HW 3

ID & Name : R06946009 林庭宇、R06946006李筑真、R06946015黃永翰

1. Model Description

Describe the models you use to, including the model architecture, objective function for G and D.

- Image Generation (2%)

Parameters:

1. Data preprocessing : Normalize像素值，縮放平移像素值到[-1, 1]區間。
2. Objective function: BCE loss
   a. For dsicriminator : $\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$

$$Minimize\ \tilde{V} = \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$$

   b. For generator:

$$Minimize\ \tilde{V} = -\frac{1}{m}\sum_{i=1}^{m} log\left(D\left(G(z^i)\right)\right)$$

3. Model architecture :
   a. Generator : 輸出層的activation function為Tanh。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 32768)             3309568
_________________________________________________________________
reshape_1 (Reshape)          (None, 16, 16, 128)       0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 32, 32, 128)       262272
_________________________________________________________________
batch_normalization_4 (Batch (None, 32, 32, 128)       512
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 64, 64, 64)        131136
_________________________________________________________________
batch_normalization_5 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 64, 64, 3)         3075
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 3)         0
=================================================================
Total params: 3,706,819
Trainable params: 3,706,435
Non-trainable params: 384
_________________________________________________________________
```

b. Discriminator :輸出層的activation function為Sigmoid。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        1568
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        32832
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 17, 17, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 17, 17, 64)        256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 17, 17, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 9, 128)         131200
_________________________________________________________________
batch_normalization_2 (Batch (None, 9, 9, 128)         512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 9, 9, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 9, 9, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 9, 9, 256)         524544
_________________________________________________________________
batch_normalization_3 (Batch (None, 9, 9, 256)         1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 9, 9, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 9, 9, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 20737
=================================================================
Total params: 712,673
Trainable params: 711,777
Non-trainable params: 896
```

- Text-to-image Generation (2%)

Parameters:

1. Data preprocessing : Normalize像素值，縮放平移像素值到[-1, 1]區間。

2. Objective function: BCE loss

a. For dsicriminator : $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$

$$Minimize\ \tilde{V} = \frac{1}{m}\sum_{i=1}^{m} logD(x^i) + \frac{1}{m}\sum_{i=1}^{m} log\left(1 - D(\tilde{x}^i)\right)$$

b. For generator:

$$Minimize\ \tilde{V} = -\frac{1}{m}\sum_{i=1}^{m} log\left(D\left(G(z^i)\right)\right)$$

3. Model architecture :

a. Generator : 輸出層的activation function為Tanh。

```
generator(
  (deconv1): ConvTranspose2d(123, 512, kernel_size=(4, 4), stride=(1, 1))
  (deconv1_bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
  (leaky_relu,negative_slope=0.2)
  (deconv2): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (deconv2_bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (leaky_relu,negative_slope=0.2)
  (deconv3): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (deconv3_bn): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (leaky_relu,negative_slope=0.2)
  (deconv4): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (deconv4_bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
  (leaky_relu,negative_slope=0.2)
  (deconv5): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
)
```

b. Discriminator :輸出層的activation function為Sigmoid。

```
discriminator(
  (conv1): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (leaky_relu,negative_slope=0.2)
  (conv2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2_bn): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
  (leaky_relu,negative_slope=0.2)
  (conv3): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv3_bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
  (leaky_relu,negative_slope=0.2)
  (conv4): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv4_bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
```

(leaky_relu,negative_slope=0.2)
(conv5): Conv2d(535, 512, kernel_size=(4, 4), stride=(1, 1))
(conv5_bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
(leaky_relu,negative_slope=0.2)
(linear): Linear(in_features=512, out_features=1, bias=True)
)

2. Experiment settings and observation

Show generated images

- Image Generation (1%)

Experiment settings ：

1. training iterations : 80000
2. batch size : 64
3. optimizer : Adam (lr = 0.0002,beta_1 = 0.5)

Results :

上圖為經過80000個iterations後的生成結果，baseline偵測為25張人臉，基本上動漫妹子裡頭的重要特徵兩個大眼睛都有學習到，不過嘴巴就比較少出現，可能因為動畫妹子嘴吧確實也挺小的，不太好學習到。另外加了在將input normalization後，部分pixel崩壞問題似乎好了許多。

