

MLDS HW3 Report

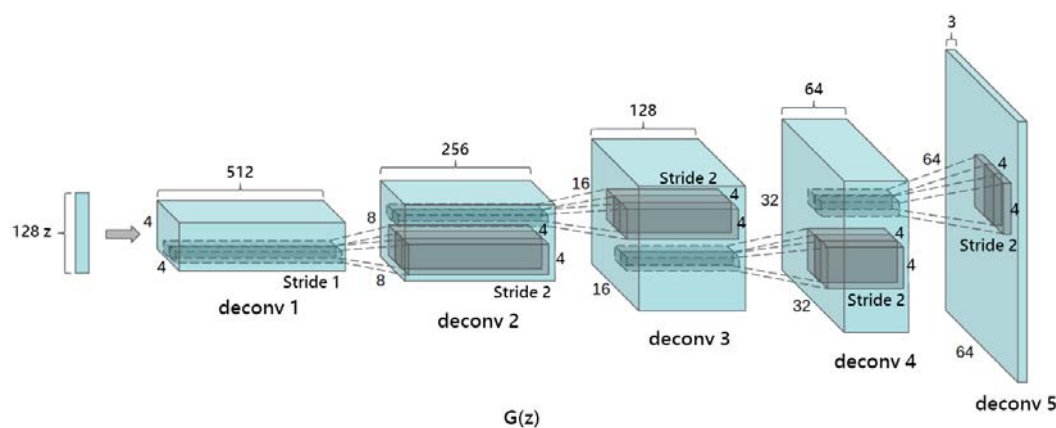
林宣竹 B03901065 楊其昇 B03901101 郭恆成 B03901145

I. HW 3-1

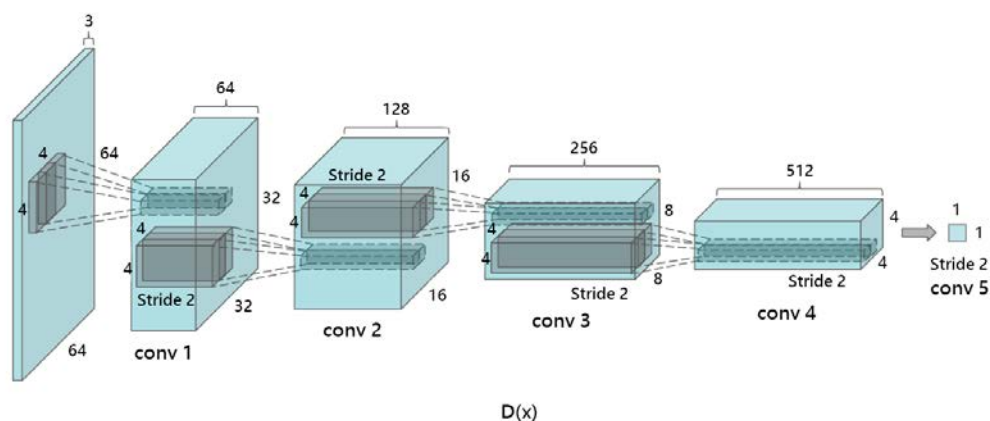
1. Model Description

實作 DCGAN 的架構 (參考: <https://github.com/pytorch/examples>)，值得注意的是，架構中完全沒有使用 fully-connected layer。

如下圖所示，Generator 的部分一開始輸入為 128 維的 noise (將 noise 視為有 128 個 1x1 的 feature map)，經過 4x4 的 filter 進行 deconvolution 將圖片 upsampling，總共有五層的 deconvolution layer，最後輸出 64x64x3 的圖片。其中每層 deconvolution 輸出都會依序經過 batch normalization、ReLU 的處理，但在最後一層的 activation function 改為 tanh。



Discriminator 架構和 Generator 類似，主要是進行 Generator 的反向操作：deconvolution 換成 convolution，共五層。其中 filter 一樣是 4x4，但 filter 的個數依序為 64、128、256、512 和 1。此外在 convolution 結束後，會依序進行 batch normalization、Leaky ReLU (leakage = 0.2) 的處理，而最後一層的 activation function 則為 sigmoid。



Generator 和 Discriminator 訓練時的 objective function 各自如下，其中 x^i 為隨機 sample 的真實圖片， z^i 為隨機 sample 的 128 維 $N(0, 1)$ 的 noise：

$$\text{Discriminator: } \max_D \frac{1}{m} \sum_{i=1}^m \log(D(x^i)) + \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^i)))$$

$$\text{Generator: } \min_G \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^i)))$$

在實際操作的時候，一般希望要 minimize objective function，另外在當 input 幾乎是 0 的時候 log function 會產生 Nan，因此實際上訓練的 objective function 如下，其中 ϵ 是一個很小的數字，此實驗中設為 $1e-35$ ：

$$\text{Discriminator: } -\frac{1}{m} \sum_{i=1}^m \log(D(x^i) + \epsilon) + \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^i)) + \epsilon)$$

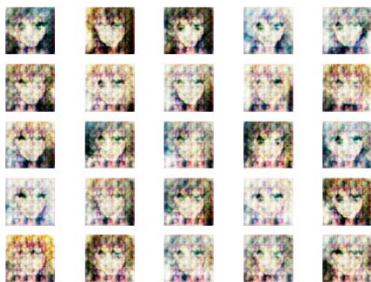
$$\text{Generator: } \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^i)) + \epsilon)$$

並藉由選擇的 optimizer 去分別 minimize 上述的 loss。

2. Experiment settings and observation

baseline model 架構如第一題所述，訓練時使用 Adam optimizer，不論 Generator 或 Discriminator 的 learning rate= $2e-4$ 、 $\beta_1=0.5$ ，其餘為預設值。Input image 輸入 network 前均會除以 255 做簡單的 normalization。每個 training step 均會訓練一次 discriminator 再訓練一次 Generator，共訓練 100 個 epoch。在輸出圖片時，會先將 Generator 的 output 值乘上 255，再將小於 0 或大於 255 的部分 clamp 成 0 或 255。訓練過程中產生的圖片如下：

