Hw4 B03901111 汪家銘

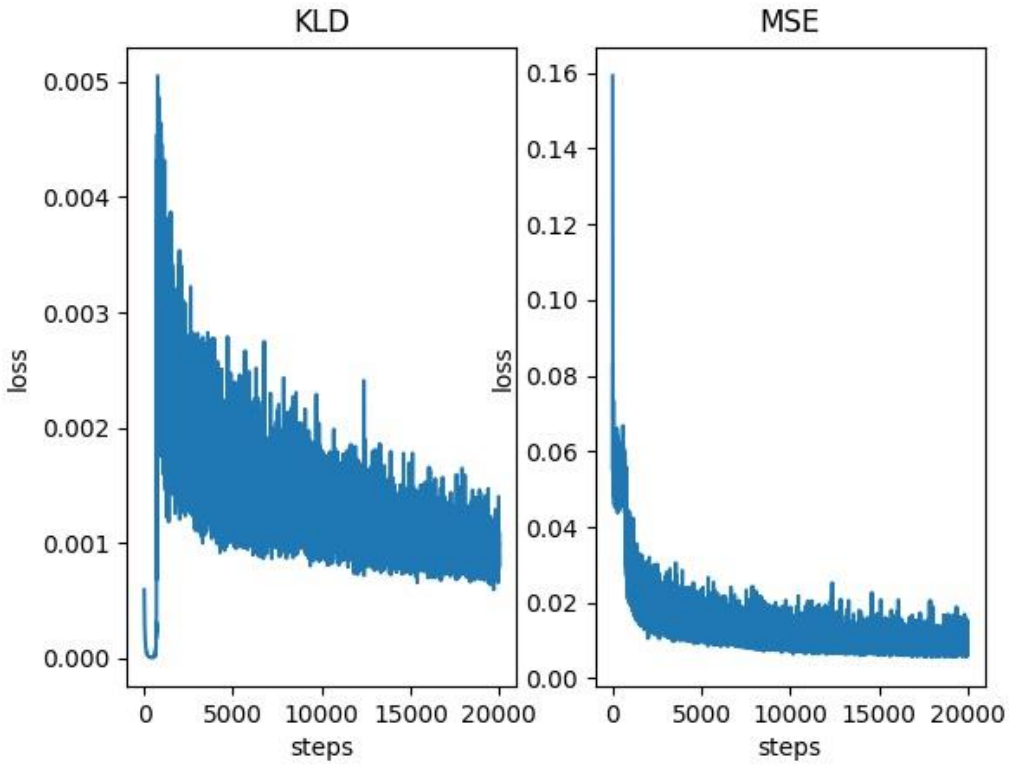
VAE

1.1

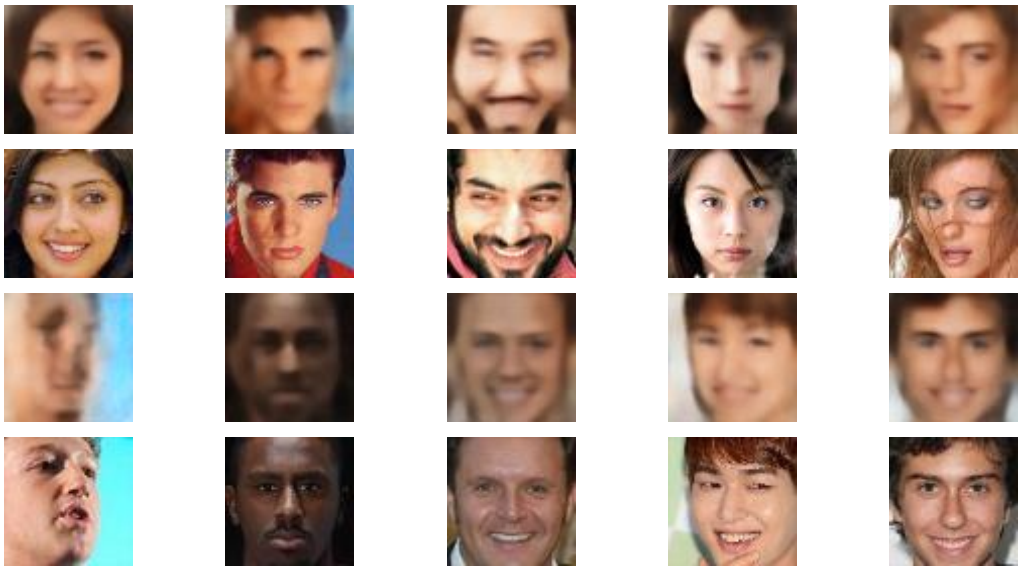
```
VAE(
  (encoder): E(
    (E1C): Sequential(
      (0): Conv2d(3, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
    (E11C): Sequential(
      (0): Conv2d(3, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
    (E2C): Sequential(
      (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
    )
    (E3C): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
    )
    (E31C): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
    (E4C): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
    )
    (E5L): Linear(in_features=2048, out_features=512, bias=True)
  )
```

```
  (decoder): D(
    (D1H): Linear(in_features=512, out_features=512, bias=True)
    (D2H): Linear(in_features=512, out_features=32768, bias=True)
    (D3U): Sequential(
      (0): UpsamplingNearest2d(scale_factor=2, mode=nearest)
      (1): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (2): LeakyReLU(negative_slope=0.2)
    )
    (D4U): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): UpsamplingNearest2d(scale_factor=2, mode=nearest)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (D41U): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
    (D5U): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): UpsamplingNearest2d(scale_factor=2, mode=nearest)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (D51U): Sequential(
      (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
    (D6C): Sequential(
      (0): Conv2d(32, 3, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
  )
  (mean): Linear(in_features=512, out_features=512, bias=True)
  (log_var): Linear(in_features=512, out_features=512, bias=True)
)
```

