

## 機器學習及其深度化與結構化 HW3

姓名	學號	工作
吳政軒	R06922118	LSGAN, 3-3, Report
黃敬庭	R06944049	3-2, Report
馬欣婕	R06946010	3-1, Report

### 1. Image Generation

#### ○ Model Description

我們採用的是 DCGAN 的架構，詳細如下，G 和 D 的 loss 均是使用 binary\_cross\_entropy。

```
Generator(  
  (l1): Sequential(  
    (0): Linear(in_features=100, out_features=32768, bias=True)  
  )  
  (conv_blocks): Sequential(  
    (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)  
    (1): Upsample(scale_factor=2, mode=nearest)  
    (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (3): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True)  
    (4): LeakyReLU(0.2, inplace)  
    (5): Upsample(scale_factor=2, mode=nearest)  
    (6): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (7): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True)  
    (8): LeakyReLU(0.2, inplace)  
    (9): Conv2d(64, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (10): Tanh()  
  )  
)
```

```
Discriminator(  
  (model): Sequential(  
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))  
    (1): LeakyReLU(0.2, inplace)  
    (2): Dropout2d(p=0.25)  
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))  
    (4): LeakyReLU(0.2, inplace)  
    (5): Dropout2d(p=0.25)  
    (6): BatchNorm2d(64, eps=0.8, momentum=0.1, affine=True)  
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
```

```

(8): LeakyReLU(0.2, inplace)
(9): Dropout2d(p=0.25)
(10): BatchNorm2d(128, eps=0.8, momentum=0.1, affine=True)
(11): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(12): LeakyReLU(0.2, inplace)
(13): Dropout2d(p=0.25)
(14): BatchNorm2d(256, eps=0.8, momentum=0.1, affine=True)
)
(adv_layer): Sequential(
  (0): Linear(in_features=4096, out_features=1, bias=True)
  (1): Sigmoid()
)
)

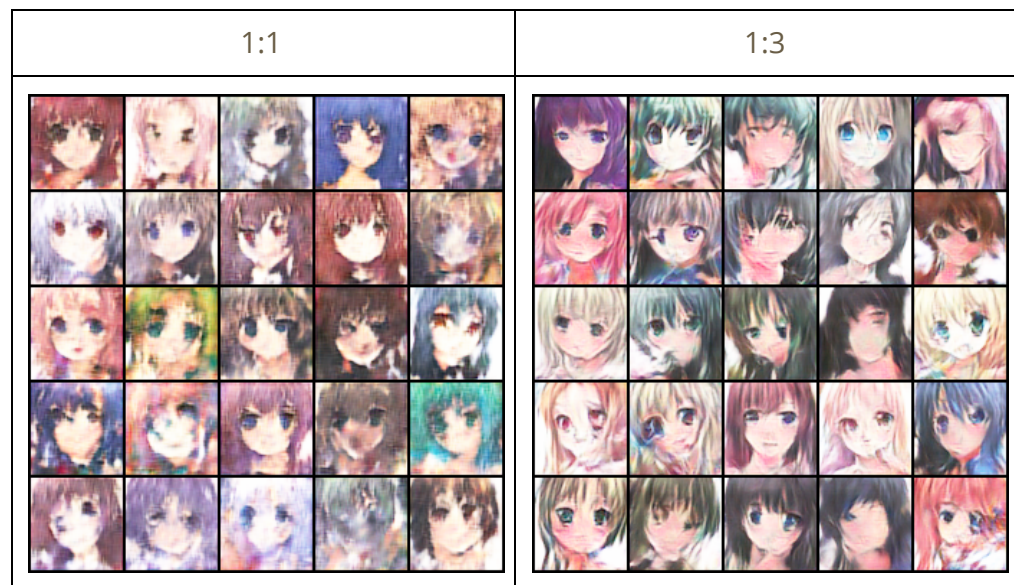
```

