MLDS HW4 - Generative Adversarial Networks

學號：B04901066 系級：電機三 姓名：洪國喨

1. Model description(2%)

模型：Deep Convolution Auxiliary Classifier Generative Adversarial Network

使用 keras 實作，color tags 使用 one-hot 紀錄，故 hair tag 是12維，eyes tag 是11維。

每個 Epoch 隨機從所有 training data sample 64張圖片，即 Epoch count等價於Batch count。

for both G and D：

訓練D and G次數 - 1：2

Optimizer： Adam(lr=0.0002, beta_1=0.5, beta_2=0.999)

Batch size：64

loss_weights：[1, 0.3, 0.35](實驗參數)

objective function：

"minimize loss 就是 maximize likelihood." (ref to 李弘毅教授上課內容)

模型會minimize weighted total loss.

loss = ['binary_crossentropy', 'categorical_crossentropy', 'categorical_crossentropy']

total loss function 即為 loss_weights * loss 的總和。

訓練 discriminator 時：

輸入 真正的圖片 , 給予 label = [1, 真正的髮色, 真正的眼色] ,

輸入 generator生成之圖片, 給予 label = [0, 無label, 無label]

訓練 generator 時，generator輸出直接當作discriminator輸入，鎖定discriminator：

輸入 [噪音, 隨便選一個髮色, 隨便選一個眼色] 給 generator

給予 discriminator 模型 label = [1, 隨便選一個髮色(同generator輸入), 隨便選一個眼色(同generator輸入)]

即 generator 要盡量使 discriminator 認為自己產生的圖片是該 label。

Generator model：

輸入：[noise 100維, one-hot hair tag, one-hot eyes tag]

輸出：96 * 96 * 3 影像

BatchNormalization momentum = 0.75

Operation	Units/filters	Kernel	Strides	activation
Concatenate三個輸入				
Dense	128*6*6			
Reshape	(6, 6, 128)			
BatchNormalization				
UpSampling2D				
Conv2D	512	3 × 3	1 × 1	ReLU
BatchNormalization				
UpSampling2D				
Conv2D	256	3 × 3	1 × 1	ReLU
BatchNormalization				
UpSampling2D				
Conv2D	128	3 × 3	1 × 1	ReLU
BatchNormalization				
UpSampling2D				
Conv2D	64	3 × 3	1 × 1	ReLU
BatchNormalization				
Conv2D	3	3 × 3	1 × 1	Tanh

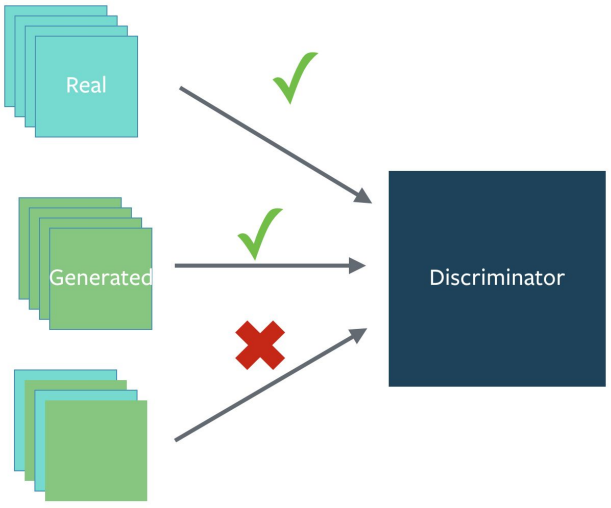
Discriminator model：
輸入：96 * 96 * 3 影像
輸出：[是否是真的, one-hot hair tag, one-hot eyes tag]
Leaky ReLU slope = 0.2
BatchNormalization momentum = 0.75

Operation	filters	Kernel	Strides	Dropout	activation
Conv2D	32	3 × 3	2 × 2	0.25	LeakyReLU
Conv2D(ZeroPadding2D)	64	3 × 3	1 × 1	0.25	LeakyReLU
BatchNormalization					
Conv2D	128	3 × 3	2 × 2	0.25	LeakyReLU
BatchNormalization					
Conv2D	256	3 × 3	1 × 1	0.5	LeakyReLU
BatchNormalization					
Conv2D	256	3 × 3	2 × 2	0.5	LeakyReLU
Flatten					
三個輸出 Dense					sigmoid softmax softmax

2. How do you improve your performance (2%)

參考：https://github.com/soumith/ganhacks

(1) BatchNorm：
訓練 discriminator 時，每個 batch 裡必須 全部都是真的影像 或 全部都是生成的影像。若是不這樣處理，最後 generator 很容易就一直輸出雜訊。(圖片來源：https://github.com/soumith/ganhacks/blob/master/images/batchmix.png)



