學號：b07901112 系級：電機二　　姓名：劉聿珉

1. (2.5%) 訓練一個 model。

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

第一題所使用的 model 訓練方式就是用助教的 sample code 為基底下去做的。

模型架構：

(in_dim = 100)

```
class Generator(nn.Module):
    """
    input (N, in_dim)
    output (N, 3, 64, 64)
    """
    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=False),
                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=1),
            nn.Tanh())
        self.apply(weights_init)
```

(in_dim = 3)

```
class Discriminator(nn.Module):
    """
    input (N, 3, 64, 64)
    output (N, )
    """
    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(0.2),
            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)
```

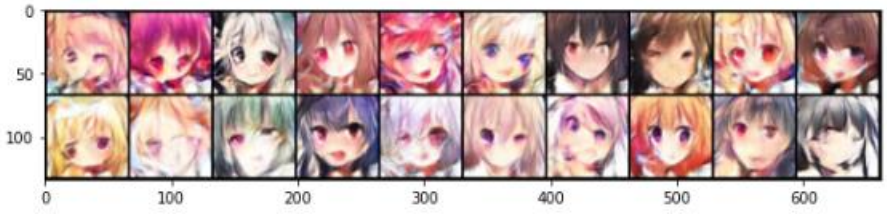
Optimizer：Generator 和 Discriminator 都使用 Adam，learning rate 為 1e-4，betas = (0.5,0.999)

Loss：binary cross-entropy loss

Dataset：助教提供的圖片

總共跑了10個epoch

b. (1.5%) 請畫出至少 16 張 model 生成的圖片。



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）
   a. (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）。
   這題我所使用的是 WGAN model
   Model 架構：
   (in_dim = 100)

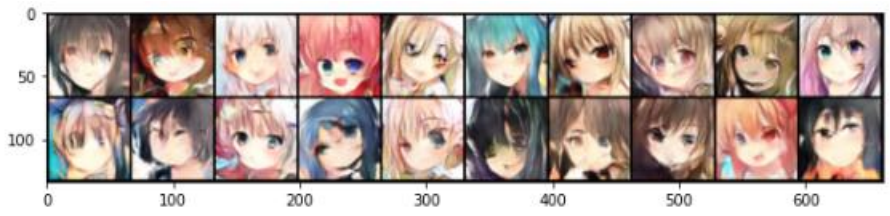
```
class WGenerator(nn.Module):
    """
    input (N, in_dim)
    output (N, 3, 64, 64)
    """
    def __init__(self, in_dim, dim=64):
        super(WGenerator, self).__init__()

        self.cnn = nn.Sequential(
            nn.ConvTranspose2d(in_dim, dim*8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(dim*8),
            nn.ReLU(True),

            nn.ConvTranspose2d(dim*8, dim*4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(dim*4),
            nn.ReLU(True),

            nn.ConvTranspose2d(dim*4, dim*2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(dim*2),
            nn.ReLU(True),

            nn.ConvTranspose2d(dim*2, dim, 4, 2, 1, bias=False),
            nn.BatchNorm2d(dim),
            nn.ReLU(True),

            nn.ConvTranspose2d(dim, 3, 4, 2, 1, bias=False),
            nn.Tanh(),
        )
        self.apply(weights_init)
```

(in_dim = 3)

```
class WDiscriminator(nn.Module):
    """
    input (N, 3, 64, 64)
    output (N, )
    """
    def __init__(self, in_dim, dim=64):
        super(WDiscriminator, self).__init__()

        self.cnn = nn.Sequential(
            nn.Conv2d(in_dim, dim, 4, 2, 1, bias=False), # (N, dim, 32, 32)
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(dim, dim*2, 4, 2, 1, bias=False), # (N, dim*2, 16, 16)
            nn.BatchNorm2d(dim*2),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(dim*2, dim*4, 4, 2, 1, bias=False), # (N, dim*4, 8, 8)
            nn.BatchNorm2d(dim*4),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(dim*4, dim*8, 4, 2, 1, bias=False), # (N, dim*8, 4, 4)
            nn.BatchNorm2d(dim*8),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(dim*8, 1, 4, 1, 0, bias=False), # (N, 1, 1, 1)
            # Modification 1: remove sigmoid
            # nn.Sigmoid()
        )
        self.apply(weights_init)
```

Optimizer：Generator 和 Discriminator 都改成 RMSprop
learning rate = 1e-4。
Loss：將BCE loss中的log拿掉，也就是將BCE loss改成Wasserstein loss
Dataset：助教提供的圖片
總共跑了10個epoch

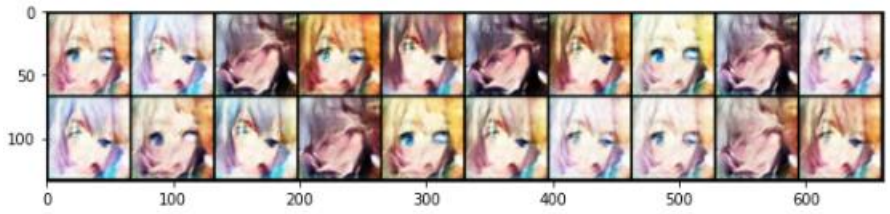
b. (1.5%) 和 1.b 一樣，就你選擇的 model，畫出至少 16 張 model 生成的圖片。



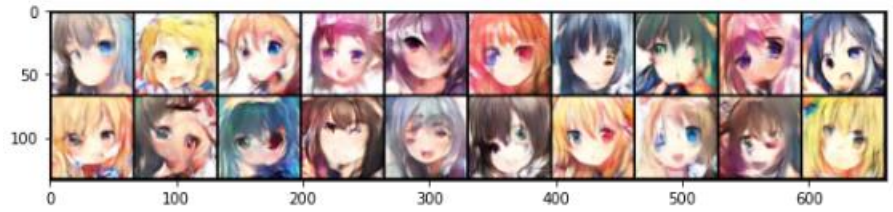
c. (1%) 請簡單探討你在 1. 使用的 model 和 2. 使用的 model，他們分別有何性質，描述你觀察到的異同。
我發現當我用 1.的方法 train model 時，前 1~3 個 epoch 印出來圖都十分模糊且色調十分接近，是到越來越後面才會開始倒吃甘蔗，越來越清楚。但是 2.的 model 則比較不會有這樣的問題。

另外，有一次我不小心將 epoch 調成 20，我發現 1.的 model 會產生 mode collapse 但是 2.的 model 卻幾乎不會。所以我也將此特性應用在第三題中。

3. (4%) 請訓練一個會導致 mode collapse 的 model。
   a. (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）。
   我所使用的 model 就是我 1.的 model，我將 epoch 調成 25，loss、optimizer、模型架構、Dataset 等等都維持相同不變。
   b. (1.5%) 請畫出至少 16 張 model 生成且具有 mode collapse 現象的圖片。



   c. (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。



   我將 model 的架構改成 2.的架構，並且將 loss 也改成 Wasserstein loss，在不改變 epoch 以及 optimizer 的狀況下，生成的圖片如上，發現 mode collapse 的現象有改善許多。