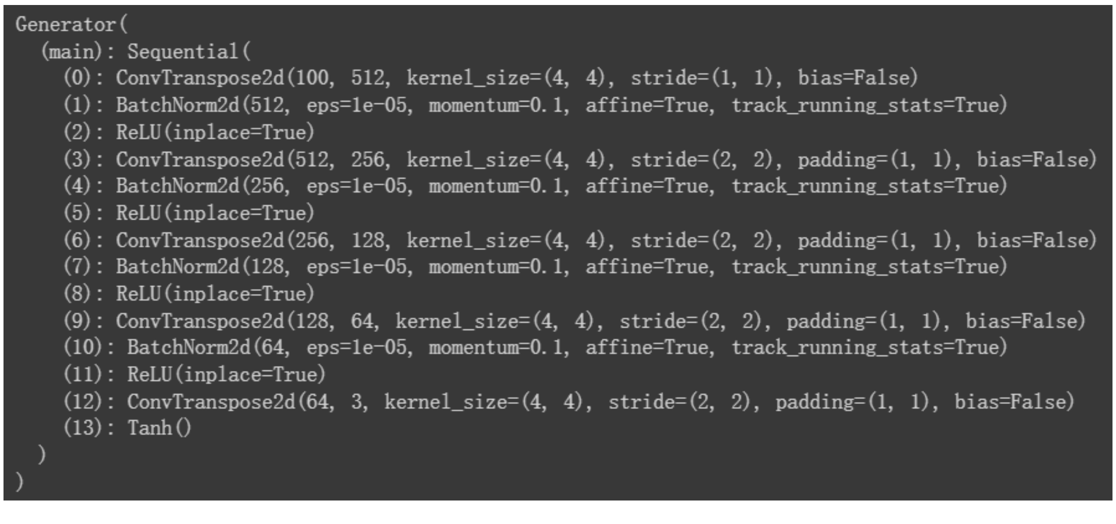
In our experiment, the hyperparameter we set is shown as below:

- batch_size: 128
  the batch size during training
- image_size: 64
  the spatial size of training image
- nc: 3
  number of channel of the training image
- nz: 100
  size of z vector
- ngf: 64
  size of feature map in generator
- ndf: 64
  size of feature map in discriminator
- num_epochs: 200
  number of training epochs
- lr: 0.0002
  learning rate of optimizers
- beta1: 0.5
  hyperparameter of Adam optimizers

2-1-2 Network structure

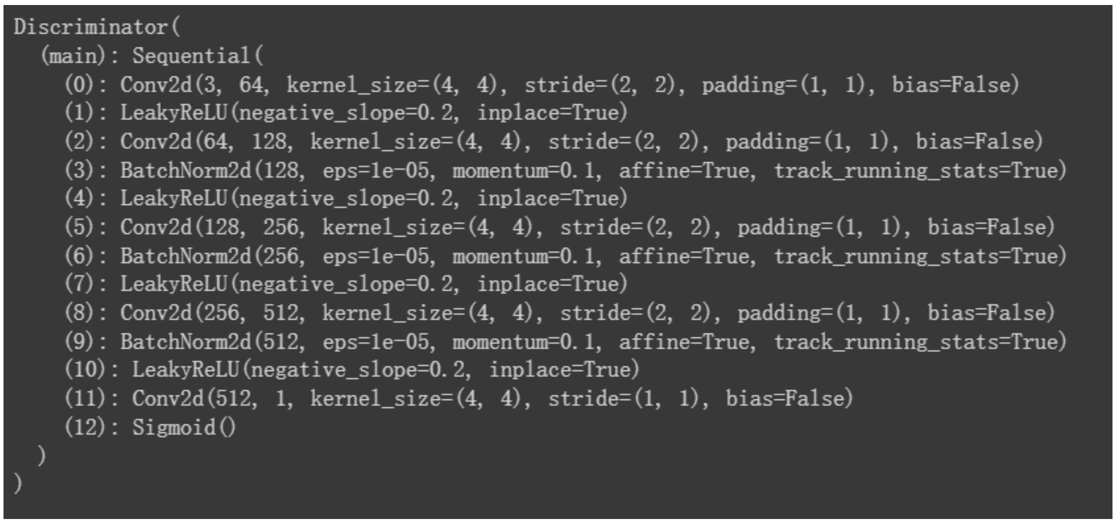
Our network have two parts, Generator and Discriminator, the structure are shown as below.