### ○ Experiment Settings and Observation

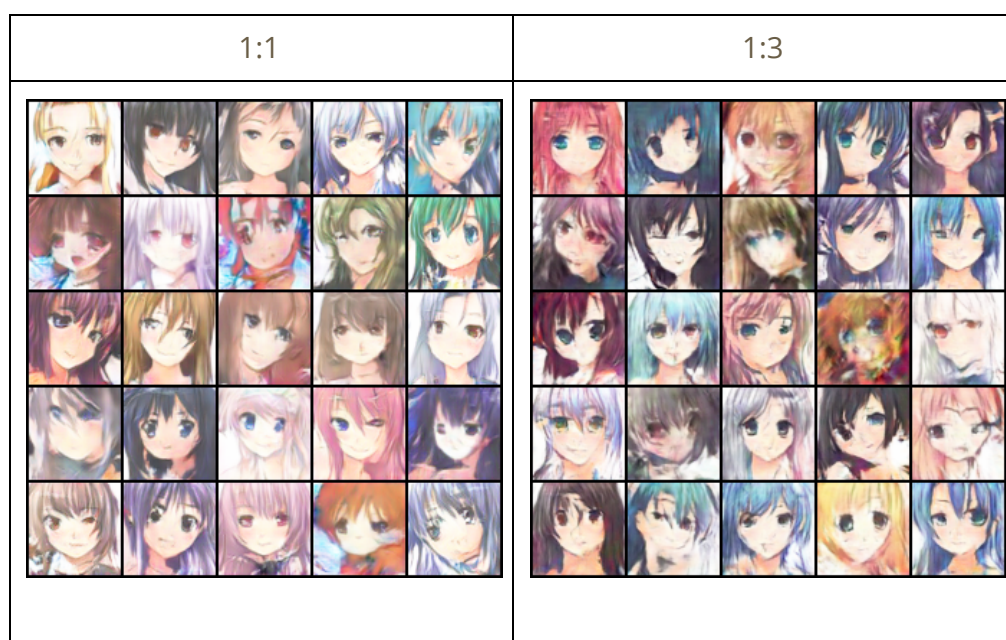
訓練時是把所有圖片都 resize 成 64\*64 後一起下去 train，G 和 D 使用的 optimizer 均為 ADAM，learning rate = 0.0002，beta 為 0.5 及 0.999。

比較 G 和 D training 比例為 1:1 及 1:3 兩種時的情形，可發現 1:3 時 Discriminator 訓練得較好，Generator loss 比較高，會在比較早的時候即產生較清晰的圖片，但是完整 training 過程結束後的成果沒有太大差異。

20 epochs :



200 epochs :



○ **Compare your model with WGAN, WGAN-GP, LSGAN (choose 1)**

我們比較的是 LSGAN，training 過程中 G 與 D 的訓練比例為 1:3，整體感覺稍微亮了一點，其他好像沒有太大的差異。結果如下圖。

Generator 架構：

conv0: [None, 4, 4, 64\*8], with BatchNorm and relu  
conv1: [None, 8, 8, 64\*4], with BatchNorm and relu  
conv2: [None, 16, 16, 64\*2], with BatchNorm and relu  
conv3: [None, 32, 32, 64], with BatchNorm and relu  
imgs: [None, 64, 64, 3], with tanh

Discriminator 架構：

conv1: [None, 32, 32, 64], with BatchNorm and leaky relu  
conv2: [None, 16, 16, 64\*2], with BatchNorm and leaky relu  
conv3: [None, 8, 8, 64\*4], with BatchNorm and leaky relu  
conv4: [None, 4, 4, 64\*8], with BatchNorm and leaky relu  
conv5: [None, 1, 1, 1]  
out: [None], squeeze



