HW3

王垣尹 許哲瑋 李中原

3-1 Image Generation

1. Model Description

主要使用的是 DCGAN 的架構(預先把圖片壓縮成 64x64x3):

Discriminator:

1. input = (64, 64, 3)
2. Conv2D(32, kernel = 4, stride=2)
3. BatchNorm + LeakyRelu + dropout
4. Conv2D(64, kernel = 4, stride=2)
5. BatchNorm + LeakyRelu + dropout
6. Conv2D(128, kernel = 4, stride=2)
7. BatchNorm + LeakyRelu + dropout
8. Conv2D(256, kernel = 4, stride=1)
9. BatchNorm + LeakyRelu + dropout
10. Flatten
11. Dense(1, sigmoid)

Generator

1. input = (100,)
2. Dense(512*8*8)
3. Reshape((8, 8, 512))
4. Conv2D_Transpose(256, kernel = 4, stride=2)
5. BatchNorm + LeakyRelu + dropout
6. Conv2D_Transpose(128, kernel = 4, stride=2)
7. BatchNorm + LeakyRelu + dropout
8. Conv2D_Transpose(64, kernel = 4, stride=2)
9. BatchNorm + LeakyRelu + dropout
10. Conv2D_Transpose(3, kernel = 4, stride=1)
11. BatchNorm + LeakyRelu + dropout
12. tanh

- Objective function (Loss function)

Discriminator Loss :

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)})+\log(1-D(G(z^{(i)})))]$$

Generator Loss:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}[\log(1-D(G(z^{(i)})))]$$

2. Experiment Setting and Observation

BatchSize = 100, Training Epochs = 300, Discriminator 和 Generator update 比率為 5:1 ; Dropout 層 (Keep Prob 為 0.8) ; Training 過程基本上滿穩定的，差不多在一百多 epoch 的時候就可以產生不錯的圖像; (baseline model 大概可以辨認 15~23 張圖片)。但後來的兩百個 epoch 有調整訓練的週期(Discriminator 和 Generator update 比率改成 3:1)。

3. Compare Model with WGAN, WGAN-GP, LSGAN

我們選擇和 WGAN 做比較。Generator 和 Discriminator 的架構基本上和 DCGAN 的設計相同，唯一不同的便是在 Objective Function(Loss Function) 上面，用距離去衡量(Discriminator 的 output 不再是機率)。

$$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$$

$$L_G^{WGAN} = E[D(G(z))]$$

使用 Weight Clip 的方式來對 WGAN 的 Lipschitz 做限制，實作的時候這個值我們設在(-0.05, 0.05)。WGAN 的訓練有點不太穩定，一開始把 clip 的值設得太小，結果輸出如同電視機壞掉一般的黑白雜訊，但把 clip 值設的太大，會發