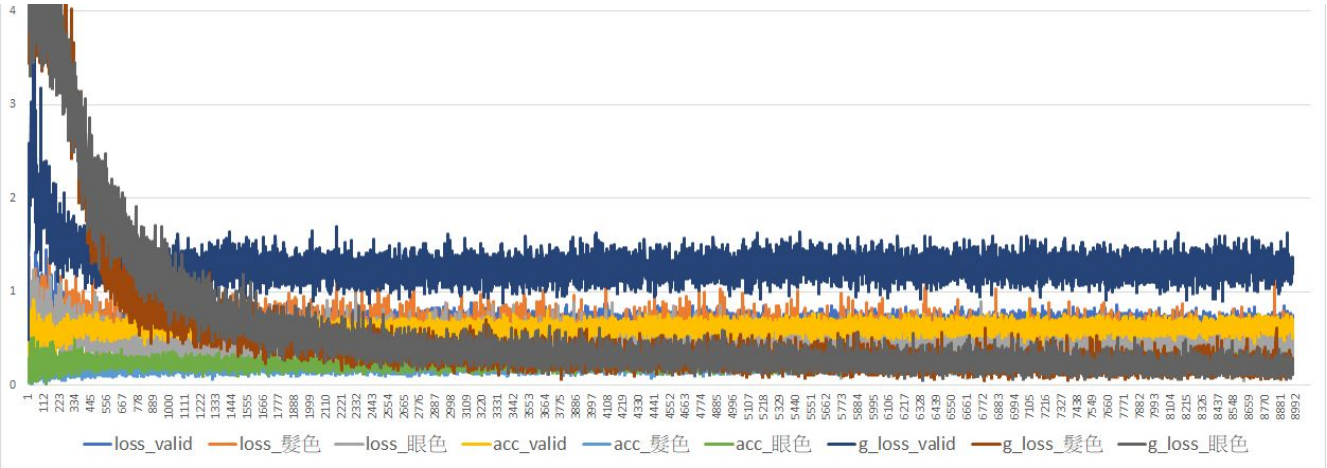
(2) 避免Sparse Gradients：使用ReLU
如果有Sparse Gradients，GAN 穩定性會受到影響，在 generator 和 discriminator 皆使用 ReLU 可以有效改善。

(3) 使用 DCGAN

(4) 提早發現訓練失敗：
discriminator loss 跑到 0：可以砍掉了
檢查 norms of gradients：如果超過100表示有問題
訓練順利的話 discriminator loss variance 很低且穩定極緩下降
如果 generator loss 持續穩定下降，可能是 generator 找到奇怪的 pattern可以一直騙過 discriminator。

(5) 不要動態的依照loss來調整 generator 和 discriminator 訓練次數：很容易就爛掉。

(6) generator loss 不正常增加：
可能代表 discriminator 把所有 training data 真實的影像都背下來了(overfitting)，
可透過在真實的影像加一些noise處理。(參考：ttps://github.com/soumith/ganhacks/issues/14)



正常的訓練過程loss：因為髮色之權重較低，故loss較高。

(7) 用人眼看輸出結果：
因為要peer review，顯然用人類判斷會比較好。

(8) 移除特定tags：
我發現有 polychromatic 這個 tag 的圖大多是只有線條或單色的，把這些我不喜歡的圖移除，應該可以使輸出更接近我的理想，其他也有一些 tag 可以輔助訓練。

(9) 使用 loss weights：
讓 model 更專注於圖片的真實度，真實度 train 起來後再讓顏色 tags loss 慢慢下降。

3. Experiment settings and observation (2%)

(1) 一些實驗失敗的例子：

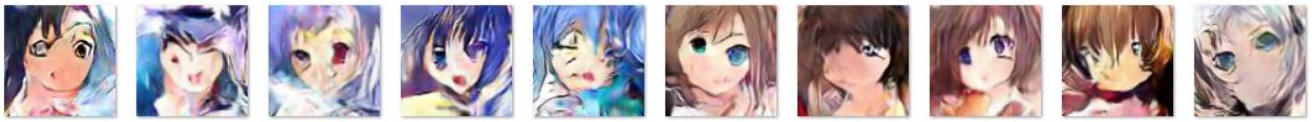
(a) generator model 的 loss 持續小於 0.5，generator 找到奇怪的 pattern可以一直騙過 discriminator，可發現到顏色的tag基本上是正確的->可能是color tag loss weight太大。

(b) 沒作BatchNorm，輸出都是雜訊。

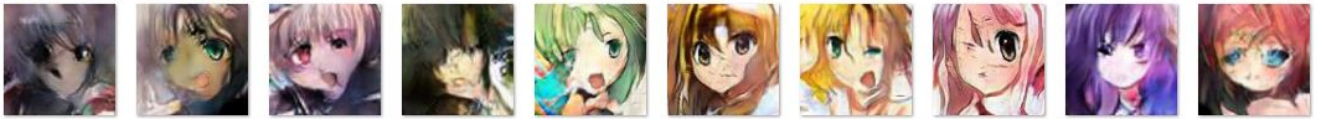
(2) 調整 loss_weights：
大約 [1, 0.1, 0.1] 到 [1, 0.8, 0.8] 都是train得起來的，結果其實也沒有很明顯的差異，唯調整後兩者大小關係，會影響model關注的地方，例如 [1, 0.1, 0.2] ，就會使輸出眼睛部分很清楚，其他部分變得較模糊。