- Text-to-image Generation (1%)

Experiment settings ：

1. training epochs: 30
2. batch size : 64
3. optimizer : Adam (lr = 0.0002,beta_1 = 0.5)
4. Noise維度：100
5. Label維度：23 (hair的12維與eyes的11維concat)

Results :

上圖為經過30個epochs後的生成結果，baseline偵測為25張人臉，並且產生的人臉符合testing_tags所要求的特徵，data部分只用original會train不太起來，因此也加入了extra data，模型部分試過一般WGAN加線性神經網路的Generator與Discriminator，產生的圖會過於模糊，最後改用DCGAN才產生出上圖的結果。

3. Compare your model with WGAN, WGAN-GP, LSGAN (choose 1) (**Image Generation Only**)

- Model Description of the choosed model (1%)

我們選擇LSGAN (將DCGAN的Discriminator中的最後一層的sigmoid去掉，並且將loss function從BCE改成MSE)

batch size:64

learning rate: 0.0002 (both of generator and discriminator use Adam optimizer)

結構如圖所示：

```
Generator(
  (l1): Sequential(
    (0): Linear(in_features=100, out_features=8192, bias=True)
  )
  (conv_blocks): Sequential(
    (0): Upsample(scale_factor=2, mode=nearest)
    (1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (2): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True)
    (3): LeakyReLU(0.2, inplace)
    (4): Upsample(scale_factor=2, mode=nearest)
    (5): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True)
    (7): LeakyReLU(0.2, inplace)
    (8): Conv2d(64, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): Tanh()
  )
)
```

```
Discriminator(
  (model): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(0.2, inplace)
    (2): Dropout2d(p=0.25)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (4): LeakyReLU(0.2, inplace)
    (5): Dropout2d(p=0.25)
    (6): BatchNorm2d(32, eps=0.8, momentum=0.1, affine=True)
    (7): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (8): LeakyReLU(0.2, inplace)
    (9): Dropout2d(p=0.25)
    (10): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True)
    (11): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (12): LeakyReLU(0.2, inplace)
    (13): Dropout2d(p=0.25)
    (14): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True)
  )
  (adv_layer): Linear(in_features=512, out_features=1, bias=True)
)
```

- Result of the model (1%)

- Comparison Analysis (1%)

使用 lsgan training時較為穩定，原本的gan在training的過程中會有循環式的崩壞，而我們使用lsgan時並沒有發生這樣的現象，此外lsgan也較快出現品質好的人形，但是，由最終的成果而言，效果無明顯差異

4. Training tips for improvement (**Image generation Only**) (6%)

Please implement these tips on image generation . Only the following tips are accepted : 1, 2, 3, 4, 5, 6, 9, 13, 14, 17
Total : 6%, 2% for each

- Which tip & implement details (1%)
- Result (image or loss...etc.) and Analysis (1%)

Tip 1 :

Which tip & Implement details：

Which tip : 1. Normalize the inputs

Implement details：將原始pixel (0~255) 除以127.5，scale到[0, 2]。再減1，shift到[-1, 1]。

```
#--------for tip1-----------------
file_path = './faces'
dirs = os.listdir(file_path)

data_list = []
for i in dirs:
        im = Image.open(os.path.join(file_path, i))
        resize_Im_array = np.array(im.resize((64,64),resample= Image.ANTIALIAS))
        data_list.append( (resize_Im_array/127.5)- 1 )
        print(i, ' fin.')

data = np.array(data_list)
np.save('data_tip1.npy', data)
print(data.shape)
print(data[0])
print(np.max(data))
print(np.min(data))
```