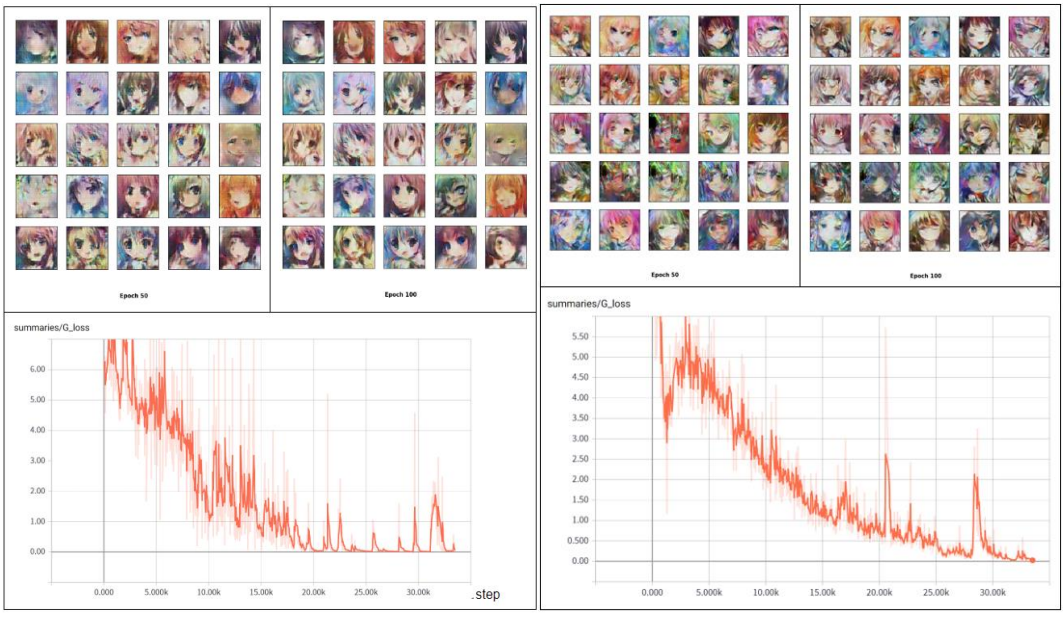
現訓練過程會有正常圖片和半綠屏雜訊的擺盪過程，因此這個值重新調整了幾次，可想而知 Weight Clip 對於限制 Lipschitz 太過於強硬。下圖為 WGAN 訓練 300 epoch 結果，和 DCGAN 比起來似乎模糊的部分以及圖片的質地都較差一些。

Vanilla GAN	WGA
	

4. Training Tips for Improvement

(tip 4)

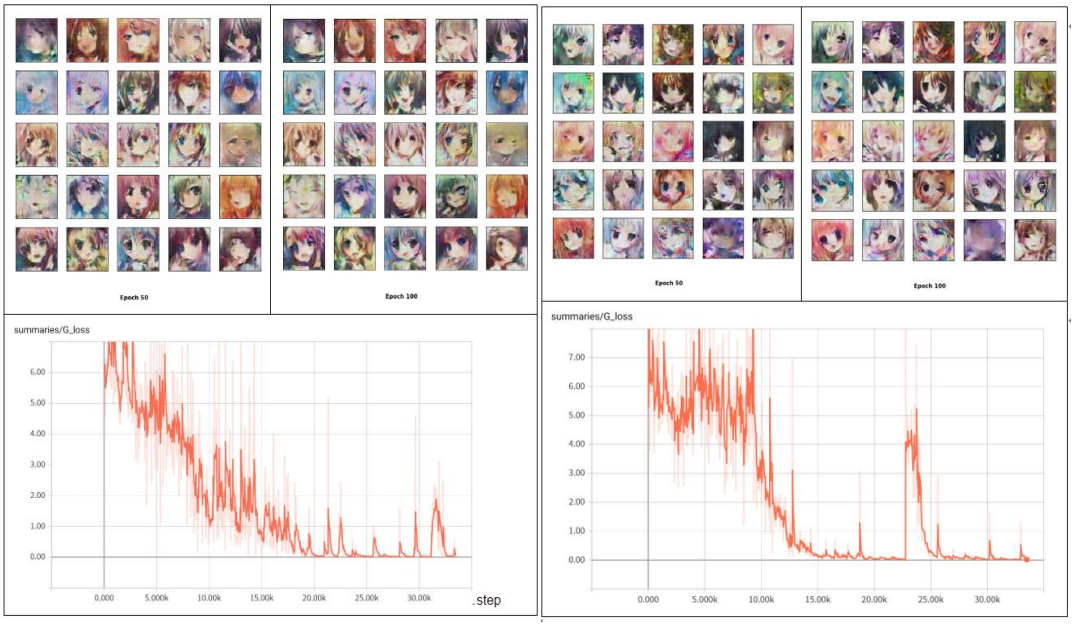
With Batch Normalization　　Without Batch Normalization



從圖可以看出，Generator 訓練的 loss 下降曲線大致相同，但是圖片的生成上，有加上 BatchNorm 會比較完整，沒有加上 BatchNorm 的則會有些綠屏的感覺。