(3) 改變 generator 輸入noise 高斯分布之 σ ：
σ = 0.1

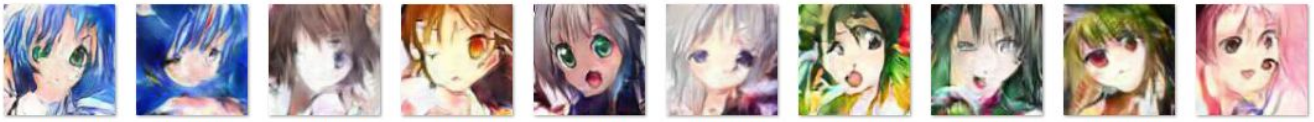
σ = 0.2



σ = 0.5



σ = 1

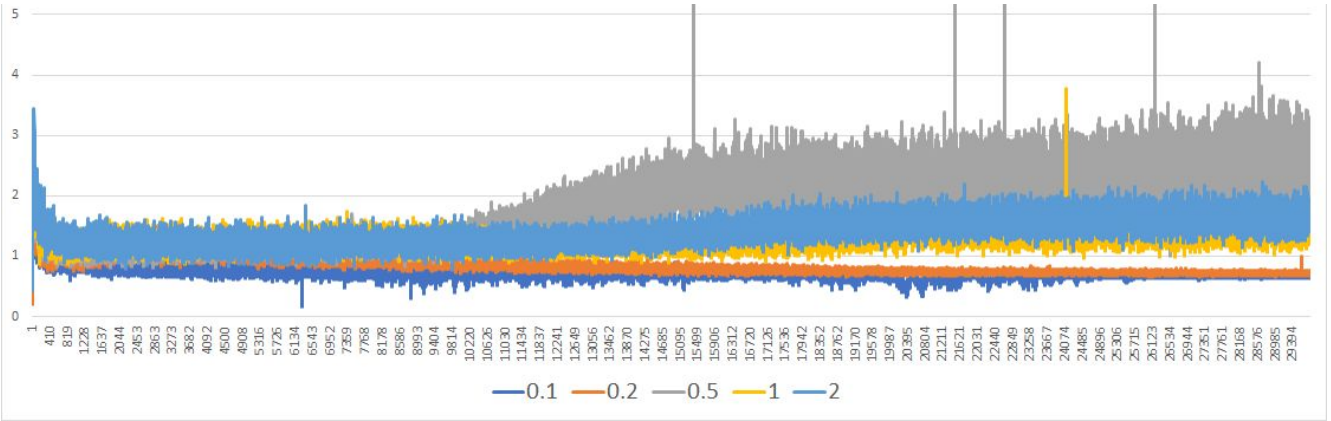


σ = 2

30000 epoch時之loss和acc：

σ	loss valid	loss 髮色	loss 眼色	acc valid	acc 髮色	acc 眼色	g_loss valid	g_loss 髮色	g_loss 眼色
0.1	0.7303	0.2668	0.2053	0.3594	0.25	0.2188	0.6531	0.0076	0.0069
0.2	0.721	0.3892	0.2459	0.3828	0.2031	0.25	0.7364	0.0106	0.0206
0.5	0.5436	0.3144	0.3494	0.8047	0.1797	0.2578	2.7546	0.3158	0.1287
1	0.602	0.5645	0.4471	0.6562	0.2656	0.2422	1.5844	0.1037	0.0498
2	0.5559	0.3878	0.2348	0.6953	0.2266	0.2734	1.6461	0.2675	0.1324

可發現σ = 0.1和 0.2 之 g_loss_valid 較低，圖片也較醜(個人主觀)，我覺得σ = 0.5 ~ 2 的圖較好看(個人主觀)，此外 loss 太大(如σ = 0.5)可能眼睛點錯之機率較高(個人主觀)，可見 g_loss_valid 對於訓練狀況之顯示，具有很重要的指標性。



g_loss_valid：不同 σ 之訓練過程 g_loss (到三萬個epoch)

4. Bonus (2%)

style-transfer

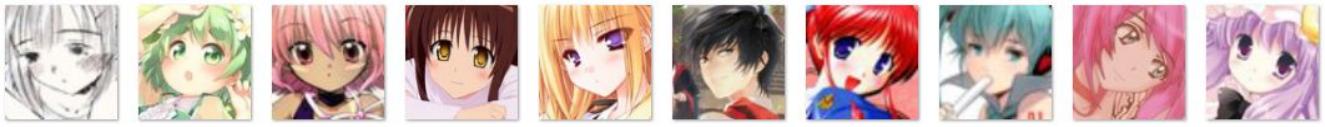
source dataset：助教提供之動畫頭像

target dataset：浮世繪 (日文漢字：浮世絵、日文假名：うきよえ、日文拼音：Ukiyo-e) 圖片，來源 Wikiart (https://www.wikiart.org/)：總共1433張

使用 github repo 的 code：https://github.com/xhujoy/CycleGAN-tensorflow

用 GTX 1080ti 約訓練三小時。

輸入：



輸出：

輸入：

輸出：

輸入：

輸出：

輸入：

輸出：