2-1-2-1 Generator



```
Generator(
  (main): 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, track_running_stats=True)
    (2): ReLU(inplace=True)
    (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, track_running_stats=True)
    (5): ReLU(inplace=True)
    (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, track_running_stats=True)
    (8): ReLU(inplace=True)
    (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, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```

2-1-2-2 Discriminator



```
Discriminator(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (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, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (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, track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (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, track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.2, inplace=True)
    (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (12): Sigmoid()
  )
)
```

2-1-2-3 Other techniques used

- Weight Initialization

  Here we use these rules to initial the weights

  - Conv
    - weight: normal distribution of mean=0, std=0.02
  - BatchNorm
    - weight: normal distribution of mean=1.0, std=0.02
    - bias: 0
- Optimizers

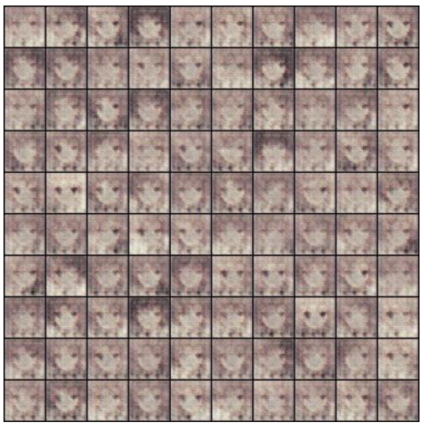
  Here, I use Adam optimizers

A.2-2 Place the generated image series from various epochs during the training process here and provide a discussion of your observations. (5%)

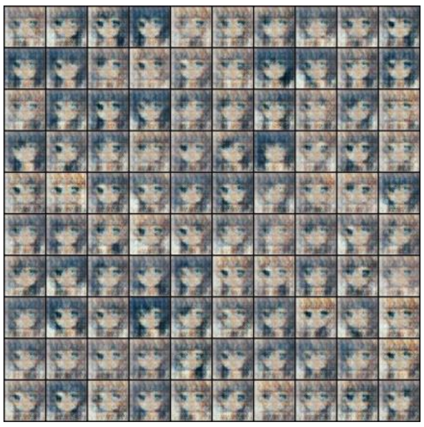
We can divide the training process into several stages:

1. Early stage (Epoch 1~3)

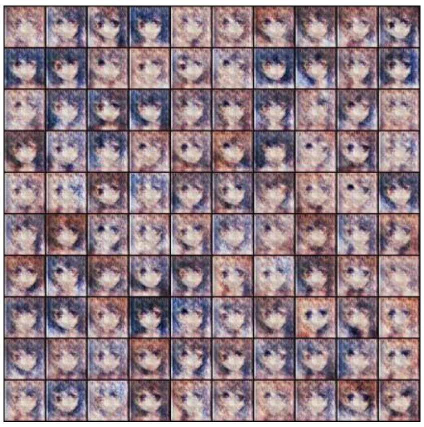
   From the beginning of the training, the image generated after epoch 1 training start to generate something with a lot of noise, take a closer look we can observe some features starts to appear, such as two black spots in the middle of the image, but we can’t obviously tell what is that, and the first few epochs, the images generated in the same epoch seems don’t have much different with each other.



Epoch 1



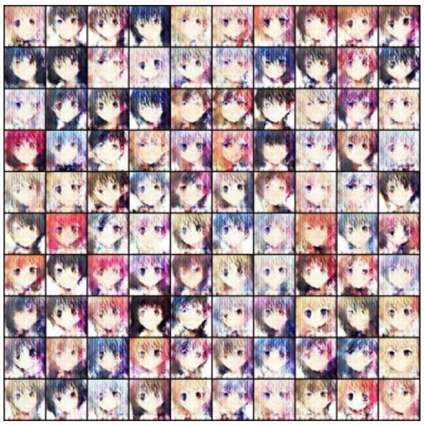
Epoch 2

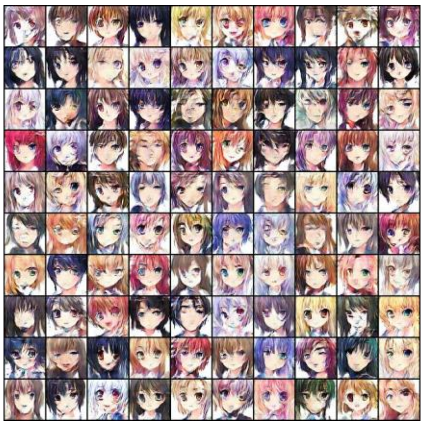


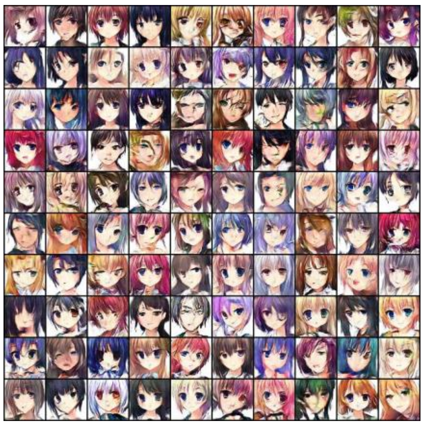
Epoch 3

2. Middle stage (Epoch 10~50)