Result and Analysis：

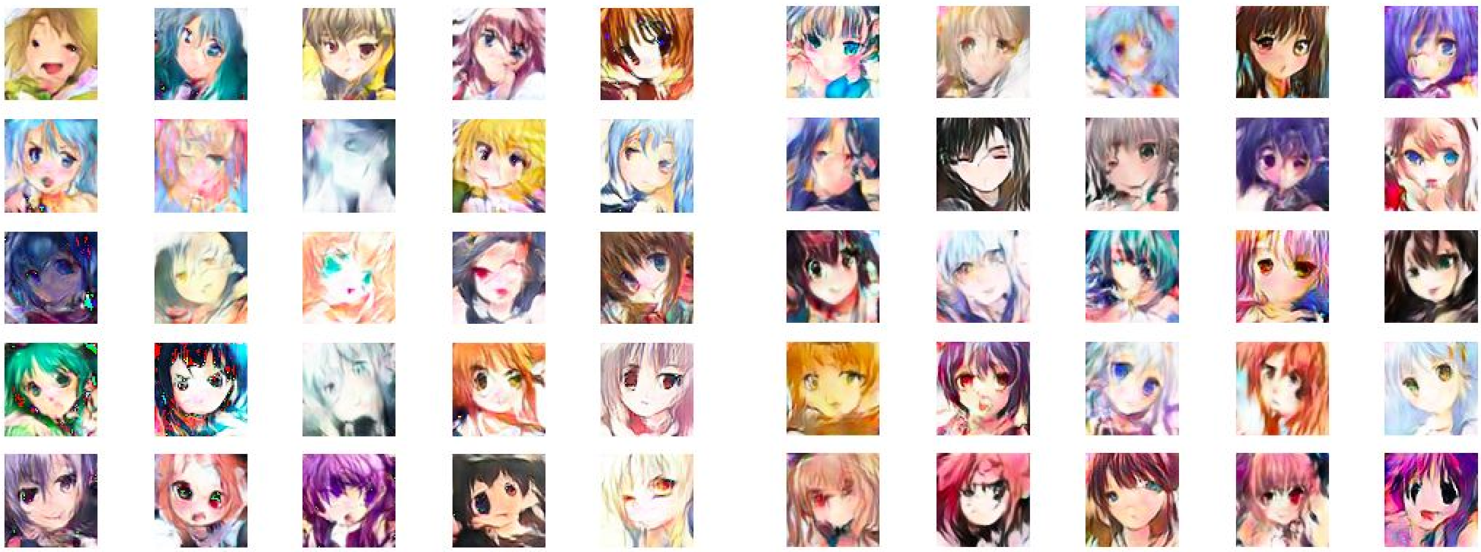


Fig. 1 左圖為沒有Tip 1的模型, 右圖為使用Tip 1。

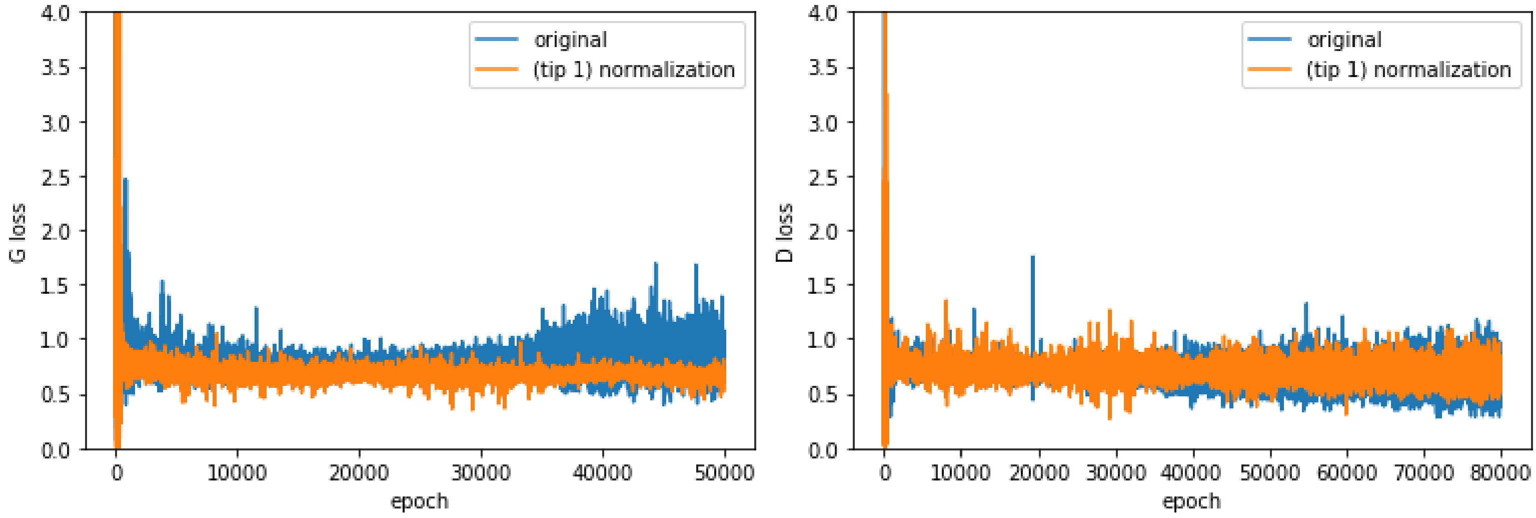


Fig. 2 loss趨勢圖。左爲生成器的，右為判別器的。

由以上結果所示，從loss的角度來講，用了Tip後收斂的比較好，35000 epoch後，沒有用Tip的loss震盪情形比較嚴重。從圖片品質來看，沒有用Tip的時候，有些pixels看起來有點壞掉，看起來變成明顯的雜訊。以baseline的偵測準確度來看，沒有加的為22/25，有加的為24/25。看起來有加會有比較好的表現。由於輸出層activation function, Tanh的特性（線性區集中在-2至2區間），使用normalize到[-1,1]的input可以避免掉使用 sigmoid 和 tanh 函數容易發生梯度消失問題（其飽和區求導數趨近0），使訓練較為穩定。

Tip 2 :

Which tip & Implement details:

Which tip : 5. Avoid Sparse Gradients: ReLU, MaxPool

Implement details：

Model 1: Without Tip 2

Generator : 使用Conv2D

```python
def build_generator(self):

    model = Sequential()

    model.add(Dense(128 * 16 * 16, activation="relu", input_dim=self.latent_dim))
    model.add(Reshape((16, 16, 128)))
    model.add(UpSampling2D())
    model.add(Conv2D(128, kernel_size=4, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Activation("relu"))
    model.add(UpSampling2D())
    model.add(Conv2D(64, kernel_size=4, padding="same"))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Activation("relu"))
    model.add(Conv2D(self.channels, kernel_size=4, padding="same"))
    model.add(Activation("tanh"))

    model.summary()

    noise = Input(shape=(self.latent_dim,))
    img = model(noise)

    return Model(noise, img)
```

Discriminator：使用maxpooling

```python
model = Sequential()

model.add(Conv2D(32, kernel_size=4, strides=2, input_shape=self.img_shape, padding="same"))
model.add(Activation("relu"))
model.add(Dropout(0.25))
model.add(Conv2D(64, kernel_size=4, strides=2, padding="same"))
model.add(BatchNormalization(momentum=0.8))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, kernel_size=4, strides=2, padding="same"))
model.add(BatchNormalization(momentum=0.8))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(256, kernel_size=4, strides=1, padding="same"))
model.add(BatchNormalization(momentum=0.8))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.summary()

img = Input(shape=self.img_shape)
validity = model(img)
```

Model 2: With Tip 2

Generator：如題一的generator，將Conv2D層換成Conv2DTranspose層，且將relu換成leakyrelu。

Discriminator : 如題一的discriminator，將Model 1的Maxpooling層拔掉且將relu換成leakyrelu。

Result and Analysis :