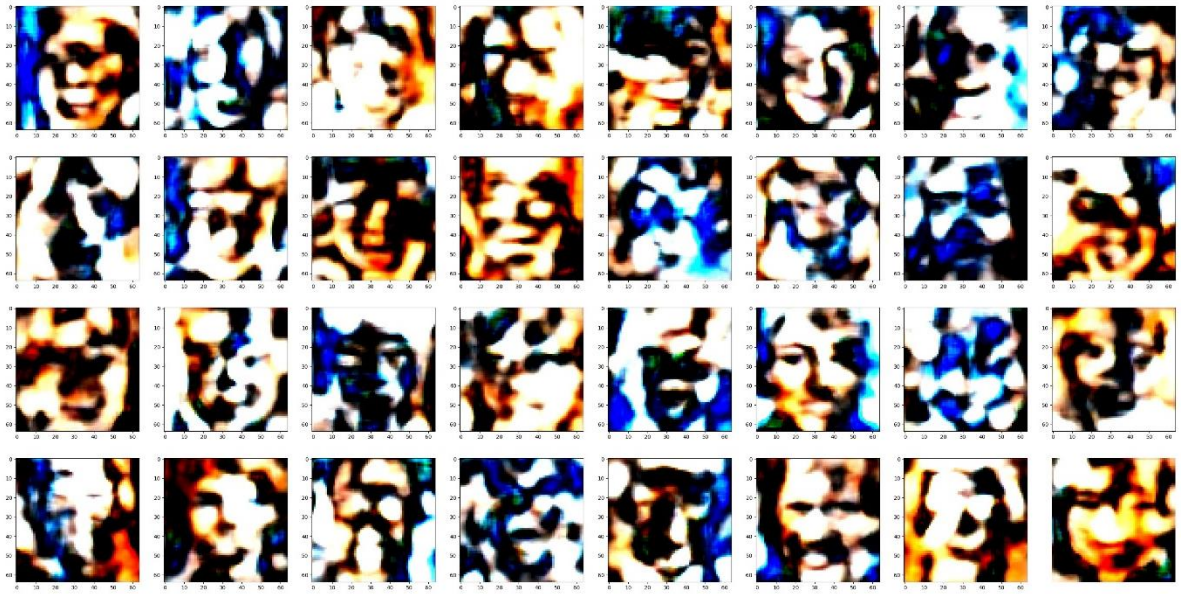
1.2 learning curve



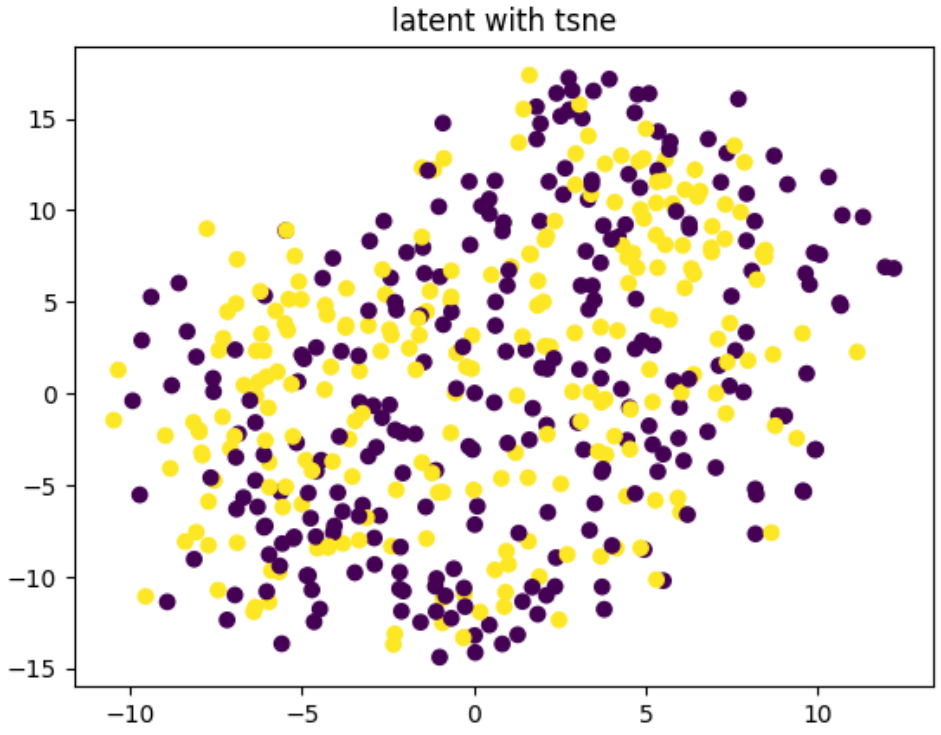
1.3 testing Images



1.4 random Image



1.5 t-SNE(by SMILE)



1.6

1.層次不是越多越好

2.batchnorm 很有效

3.檔案管理要做好，不然容易毀掉珍貴的 model

GAN

2.1 我套用的是 DCGAN 的架構，所以是設 stride=2，每層 size 加倍，從(32,8,8)開始，Loss 是用 BCE，有試過 WGAN 的 mean，不過效果不彰就放棄了。

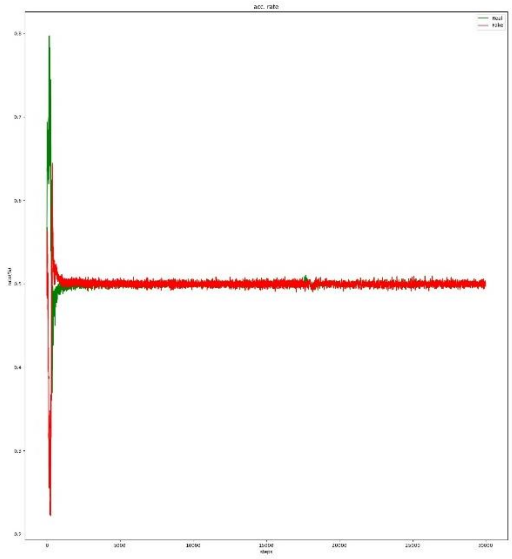
```
Discreminator(
  (encoder): E(
    (E1C): Sequential(
      (0): Conv2d(3, 4, kernel_size=(4, 4), stride=(2, 2), padding=
(1, 1))
      (1): LeakyReLU(negative_slope=0.2)
    )
    (E2C): Sequential(
      (0): Conv2d(4, 8, kernel_size=(4, 4), stride=(2, 2), padding=
(1, 1))
      (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, tra
ck_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (E3C): Sequential(
      (0): Conv2d(8, 16, kernel_size=(4, 4), stride=(2, 2), padding
=(1, 1))
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, tr
ack_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (E4C): Sequential(
      (0): Conv2d(16, 32, kernel_size=(4, 4), stride=(2, 2), paddin
g=(1, 1))
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, tr
ack_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
  )
  (Dis3L): Sequential(
    (0): Conv2d(32, 1, kernel_size=(4, 4), stride=(1, 1), bias=Fals
e)
    (1): Sigmoid()
  )
)
```