○ **Training tips for improvement**

i. **Tip No. 3: using a spherical z**

比較從  $\text{uniform}(-1, 1)$  以及  $\text{normal}(\mu=0, \sigma^2=1)$  distribution sample 出來的  $z$  的差別。

uniform sample 結果如下，基本上和 normal sample  $z$  並沒有看出太大的差別：



ii. **Tip No. 6: using soft and noisy labels**

比較使用 soft labels (true: 0.7~1.2, false: 0~0.3) 和 binary labels (true: 1, false: 0) 的差別。使用隨機亂數產生範圍內的 soft labels 取代原先的 0/1，訓練過程和結果都沒有太大改變，結果如下圖：





### iii. Tip No. 9: using ADAM optimizer

比較 G 和 D 都使用 ADAM、都使用 SGD、以及 D 使用 SGD 且 G 使用 ADAM 的差別。

原先 model 中 G 和 D 使用的 optimizer 即為 ADAM，兩者都改成 SGD 後發現完全 train 不起來，依照 tip 中建議嘗試只讓 D 使用 SGD，發現 D 太強，G 的 loss 都偏高，整體效果不佳。結果如下：



## 2. Text-to-Image Generation

### ○ Model Description

#### **data preprocessing:**

首先是資料處理的方面，由於一開始給的資料label很多很雜，過濾時只接受有一個hair label 和 一個 eyes label的資料，若其中一個缺少超過一種以上就直接刪除，最後在原本的資料中留下11k筆資料左右，而extra的資料因為label很乾淨則直接拿來用。生成negative feature和image的時候就從negative的set中隨機抽取一個來用，幾本上不要抽取到對的就好了。

#### **Model:**

model的部份也是採用cDCGAN下去訓練，condition的方面是對hairs 和 eyes 各作一個one hot vector然後直接concatenate，所以總共加起來是一個22維的vetcor，再過一層fully connected 的 layer之後和100維的noise vector接起來，再加到generator中反向的CNN中去生成圖片。而discriminator則是較為普通的數層CNN + NN 的架構。generator和discriminator中我們都加入了Batch Normalization和dropout下去以求最佳的結果。

#### **Discriminator Loss:**

和3-1的架構比較不同的是，在discriminator的loss這邊總共有4種loss必須考慮4種loss: <真的圖片，對的特徵>，<假的圖片，對的特徵>，<正確的圖片，錯誤的特徵>，<錯誤的圖片，正確的特徵>。訓練時的loss我們採用的是將第一項與其餘三項的平均相加。



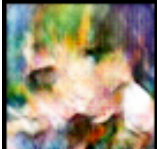




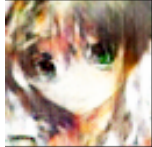
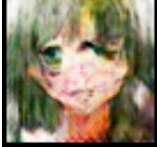
### ○ Experiment Settings and Observation

1: 將feature vector與noise layer直接相連才過NN再接CNN的成果，結果是感覺feature直接被當成noise了所以連特徵都沒學到，所以feature感覺要自己先過一層NN比較好。

2: 讓資料有50%機率水平翻轉，model架好之後直接train下去的結果，有學到特徵，但是發現generator的loss很高，生成圖片的質量也不怎麼高。

best: 讓資料有50%機率水平翻轉，在前50 epoch先加強訓練discriminator然後在之後的epoch中若discriminator loss 小於0.5就不train他，然後加入label smoothing，讓label為1的在0.9~1.1間跳動，0的在0~0.2之間跳動，可以將generator的loss降低許多，並生成較好看的圖片。

collapse: 另外我們發現如果一路加強訓練discriminator最後會直接產生mode collapse，也就是全部都生成一樣的圖片，所以最後改為門檻值而非固定的比例。

label	best	1	2
blue hair blue eyes			
blue hair green eyes			
blue hair red eyes			
green hair green eyes			
green hair red eyes			

### 3. Style Transfer

- Show Your Result



- Analysis

我們使用MSG-Net，可以清楚地將原本的圖轉換成鉛筆素描畫，不過天空左上角的部分色塊比較怪一點。