(tip5) Avoid Sparse Gradient: relu, maxpool

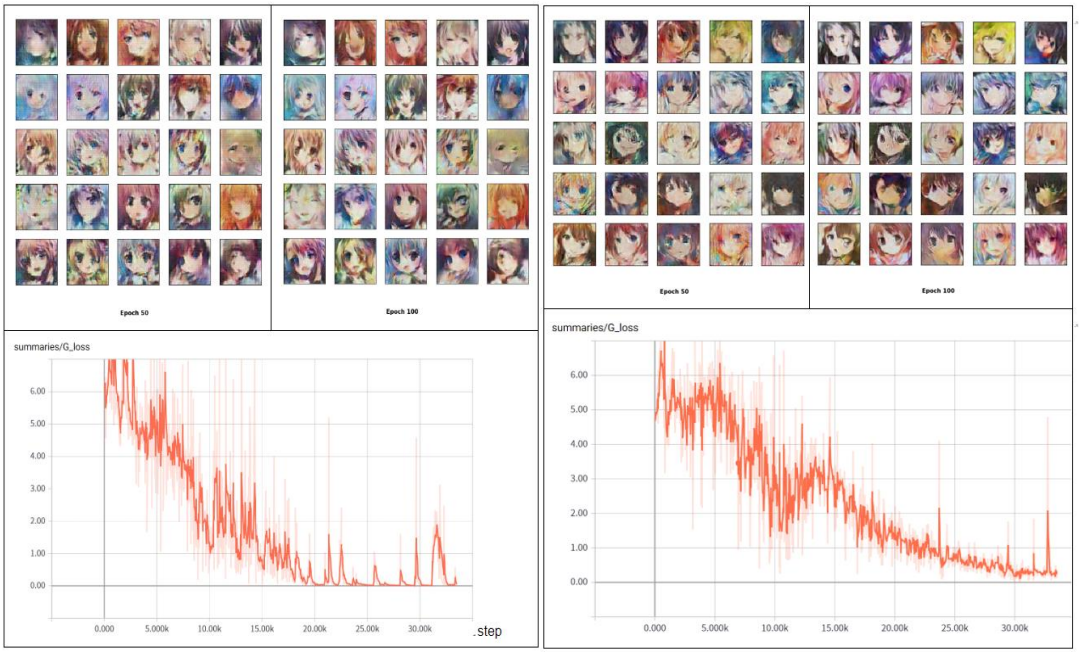
With Leaky Relu　　Without Leaky Relu



左圖是使用 Leaky Relu，對比於右圖使用 Relu，從照片看來差距並不大，但是 generator loss curve 似乎有比較平滑的傾向。

(tip 17) (training and testing in G)

Without dropout　　With dropout(keep prob 0.8)



右圖是在 generator 的層與層之間加入 dropout，基本上 dropout 如果沒有拔掉太多神經元，效果看起來差不多。但我們也試過把 dropout 開到 0.5，結果會完全學不起來(手腳被綁住太多)。

3-2 Conditional GAN

Model Description:

CGAN + DCGAN:

我們照著提供的架構，只是在 Discriminator 最後一層將 Dense 改為 Convolution，另外我們頭髮跟眼睛顏色的 Embedding 分成兩個矩陣來 train，是平行且獨立的。 Embedding Size 為 30+30 = 60，後來發現如果 Embedding Size 太大這樣 Random Z 就會被 Dominate，結果就是非常容易 Mode Collapse，產生出來的圖幾乎都是一樣的臉型。