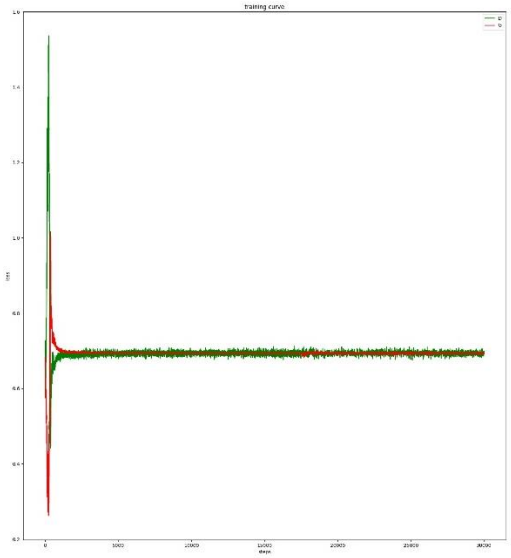
Discriminator

```
D(
  (D1U): Sequential(
    (0): ConvTranspose2d(50, 32, kernel_size=(4, 4), stride=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2)
  )
  (D2U): Sequential(
    (0): ConvTranspose2d(32, 16, kernel_size=(4, 4), stride=(2, 2),
 padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2)
  )
  (D3U): Sequential(
    (0): ConvTranspose2d(16, 8, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1))
    (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track
_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2)
  )
  (D4U): Sequential(
    (0): ConvTranspose2d(8, 4, kernel_size=(4, 4), stride=(2, 2), p
adding=(1, 1))
    (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track
_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2)
  )
  (D5U): Sequential(
    (0): ConvTranspose2d(4, 3, kernel_size=(4, 4), stride=(2, 2), p
adding=(1, 1))
    (1): Tanh()
  )
)
```

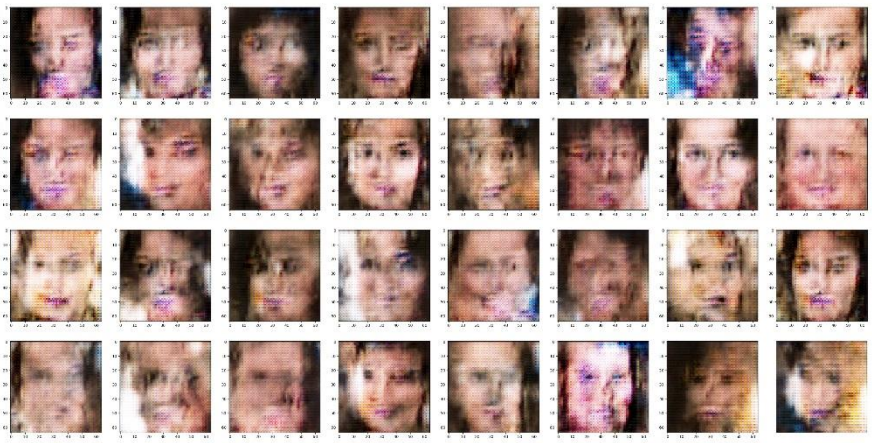
Generator

2.2





2.3 random Image



2.4

DCGAN 很實用

WGAN 反而容易壞掉

如果一開始結果不好，後面加時間真的很難挽救(50 epoch 也沒辦法增加效果)

2.5（在 train 不好的情況下）

VAE：輪廓清楚，但是色彩不好

GAN：輪廓模糊，但是色彩比較逼真(我猜是因為我用 BCE train 吧)

ACGAN

3.1

和 GAN 部分用的是相同的 DCGAN 架構，不過沒時間寫完，所以就講大概，就是把前面加上 13 維，D 加上 int 形態的 class 輸出而已。

3.2