Epoch 1



Epoch 10



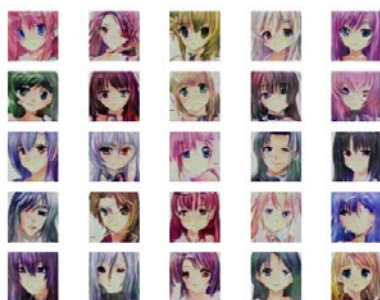
Epoch 20



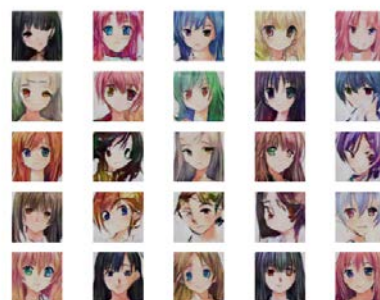
Epoch 30



Epoch 40



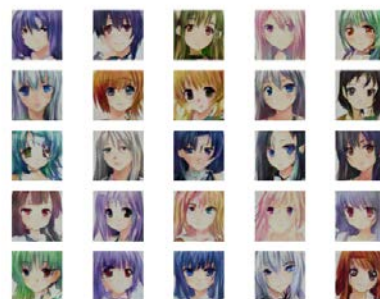
Epoch 50



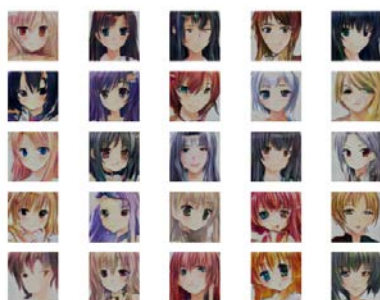
Epoch 60



Epoch 70



Epoch 80



Epoch 90

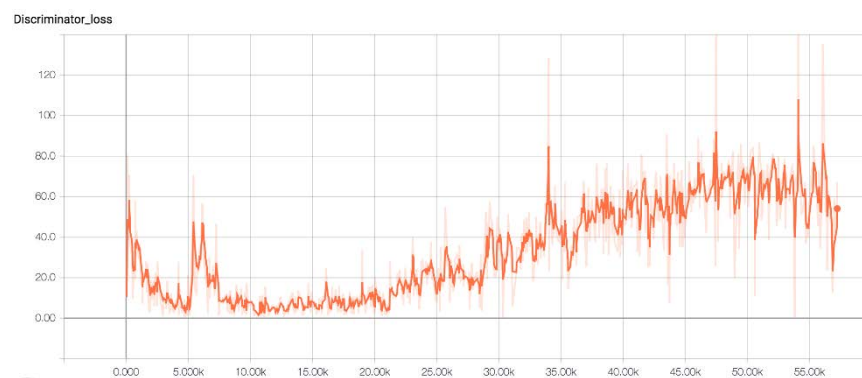


第 100 個 epoch 產生的結如下:

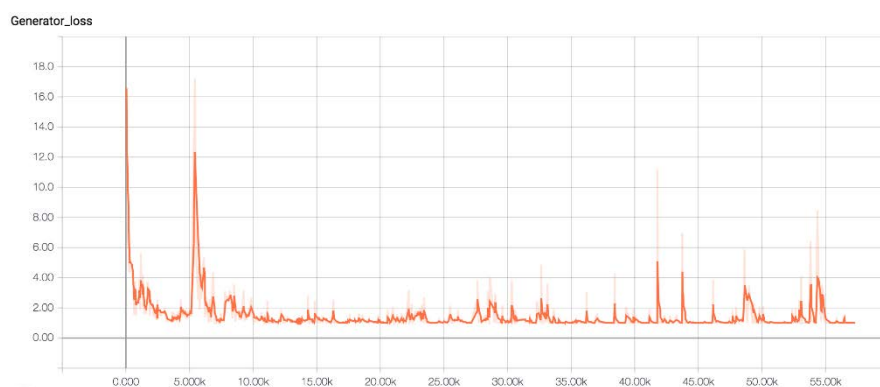


而實驗的設定為 image augmentation，將訓練 discriminator 所需的圖片每次隨機選轉 $+10^{\circ} \sim -10^{\circ}$ ，interpolation 的模式則選擇”nearest”，其餘設定則完全一樣保持不變。訓練過程的 loss 變化如下：

Discriminator:

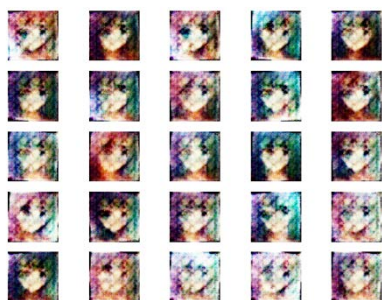


Generator



訓練過程中產生的圖片如下：

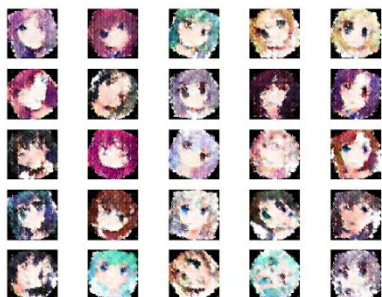
Epoch 1



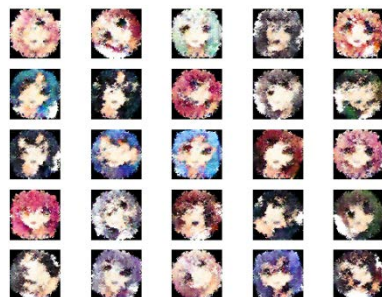
Epoch 10



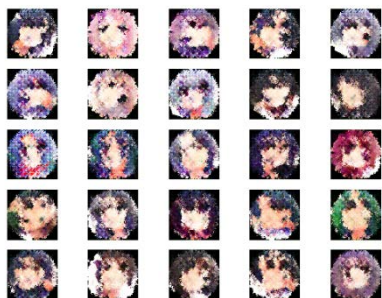
Epoch 20



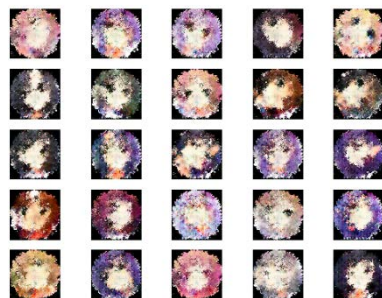
Epoch 30



Epoch 40



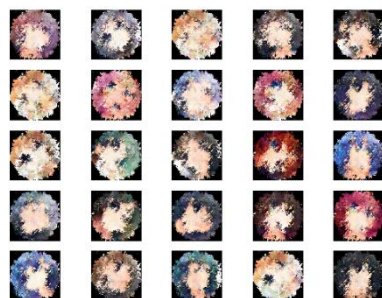
Epoch 50

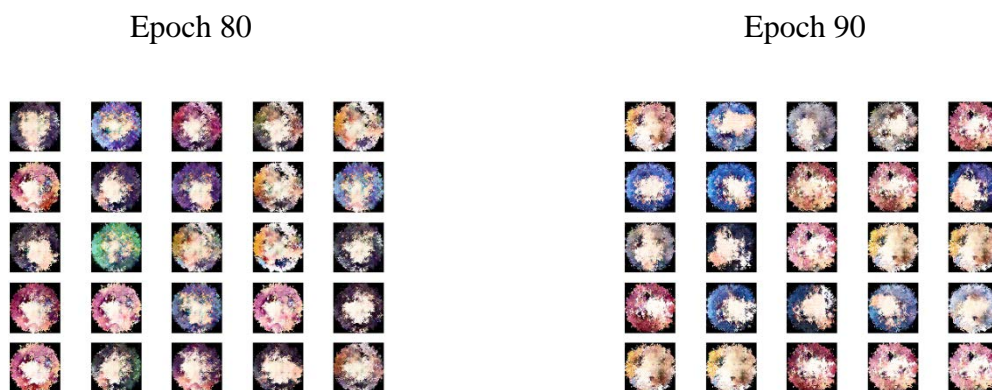


Epoch 60



Epoch 70





對照 loss 的訓練過程，可以發現大約在第 10 個 epoch 的時候，Generator 的 loss 突然往上衝，以下為第 9, 10, 11 個 epoch 所產生的圖片：



和 Generator 的 loss 向上衝的時間點相對應，可以看到第 10 個 epoch 的時候不知道出了什麼事，導致 Generator 和 Discriminator 的 loss 雙雙爛掉，雖然後來看似 Generator 和 Discriminator loss 似乎又慢慢下降，但根據產生的圖片可以發現應該是 Discriminator 整個爛掉，他無法分辨選轉的圖片和 Generator 產生一團圓圓受扭曲的圖片的差別（或許也可以說 Generator 找到一種方式可以騙過 Discriminator），這應該也是訓練後期 Discriminator 一直降不下來的原因。

II. HW 3-1 Compare with WGAN

1. Model Description

使用 model 為 WGAN，架構基本上和 DCGAN 一模一樣，只有 Discriminator 的 output layer 的 activation function 從 tanh 換成 identity。

訓練時的 objective function 分別如下：

$$\text{Discriminator: } \frac{1}{m} \sum_{i=1}^m D(G(z^i)) - \frac{1}{m} \sum_{i=1}^m D(x^i)$$

$$\text{Generator: } \frac{1}{m} \sum_{i=1}^m -D(G(z^i))$$

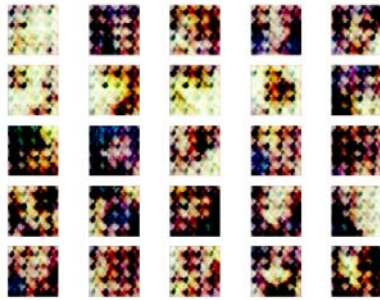
訓練時的 Optimizer 參考原始 paper，使用 RMSProp optimizer 且 learning

rate = 5e-5，更新完參數後，會針對 discriminator 的參數 clip 到 ± 0.01 。
訓練時每 5 個 training step 才會更新一次 Generator，但每個 training step 都會更新 Discriminator。

2. Result of the Model

訓練過程中產生的圖片如下：

Epoch 10



Epoch 20



Epoch 30



Epoch 40



Epoch 50



Epoch 60



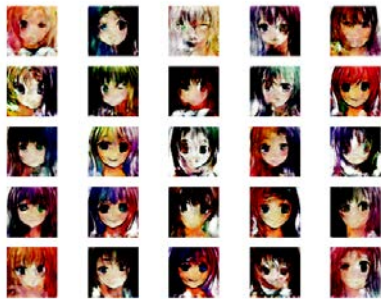
Epoch 70



Epoch 80



Epoch 90



Epoch 100

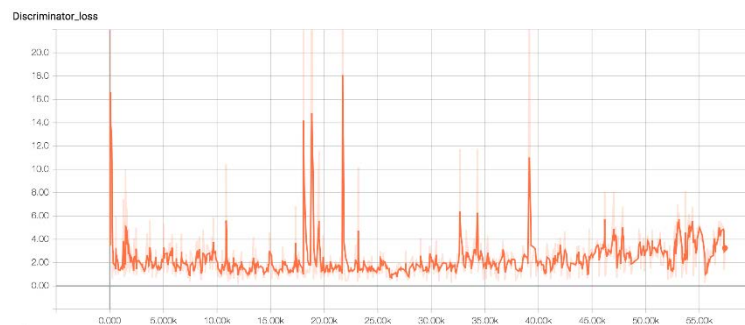


雖然仍然看得出來產生的圖片是動漫人物，但是畫風相對於 DCGAN 而言就粗獷了許多。(DCGAN 像用色鉛筆畫的，WGAN 則像蠟筆畫或是油畫)

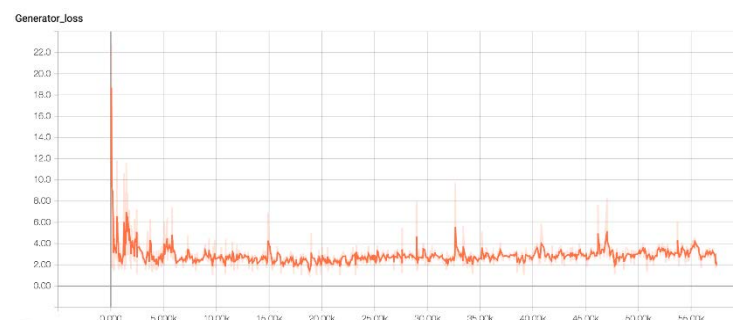
3. Comparison Analysis

DCGAN 訓練過程的 loss 變化如下：

Discriminator:

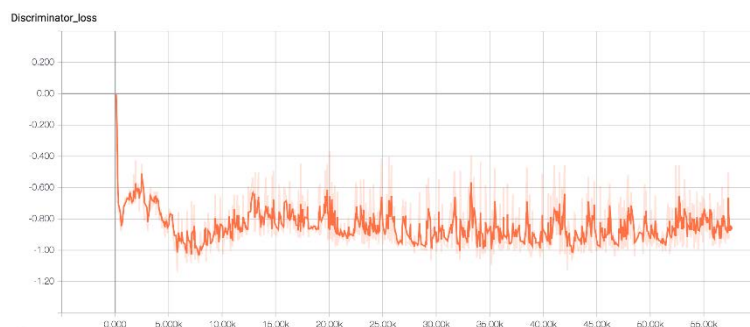


Generator:

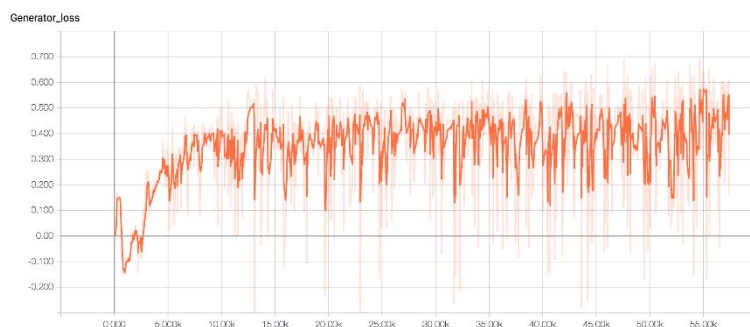


WGAN 訓練過程的 loss 變化如下：

Discriminator:



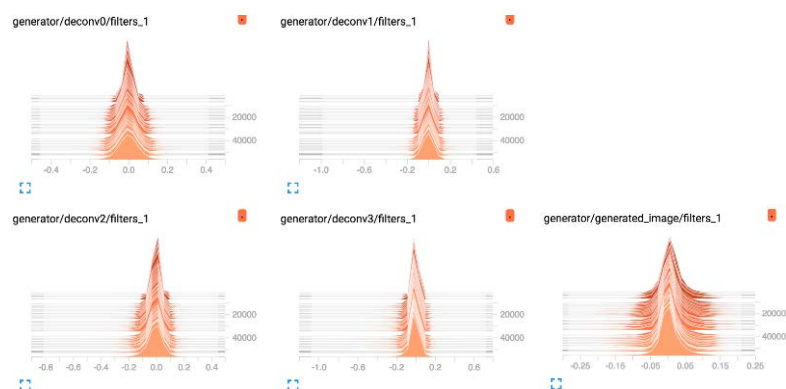
Generator:



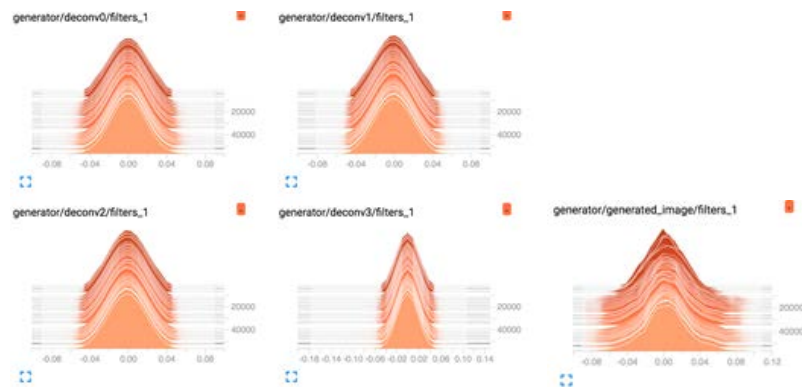
雖然兩種 GAN 的 loss function 不一樣，直接比較兩者的數值沒有什麼意義，但仍然可以看得出來，DCGAN 的 loss 相較於 WGAN，變化明顯比較劇烈(兩者圖的縱軸 scale 不同)。此外，WGAN Generator 可以看得出來，Generator 的 loss 卡在某個地方降不下來。

兩種 GAN 唯一完全相同的部分便是 Generator，此外 Generator 也和最後產生的圖片品質息息相關，因此簡單觀察兩者 Generator 的訓練過程的分佈。

DCGAN 的 Generator:



WGAN 的 Generator:



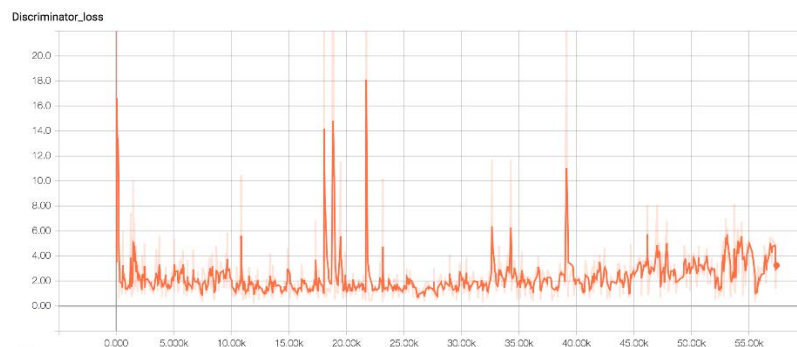
兩種 GAN 的 Generator 產生的 filter 參數分佈都滿接近 truncated 的常態分佈(truncated normal distribution)，差別在於 distribution 的標準差。

觀察產生圖片品質比較好的 DCGAN Generator，可以發現每一層的 filter 的參數分佈，都傾向集中分佈在 ± 0.02 之間。但是 WGAN 的 Generator 產生的 filter 的分布會是比較散的，而這個情形或許可以勉強解釋為什麼 WGAN 會產生比較偏深色的圖片(因為經過 filter 所產生的值相對 DCGAN 要來得大)。

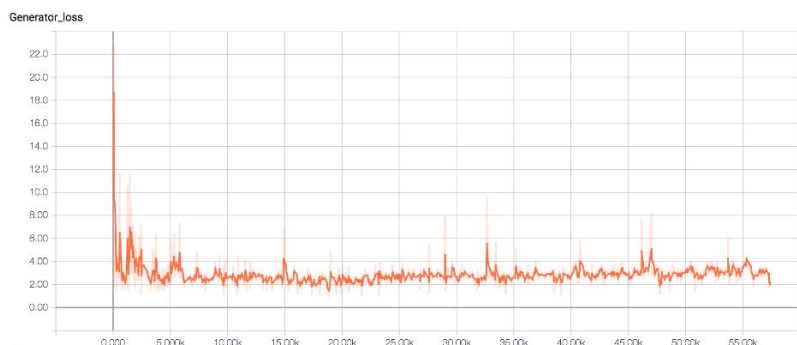
III. HW 3-1 Training Tips for Improvement