   In the middle stage of training, at epoch 10, it starts to have the prototype of the anime face with features such as hair, eyes, but with some obvious noise on their face, after another 20 epochs, the noise is getting fewer and being more like an anime face, which have hair and eyes in different color but not natural, when in epoch 50, it only have few noises on the face part, and start to have some high-quality anime face generated.


Epoch 10


Epoch 30


Epoch 50

3. Late stage (Epoch 100~200)

   After 100 epochs, the quality of the output image is getting higher and higher. In this stage images between epoch don't change as much as the previous two stage, in my experiment, the train finally stop at epoch 200.

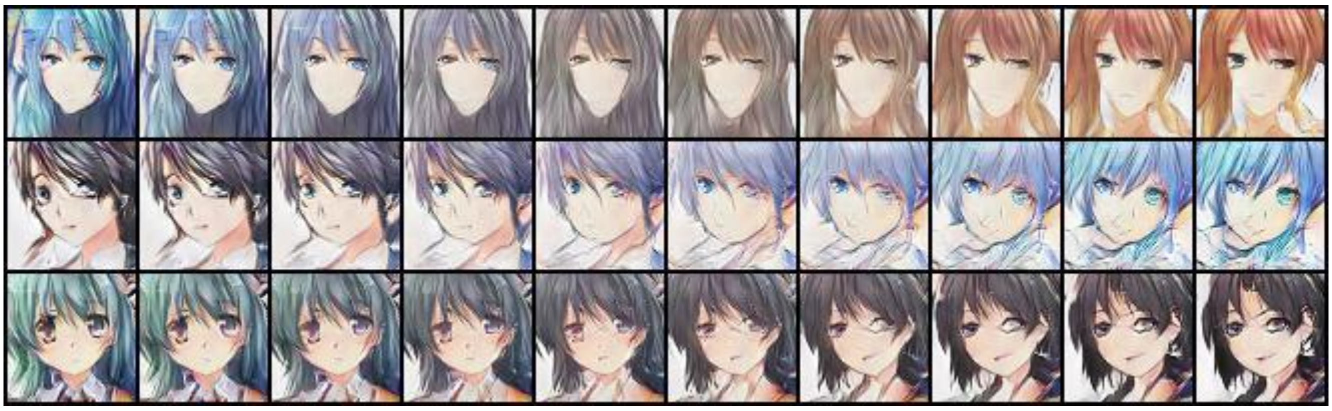
Epoch 100

Epoch 150

Epoch 200

A.2-3 Discuss about A.1-4. Please explain how the z vector influence your images here. (5%)

From the three sets of vectors in A.1-4, we observe two randomly sampled vectors, $z1$ and $z2$, and the vectors resulting from the interpolation between $z1$ and $z2$. These vectors are input into the generator, resulting in images displayed from left to right as follows.



For each set, the two randomly generated vectors, $z1$ and $z2$, produce two images, $i1$ and $i2$, respectively, with distinct features. However, during the interpolation process from $z1$ to $z2$, the features in the generated images gradually shift from those of $i1$ to those of $i2$. For example, the image in row 3, from left to right, we can observe that the hair's color is gradually changing from green and to black, and the eye is also changing from big eyes to smaller eyes.

From the result, we can observe that the content of z, as the input to the generator, influence the features of the resulting images. Moreover, throughout the interpolation within A.1-4, the features of the images change in accordance with the interpolated values of z. The higher the proportion of one vector, the closer the resulting image's features with that particular vector, showcasing a continuous transition from $i1$ to $i2$.

Hence, we can see that the z vector will influence the features of the generated images by the generator.

Reference:

The code is reference from the code given by TA in the HW's slide