學號：b06901153系級： 電機三 姓名：林瑩昇

Collaborator：B05901184 賴言禮

1. (2.5%) 訓練一個 model。

   (1%) 請描述你使用的 model（可以是 baseline model）。包含 generator 和 discriminator 的 model architecture、loss function、使用的dataset、optimizer 參數、以及訓練 step 數（或是 epoch 數）。

   (1.5%) 請畫出至少 16 張 model 生成的圖片。

   (1) model：

   A. generator：

   先輸入linear，在輸入多層的convTranspose，最後輸入tanh得到64*64的圖片(下左)

   B. discriminator

   與generator的結構相似，但是convTranspose改成用conv，最後通過的是sigmoid(下右)

```
def __init__(self, in_dim, dim=64):
    super(Generator, self).__init__()
    def dconv_bn_relu(in_dim, out_dim):
        return nn.Sequential(
            nn.ConvTranspose2d(in_dim, out_dim, 5, 2,
                               padding=2, output_padding=1, bias=Fa
            nn.BatchNorm2d(out_dim),
            nn.ReLU())
    self.l1 = nn.Sequential(
        nn.Linear(in_dim, dim * 8 * 4 * 4, bias=False),
        nn.BatchNorm1d(dim * 8 * 4 * 4),
        nn.ReLU())
    self.l2_5 = nn.Sequential(
        dconv_bn_relu(dim * 8, dim * 4),
        dconv_bn_relu(dim * 4, dim * 2),
        dconv_bn_relu(dim * 2, dim),
        nn.ConvTranspose2d(dim, 3, 5, 2, padding=2, output_padding=
        nn.Tanh())
    self.apply(weights_init)
def forward(self, x):
    y = self.l1(x)
    y = y.view(y.size(0), -1, 4, 4)
    y = self.l2_5(y)
    return y
```

```
def __init__(self, in_dim, dim=64):
    super(Discriminator, self).__init__()
    def conv_bn_lrelu(in_dim, out_dim):
        return nn.Sequential(
            nn.Conv2d(in_dim, out_dim, 5, 2, 2),
            nn.BatchNorm2d(out_dim),
            nn.LeakyReLU(0.2))
    self.ls = nn.Sequential(
        nn.Conv2d(in_dim, dim, 5, 2, 2), nn.LeakyReLU(0
        conv_bn_lrelu(dim, dim * 2),
        conv_bn_lrelu(dim * 2, dim * 4),
        conv_bn_lrelu(dim * 4, dim * 8),
        nn.Conv2d(dim * 8, 1, 4),
        nn.Sigmoid())
    self.apply(weights_init)
def forward(self, x):
    y = self.ls(x)
    y = y.view(-1)
    return y
```

   C. loss function：BCELoss

   D. optimizer：都是用Adam

   E. dataset：助教提供的dataset

   F. epoch = 10

(2) output image：

2. (3.5%) 請選擇下列其中一種 model： WGAN, WGAN-GP, LSGAN, SNGAN（不要和 1. 使用的model 一樣，至少 architecture 或是 loss function 要不同）

(1%) 同 1.a ，請描述你選擇的 model，包含 generator 和 discriminator 的 model architecture、loss function、使用的dataset、optimizer 參數、及訓練 step 數（或是 epoch 數）。