使用第一和二題所提到的模型架構和實驗細節當作參考模型(baseline model)，其中相關的訓練過程如下：

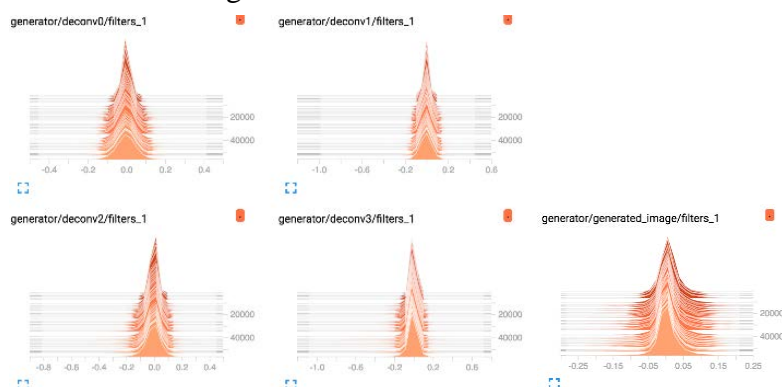
Discriminator loss:



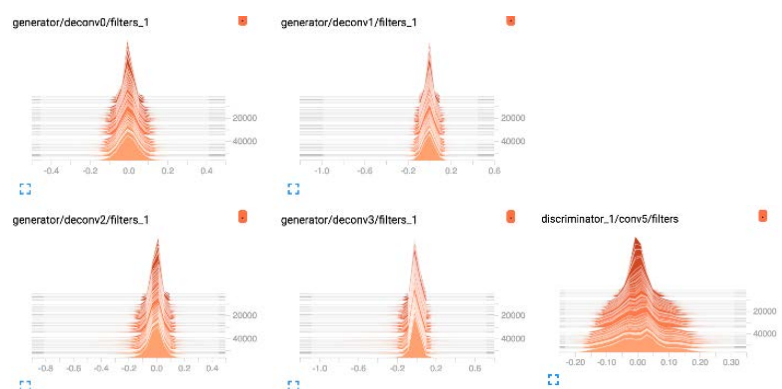
Generator loss:



Parameters distribution of generator filters:



Parameters distribution of discriminator filters:



實驗的 3 個 Training tips 分別如下：

Tip #2: Modified Loss function

將 Generator 的 loss function 改成 $\frac{1}{m} \sum_{i=1}^m -\log(D(G(z^i) + \epsilon))$ ，其餘維持不變。

訓練過程中產生的圖片如下：

Epoch 10



Epoch 20



Epoch 30



Epoch 40



Epoch 50



Epoch 60



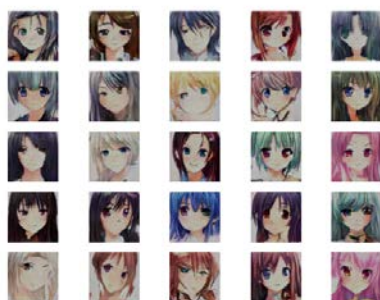
Epoch 70



Epoch 80



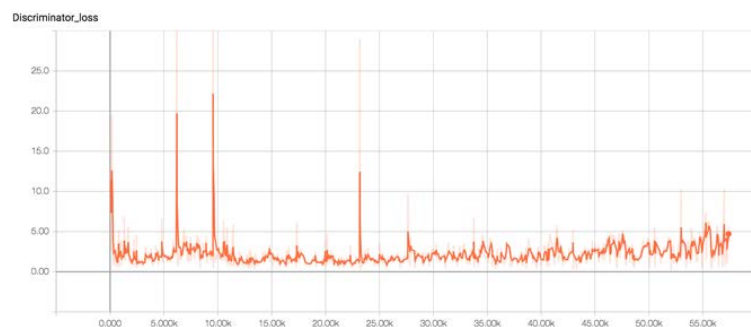
Epoch 90



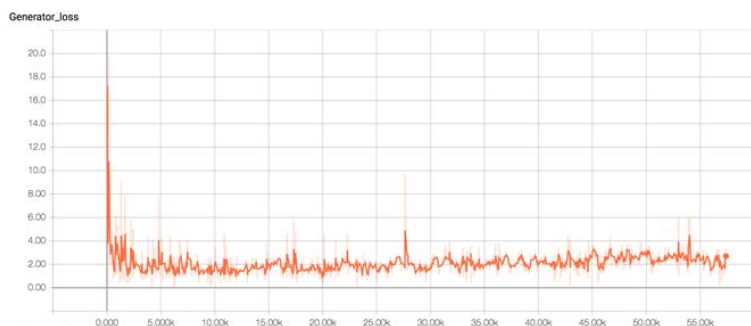
Epoch 100



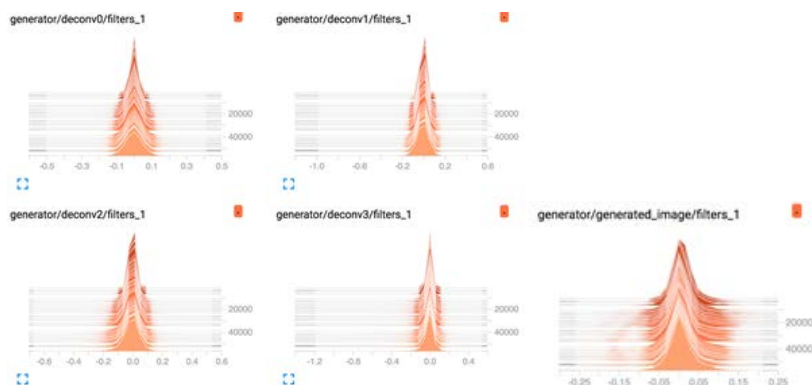
Discriminator loss



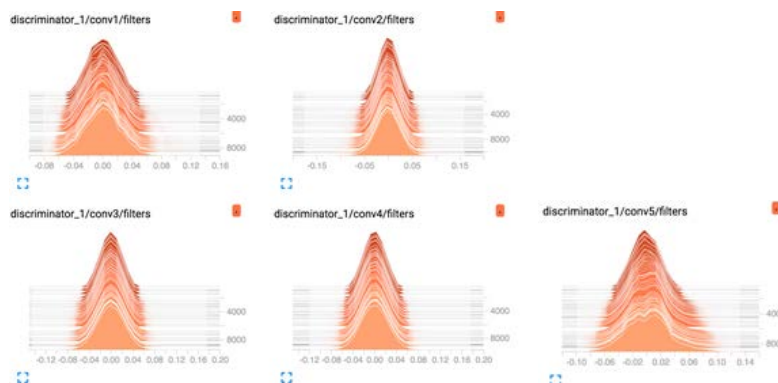
Generator loss:



Parameters distribution of generator filters:



Parameters distribution of discriminator filters:



從 Generator 產生的圖片結果來看，發現有沒有改 loss function 其實影響不大。（至少效果沒有變得比較差）。但 Discriminator 在訓練過程中會比較平滑；Generator 的 loss function 雖然不一樣了，loss 的值卻意外的沒有差太多。

Tip #14: Train discriminator more

Epoch 10



Epoch 20



Epoch 30



Epoch 40



Epoch 50



Epoch 60



Epoch 70



Epoch 80



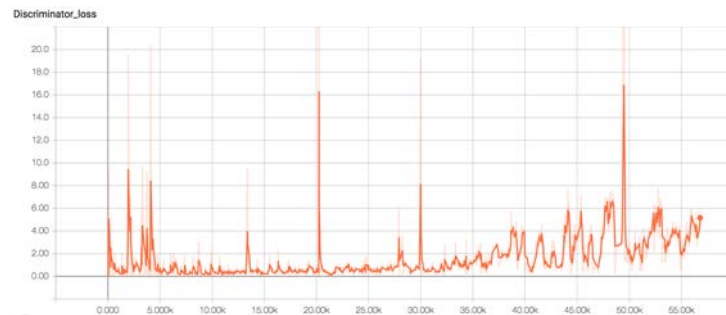
Epoch 90



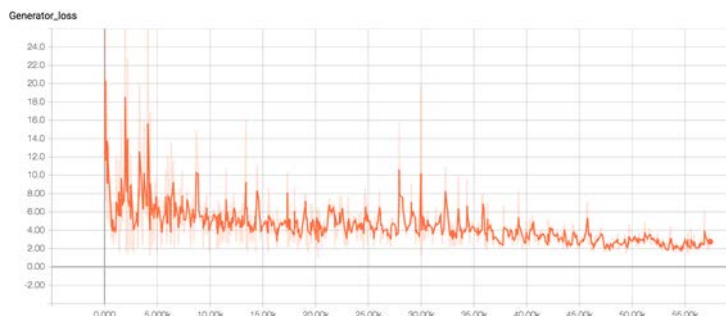
Epoch 100



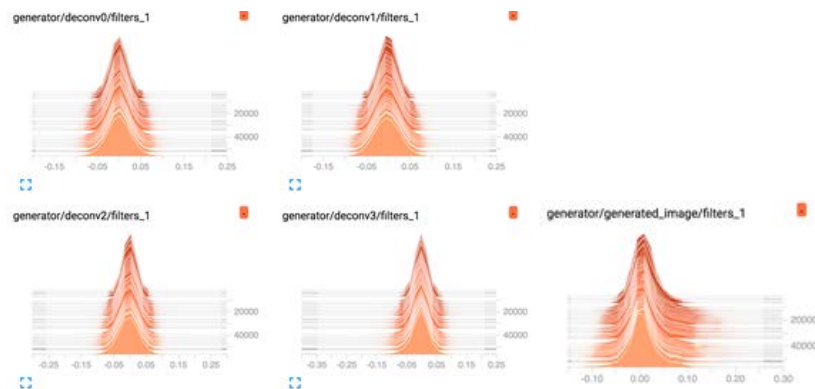
Discriminator loss:



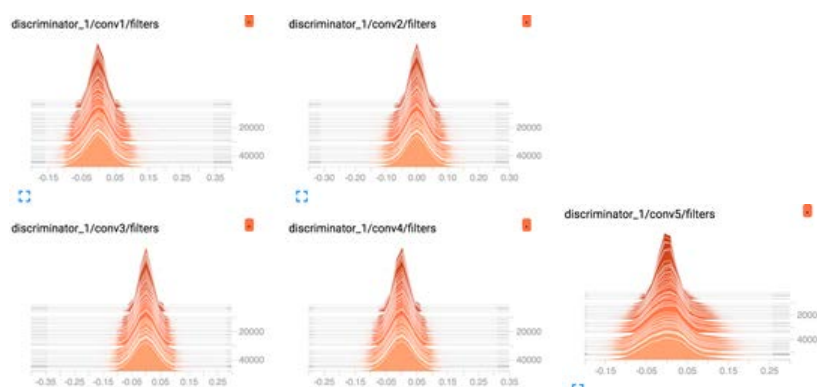
Generator loss:



Parameters distribution of generator filters:



Parameters distribution of discriminator filters:



從 Generator 產生的圖片來看，似乎 Discriminator 訓練比較多次沒有太明顯的進步，但至少沒有變差。而訓練過程中，Discriminator 的 loss 在訓練的末期，loss 有上升的現象，而 Generator 的 loss 長期來看，在訓練的末期仍然可以緩慢的下降。或許 Discriminator 訓練比較多次的意義在於 Generator 的 loss 可以比較順利往下降。

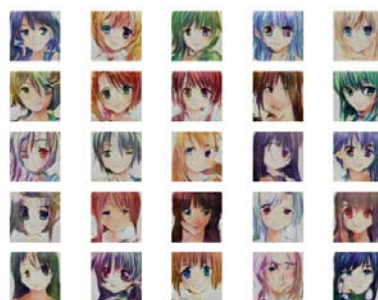
Tip #17: Use Dropouts in G in both train and test phase

本實驗中，在 Generator 的每層 deconvolution layer input 前加上 dropout layer，dropout rate = 0.5。訓練過程中產生的圖片如下：