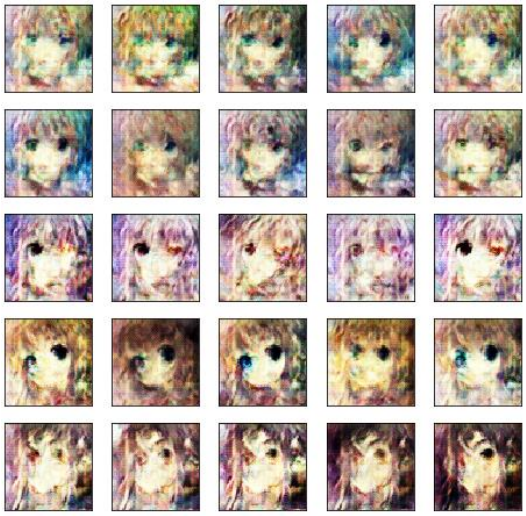
Result

CGAN + DCGAN:

結果如下圖，我們可以發現，CGAN 在訓練的時候收斂得很快(DCGAN+ Condition)，所以有沒有 train 起來其實一開始前幾個 epoch 就會知道。我們的經驗是，CGAN 很容易在 train 好之後 Collapse，所以基本上，以我們的 Model 而言，最好的結果都落在第 100~150 個 epoch 之間，超過反而效果很差，儘管可能 Loss 仍然在下降。

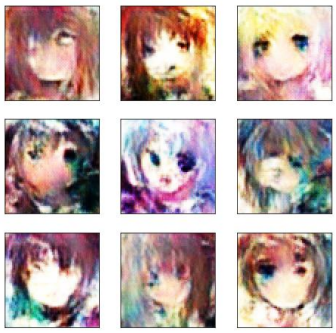
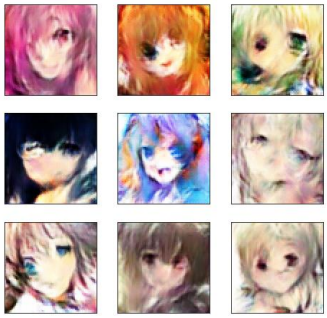
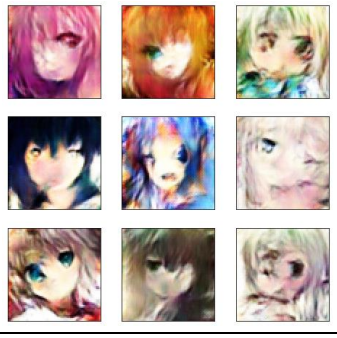
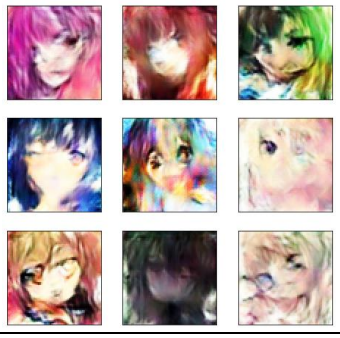
Epoch 50

Epoch 100

	
Epoch 150	epoch 160(Mode Collapse)

CGAN + WGAN:

一換成 WGAN 之後，最明顯的直接影響就是收斂速度變得”超級”慢，基本上前 50 個 epoch 幾乎都看不到一張好的圖片，到第一百個 epoch 時勉勉強強，但很多頭髮顏色都還是錯的如下圖，

		
Epoch 50	Epoch 100	Epoch 150
		
Epoch 200	Epoch 250	

參數討論:

CGAN + DCGAN:

1. Learning Rate:
   我們試過 0.0002 及 0.0005，覺得效果沒有明顯的差異，因為收斂速度都很快，所以沒有做太多的比較
2. Generator epoch vs Discriminator epoch:
   我們試過許多種不同的組合，發現效果真的不太一樣；首先 G 大 D 小的結果是完全不行，G=D 時效果不太好，而且 train 很慢，剩下的就是 G<D，這時我們發現在 G1D3 的情況下結果顯著進步，接著 G1D4 G1D5 都有滿布錯的結果，比較的話覺得見仁見智，我們一值提升到 1:8，結果反而沒有 1:5 左右來的清楚。
3. Condition Embedding size:
   100->50->20->15->10，發現大約在 15~20 之間看起來感覺最好。
4. Random parameter Initialize range:
   z 的大小從[-0.1,0.1] ->[-1,1]之後有變更好 train。
5. Batch size:
   我們試過 128, 256，256 雖然真的跑得比較快，但是效果真的比較不好，而且不知道為什麼，更容易 Mode Collapse。
6. Discriminator 最後一層架構 (Dense vs Convolution):
   儘管 Dense 看起來擁有更高的表現力(參數較多)，我們嘗試的結果是，Convolution 遠較 Dense 好。