(1.5%) 和 1.b 一樣，就你選擇的 model，畫出至少 16 張 model 生成的圖片。

(1%) 請簡單探討你在 1. 使用的 model 和 2. 使用的 model，他們分別有何性質，描述你觀察到的異同。

(1) model

A. generator：與baseline相同

B. discriminator：在module前加入spectral_norm

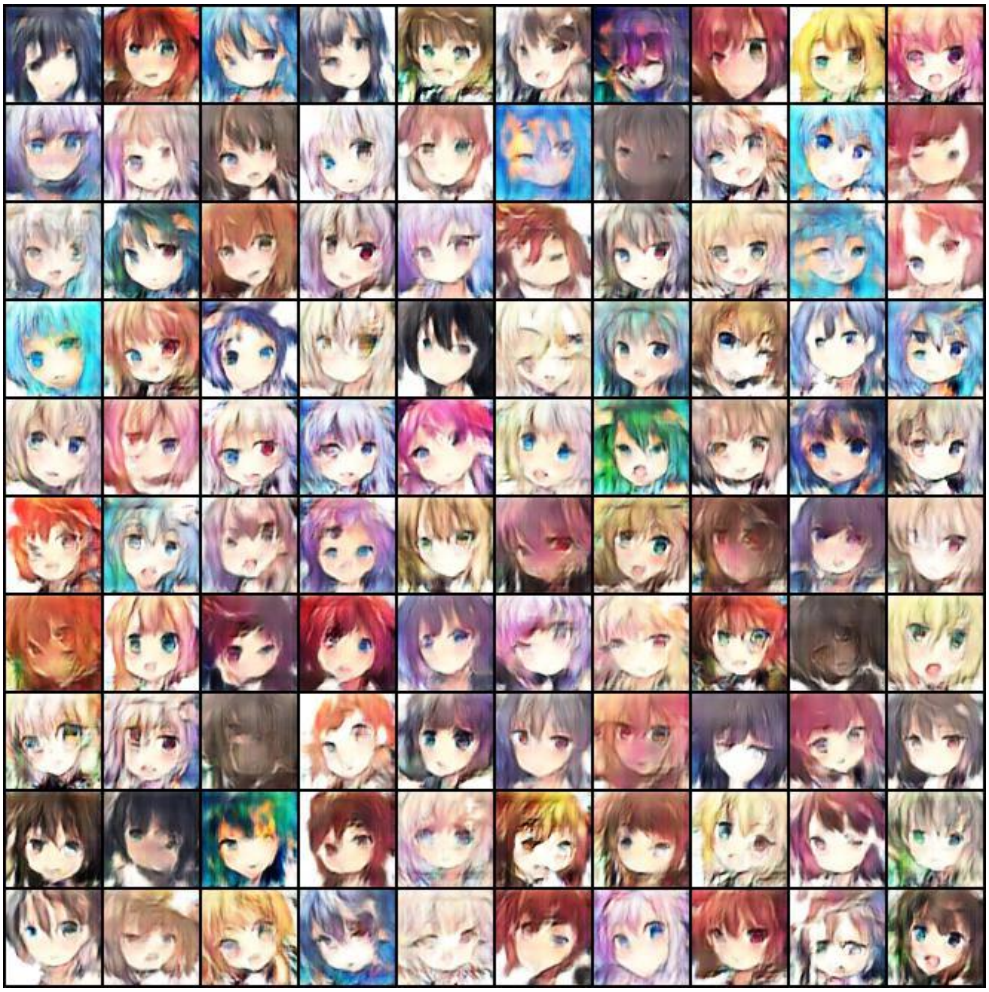
```
def __init__(self, in_dim, dim=64):
    super(Discriminator, self).__init__()
    def conv_bn_lrelu(in_dim, out_dim):
        return nn.Sequential(
            spectral_norm(nn.Conv2d(in_dim, out_dim, 5, 2, 2)),
            nn.LeakyReLU(0.2))
    self.ls = nn.Sequential(
        spectral_norm(nn.Conv2d(in_dim, dim, 5, 2, 2)), nn.LeakyReLU(0.2),
        conv_bn_lrelu(dim, dim * 2),
        conv_bn_lrelu(dim * 2, dim * 4),
        conv_bn_lrelu(dim * 4, dim * 8),
        spectral_norm(nn.Conv2d(dim * 8, 1, 4)),
        nn.Sigmoid()
    )
    self.apply(weights_init)

def forward(self, x):
    y = self.ls(x)
    y = y.view(-1)
    return y
```

C. optimizer：也是用adam，但為了使G&D比較平衡，所以lr_G = lr_D*3

D. epoch：10

(2) output image



(3) discussion：

我把baseline model訓練100個epoch時發現，loss_D與loss_G相差很多，代表discriminator比generator強太多了。而SNGAN convolution後的結果會過一個normalization，感覺與WGAN的weight clipping的效果很像，避免discriminator與generator差太多。下圖是baseline與SNGAN Loss比較。

```
Epoch [100/100] 1097/1115 Loss_D: 0.0021 Loss_G: 6.7558
Epoch [100/100] 1098/1115 Loss_D: 0.0125 Loss_G: 4.8903
Epoch [100/100] 1099/1115 Loss_D: 0.0155 Loss_G: 6.0646
Epoch [100/100] 1100/1115 Loss_D: 0.0049 Loss_G: 6.1737
```

```
Epoch [100/100] 985/1115 Loss_D: 0.6098 Loss_G: 0.8118
Epoch [100/100] 986/1115 Loss_D: 0.6060 Loss_G: 0.8482
Epoch [100/100] 987/1115 Loss_D: 0.6051 Loss_G: 0.9032
Epoch [100/100] 988/1115 Loss_D: 0.6075 Loss_G: 0.8008
```

3. (4%) 請訓練一個會導致 mode collapse 的 model。

(1%) 同 1.a ，請描述你選擇的 model，包含 generator 和 discriminator 的 model architecture、loss function、使用的dataset、optimizer 參數、及訓練 step 數（或是 epoch 數）。

(1.5%) 請畫出至少16張 model 生成且具有mode collapse現象的圖片。

(1.5%) 在不改變 optimizer 和訓練 step 數的情況下，請嘗試使用一些方法來減緩 mode collapse。說明你嘗試了哪些方法，請至少舉出一種成功改善的方法，若有其它失敗的方法也可以記錄下來。

(1) model：我用的是baseline model的結構，然後train 100 epoch

(2) mode collapse：

(3) optimize

我最一開始是想用wgan，因為看起來比較容易實作，但不知道為甚麼，改了loss function後，連圖片都無法生成。所以後來我改用SNGAN，唯一與2.不同的是train的epoch數，這次為了比較需求，epoch數是100，可以發現SNGAN與一班的GAN相比，可以train的epoch數更多，也就可以達到比較好的結果。