Epoch 10



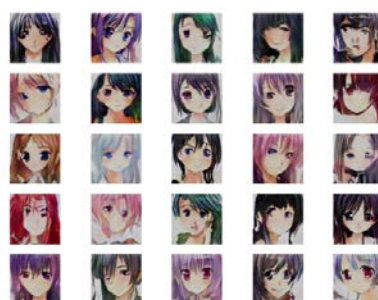
Epoch 20



Epoch 30



Epoch 40



Epoch 50



Epoch 60



Epoch 70



Epoch 80



Epoch 90

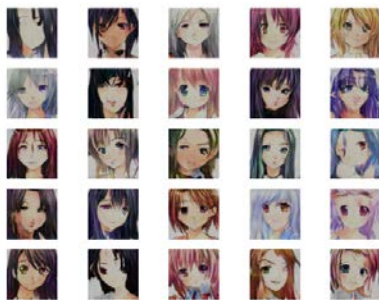


Epoch 100

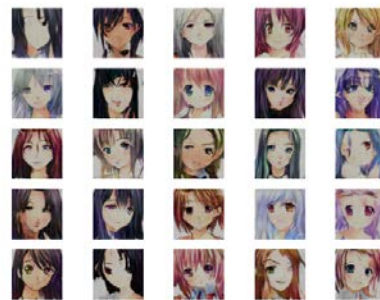


Dropout in Testing phase :

Dropout rate = 0.5

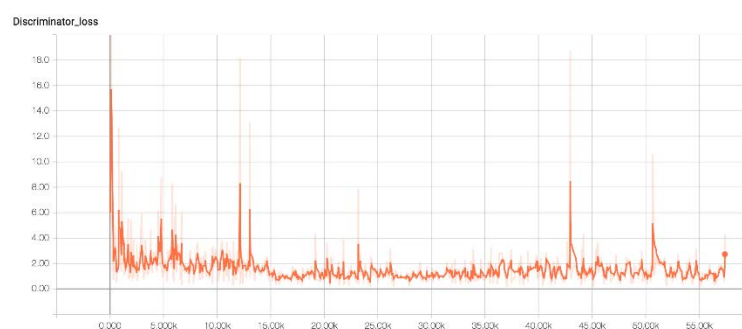


Dropout rate = 1.0

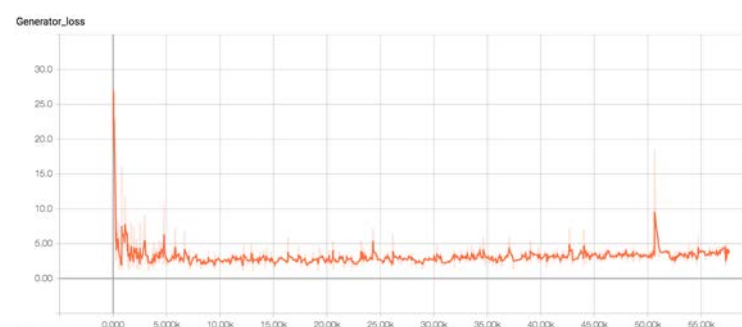


可以看出，testing phase 有沒有用 dropout 的差別並不顯著。

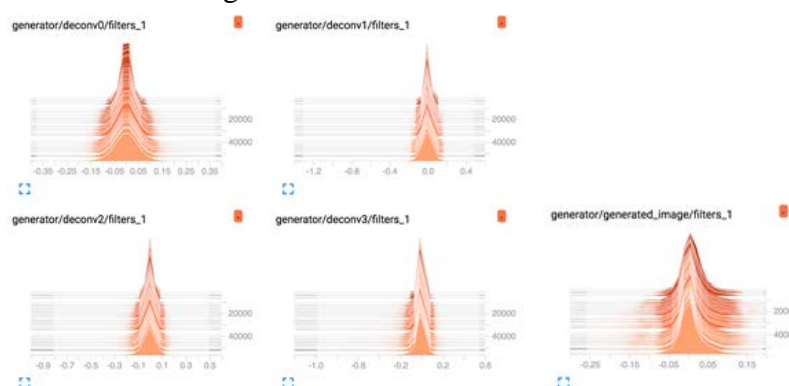
Discriminator loss:



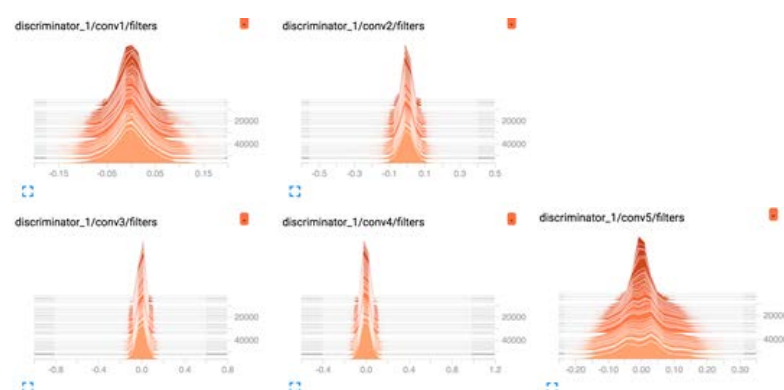
Generator loss:



Parameters distribution of generator filters:

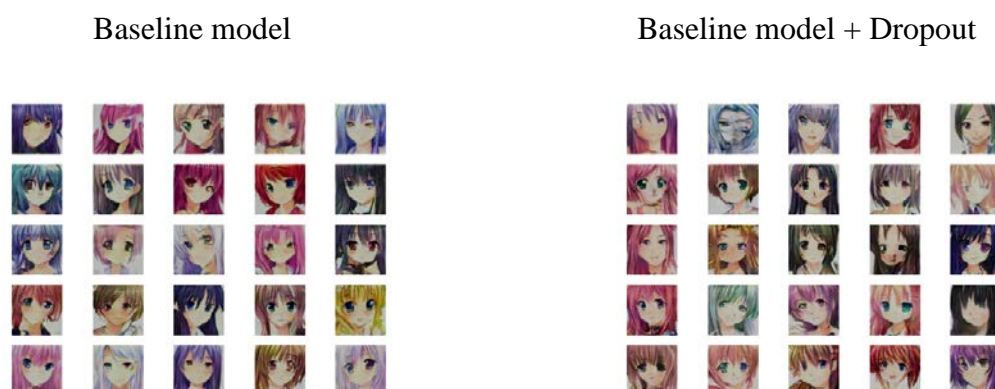


Parameters distribution of discriminator filters:



以圖片的品質來說我覺得在 Generator 加上 dropout 看起來會比較容易有崩壞的情形發生。以下圖為例，圖左和圖右分別為訓練了 100 epochs 的 model，但是圖左為沒有 dropout 的 model 產生的圖片，圖右則為有 dropout 的 model

產生的結果。雖然圖左的圖片有的五官比例看起來怪怪的，但整體上仍然是動漫人物的臉；有加 Dropout 的 model 卻比較容易產生人臉糊掉的照片，猜想或許是 Dropout 隨機的部分所造成的影響。