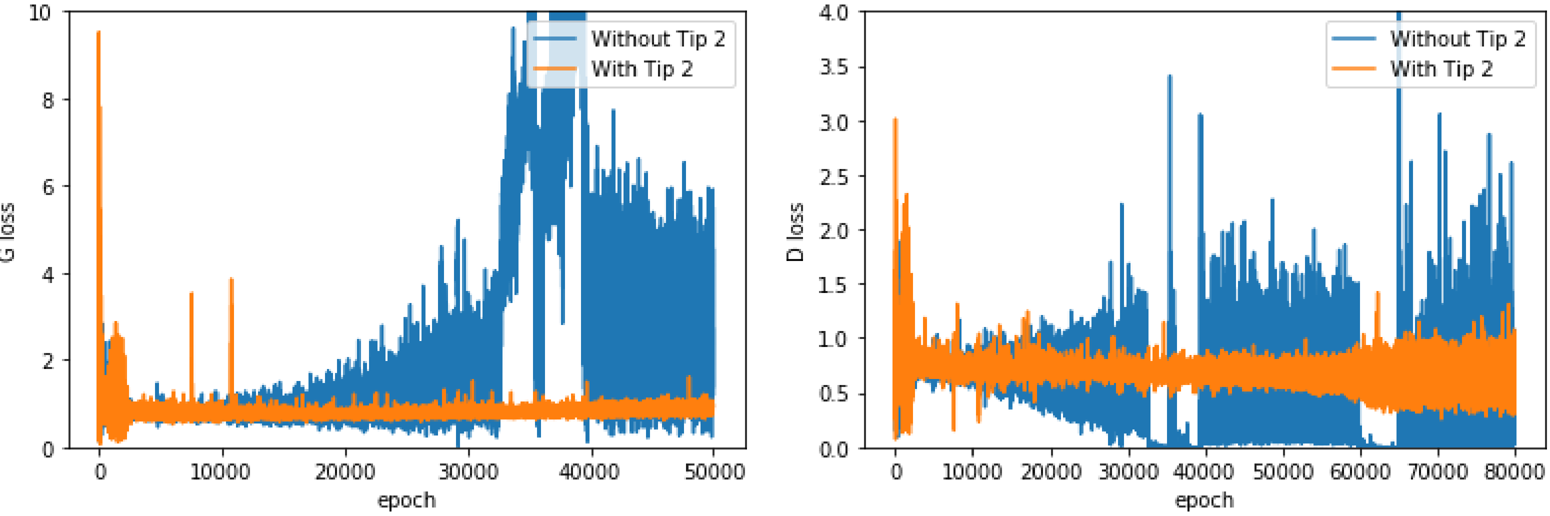


Fig. 3 loss趨勢圖。左爲生成器的，右為判別器的。沒有使用Tip的loss收斂的不穩定。

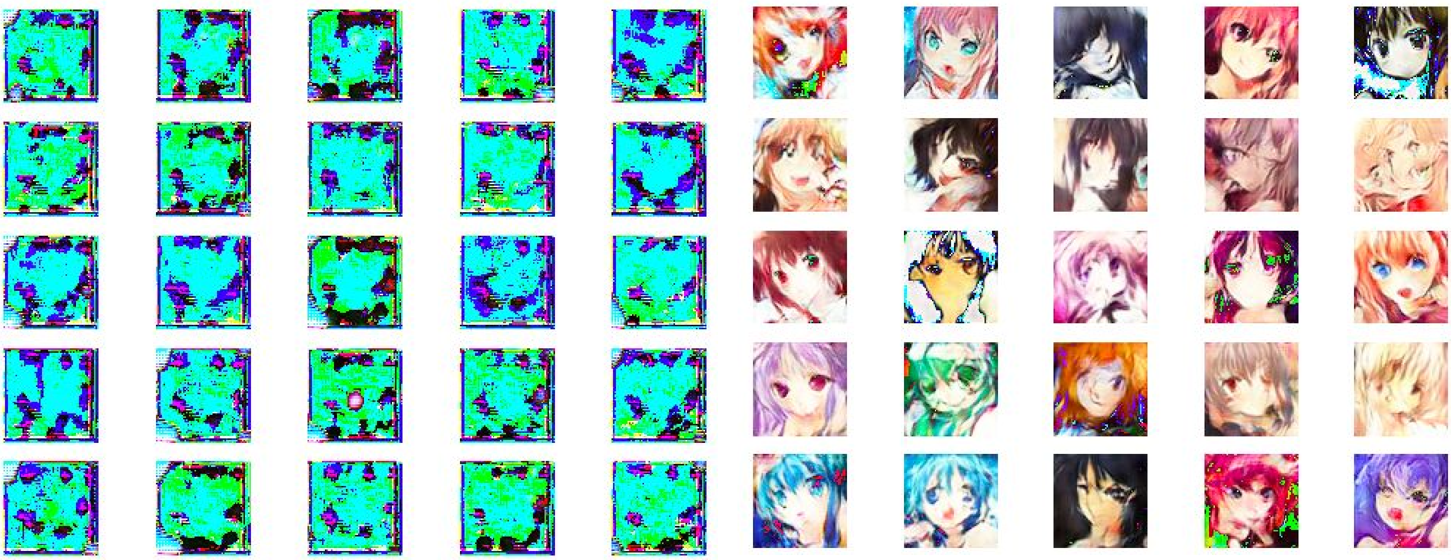


Fig. 4 epoch = 35000時，Model 1 和Model 2的生成圖。

Fig. 5 epoch = 80000時，Model 1 和Model 2的生成圖。

由loss來看，我們知道使用relu和maxpooling可能會造成梯度稀疏的問題，所以其實由整個訓練的過程可以看到，Model 1真的收斂的很不穩定到後面階段甚至loss開始遽升，使得生成器真的壞掉又重新開始收斂（如：Fig. 4左）。但把maxlpooling拔掉、改relu為leakyrelu、使用全卷積之後可以發現，在這樣的條件下，訓練收斂情形較為穩定，生成器也沒有了中途崩壞的情況。故猜想Model 1由於relu那段斜率為0的部分以及maxpooling層中為非最大值的部分（在這些部分沒有梯度），導致梯度稀疏影響整個gan訓練的穩定性。由baseline的偵測器來評分，Model 1能偵測到21張動漫人臉，Model 2可以偵測到23張動漫人臉。另外，也有看到過建議使用全卷積不要使用pooling，以避免傳遞的資訊量減少。

Tip 3 :

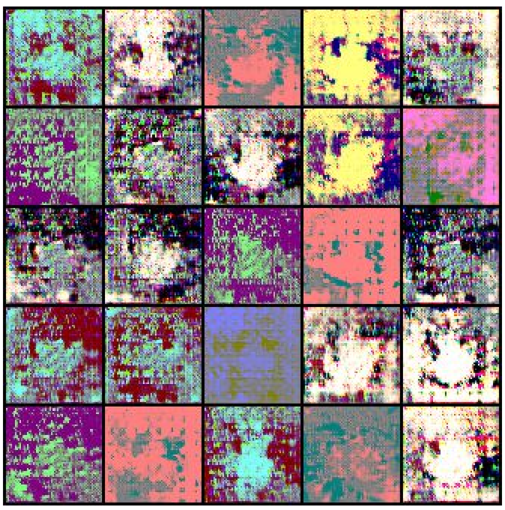
Which tip & Implement details:

tip17: 在training及testing時都在generator開啟dropout

Result and Analysis :

在開啟dropout的情況下，輸出的成果完全崩壞，無法成功train好generator產生出動漫人物大頭圖片

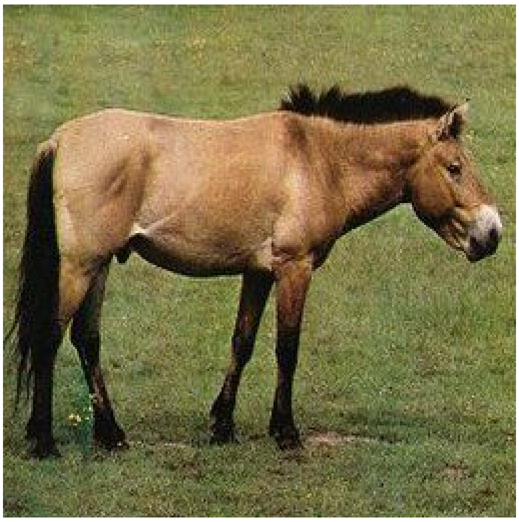
左圖為不使用tip(30000 iterations)，右圖為使用tip(30000 iterations)



5. Style Transfer

Show your result：

Domain A	Domain B
	
	

Origin	Style Transfer
	
	

Analysis：

以上結果是研究投影片所附的CycleGAN Pytorch所得來，下載了馬與斑馬互轉的pretrain model來產生結果，使用了原本附的Testing Data，生成的斑馬圖結果還不錯，並沒有讓黑白條紋影響到背景，另外還找了一張狗圖來測試，結果也還行。

6. Division of Work

ID & Name	Works	%
r06946009 林庭宇	text-to-image generation 、Style transfer	34
r06946006 李筑真	image generation (original gan)、Tip1 & Tip2 implementation	33
r06946015 黃永翰	image generation (lsgan)、Tip3 implementation	33