學號：B06902006 系級：資工三 姓名：王俊翔

1. 訓練一個 model。
   a. 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）。

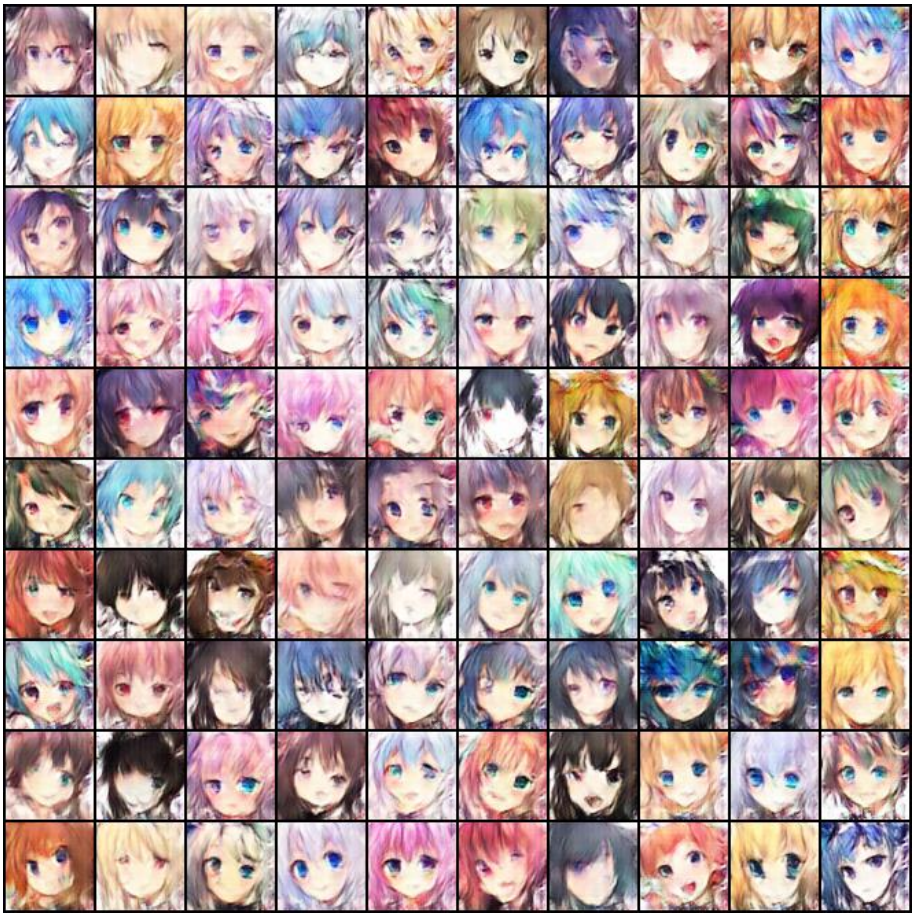
      Ans:

      這裡我的 model 為 baseline model, architecture 如下圖，loss function 都是計算其 binary cross entropy (JS divergence)，optimizer 使用的是 ADAM, learning rate 為 1e-4，總 epoch 次數為 10 次。

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

```
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)
    def forward(self, x):
        y = self.l1(x)
        y = y.view(y.size(0), -1, 4, 4)
        y = self.l2_5(y)
        return y
```

b. 請畫出至少 16 張 model 生成的圖片。



2. 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. 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）。

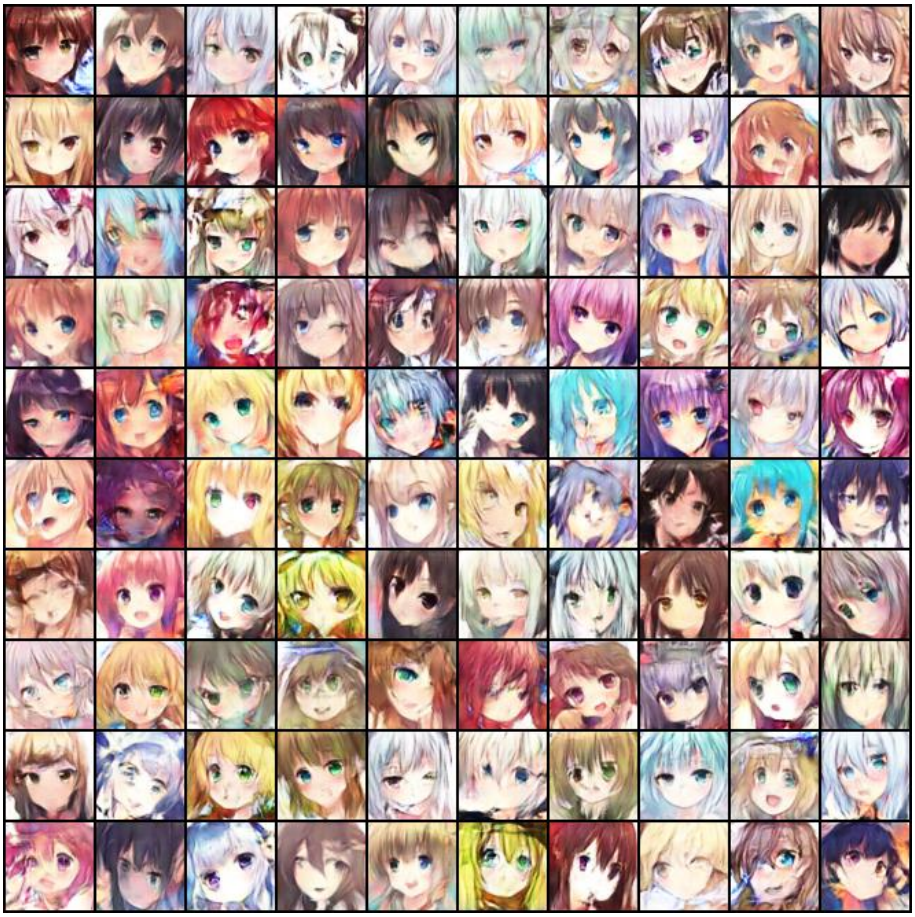
Ans:

這裡我選擇的 model 為 WGAN-GP, architecture 如下圖，loss function 的部分，discriminator 計算其 Wasserstein distance (使 real 的得分越高越好，fake 的越低越好) 並加上 gradient 的 penalty，使其越接近 1 越好，generator 則是使其產生出來的圖片的得分越高越好，optimizer 使用的是 ADAM, learning rate 為 1e-4，總 epoch 次數為 30 次。

```python
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)
    def forward(self, x):
        y = self.l1(x)
        y = y.view(y.size(0), -1, 4, 4)
        y = self.l2_5(y)
        return y
```

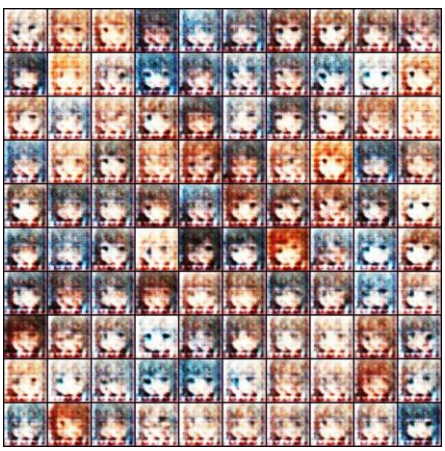
```python
class Discriminator_GP(nn.Module):
    """
    input (N, 3, 64, 64)
    output (N, )
    """
    def __init__(self, in_dim, dim=64):
        super(Discriminator_GP, self).__init__()
        def conv_lrelu(in_dim, out_dim):
            return nn.Sequential(
                nn.Conv2d(in_dim, 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_lrelu(dim, dim * 2),
            conv_lrelu(dim * 2, dim * 4),
            conv_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
```

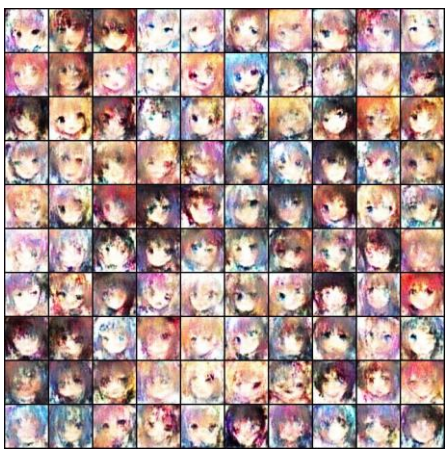
b. 和 1.b 一樣，就你選擇的 model，畫出至少 16 張 model 生成的圖片。



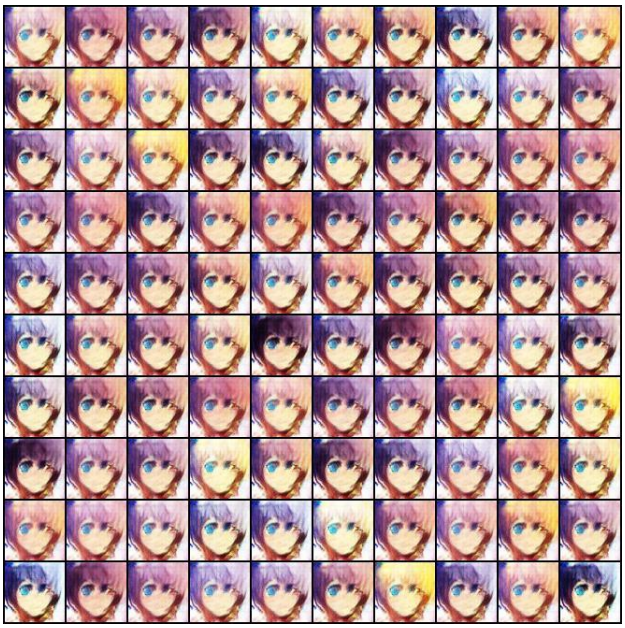
c. 請簡單探討你在 1. 使用的 model 和 2. 使用的 model，他們分別有何性質，描述你觀察到的異同。

1 的 model 的 loss 使用的是 JS divergence，所以當 discriminator 可以完全分辨真偽時，loss 會一模一樣，導致訓練時會比較困難，這個也可以很明顯在我 train 兩者時表現出來，在第一個 epoch 時，WGAN-GP 就已經可以 train 出一些還可以的圖，但是 1 的 model 就不行 (如下圖所示, 左為 1, 右為 2)，且神奇的是，1 的 model 到最後發生了 mode collapse，2 的竟然沒有，可能的原因是 2 為讓 fake 的 distribution 與 real 的越像越好，不過 1 所 generate 出的 distribution 有可能僅僅滿足某部分 data 的性質，也就是缺乏大局觀，產生出的圖比較保守。





3. 請訓練一個會導致 mode collapse 的 model。
   a. 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）。
   使用的皆為 1 的構造, loss function, optimizer，只有 epoch 調為 28 次。
   b. 請畫出至少 16 張 model 生成且具有 mode collapse 現象的圖片。



   c. 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。
   就如同第二題所說，我使用 WGAN-GP 後，成功的改善了 mode collapse 的情況，在同樣 28 次 epoch 的情況下，產生了如下的結果。