CGAN + WGAN:

1. Learning Rate:
   RMSprop with learning rate 0.00005
2. Generator epoch vs Discriminator epoch:
   我們試過 1:5 和 1:8，最終結果來看，1:5 会比較好，猜測可能是 Discriminator train 太好還是會影響 Generator 的學習。(儘管 WGAN 應該緩解了這樣的問題)
3. Weight Clipping Range:
   我們將 weight clip 在正負 0.01 之間。

	
WGAN 2 embedding (1:5)	WGAN 2 embedding (1:8)

改良(Final Result):

Embedding size: 25*2

Noise size and range: 113, random normal with mean 0 and std. 1

Learning rate: 0.0002 with Adam Optimizer

Generator epoch vs Discriminator epoch: 1:5

	
50 epoch	100 epoch

HW3-3

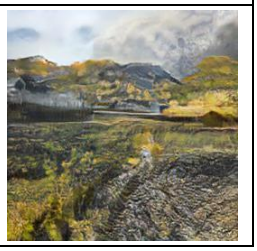
- Show your result (1%) (Two domain and transfer result)

在這個小節裡我們使用的是 Cycle-GAN。兩個 Domain 分別為 Domain A (Vangogh), Domain B(photo)。

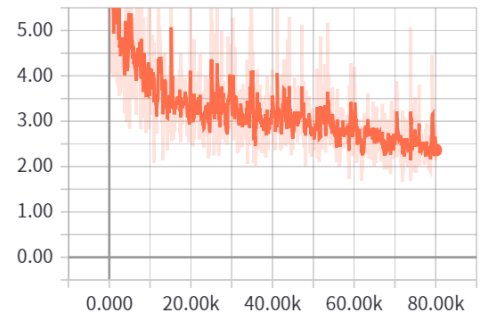
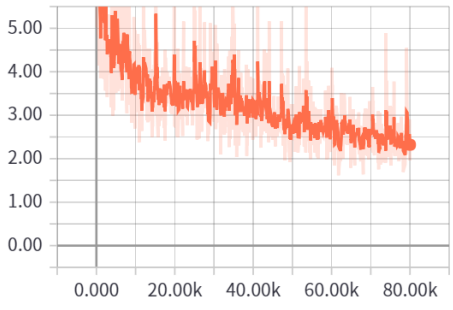
(1) Domain B transform to Domain A

Domain B			
Domain A			

(2) Domain A transform to Domain B

Domain A			
Domain B			

- Analysis (1%) (Which model you use and your observation)

Generator Loss(vangogh2photo)	Generator Loss(photo2vangoph)
	

Observations:

只要做跟人臉有關的 style transfer 結果都像是相機負片的效果，天空的部分，就如同其他有花紋的地方一樣，都有很明顯梵谷風格(捲捲雲般)，儘管我們無法確定 network 有沒有學到任何跟 2D 頻率有關的內容(因為很有可能 Network 實際上做的是單純 pixel wise 的 RGB mapping)，但的確讓人很有梵谷風的感覺。

分工表:

	王垣尹	許哲瑋	李中原
HW3-1 Code		**V**	**V**
HW3-1 Report	**V**	**V**	
HW3-2 Code	**V**		
HW3-2 Report	**V**		**V**
HW3-3 Code		**V**	
HW3-3 Report	**V**	**V**	
Uploading Details	**V**	**V**	**V**