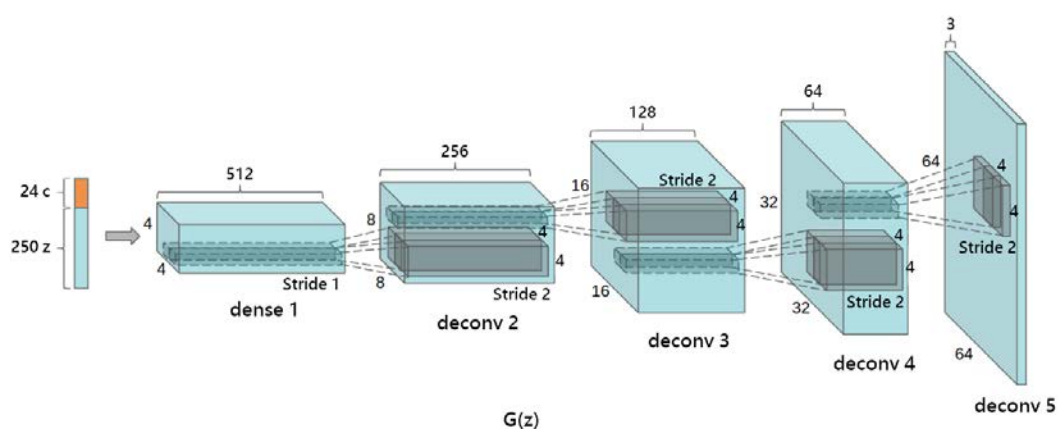


以上三個 training tips 大致上對於 Generator 產生的圖片並沒有太明顯的改進，當然這也不排除是本身的架構已經有了個很好的結果，所以這些技巧才看不出它應有的效果。

IV. HW 3-2 Text-to-Image Generator

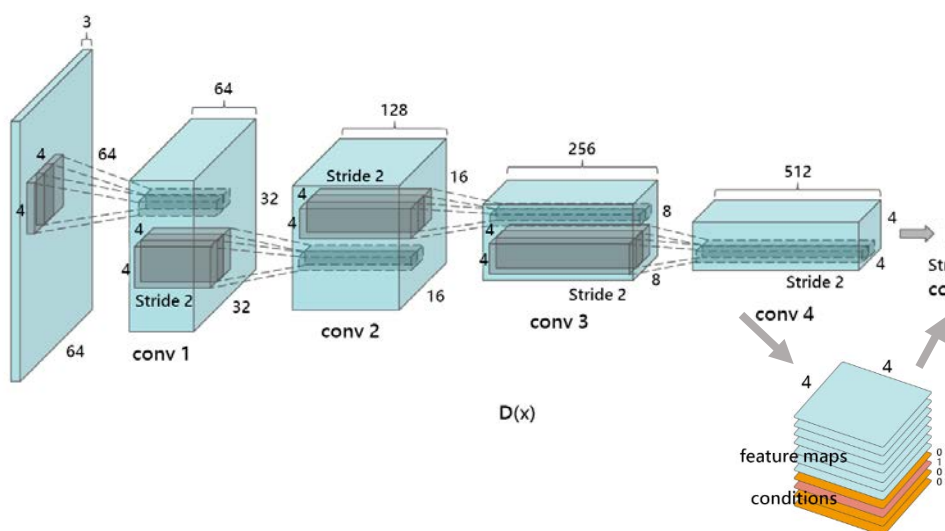
1. Model Description

Model 架構基本上是根據 3-1 的 DCGAN 修改而成的 conditional DCGAN。首先會依據 tag 裡的髮色及瞳色分別做 one hot embedding，得到一個 24 (12+12)維的 condition vector。再把 noise 跟 condition concatenate 在一起，當作 input。而與 HW3-1 最大的差別在於將第一層的 deconvolution layer 換成 fully-connected layer，再繼續接上之後 deconvolution layer。這邊可以將 fully-connected layer 視為 embedding layer，希望可以更好的抽取 condition 的資訊，得到更穩定且符合 condition 的圖片生成。



而 Discriminator 方面則是在最後一層 convolution layer 之前把 (512, 4, 4) 的 feature map 與 repeat 成 (24, 4, 4) 的 condition concatenate 在一起 (如

下圖所示)，再經最後一層 convolution layer predict 出 real/fake label。



2. Experiment settings and observation

實驗做在 data augmentation 上，而 anime data 只取 tag 檔中，髮色的 tag 以及瞳色的 tag 恰好只有一個的圖片，避免訓練過程中對 model 造成混淆。而實驗中訓練在兩種 data 組合上：

(1) extra data

(2) extra data + 水平翻轉

本次實驗中並不做一般的 rotate，因為根據 HW3-1 的實驗中可以發現 rotate 會讓圖片的四角產生黑邊，而 model 幾乎都會學到這個特徵，但並未顯著的增加 Generator 產生的圖片品質，甚至降低，故不使用。

實驗結果如下：

Dataset (1)

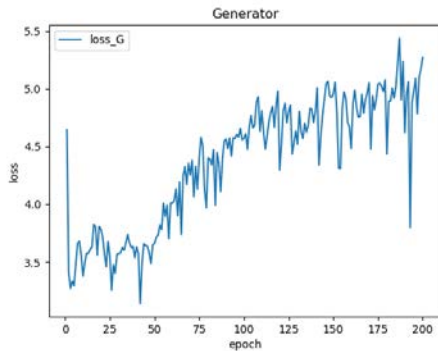


Dataset(2)

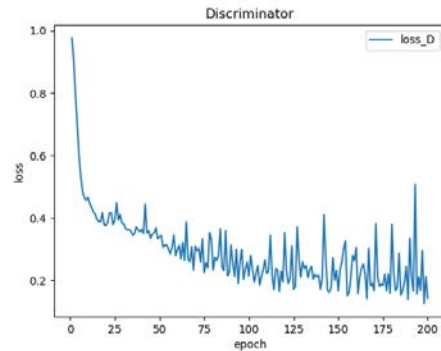


可以發現兩者的 label 都是明顯正確的，但是未水平翻轉 dataset (1)，眼睛的形狀明顯較扭曲，大小眼的問題嚴重，另外臉型方面也較多不對稱，可以觀察到做了水平翻轉的 dataset (2) 整以而言穩定度較高，產生的圖片都有一定的水準，但仍然會有些微扭曲產生。所以藉由水平翻轉得到 data augmentation 的效果，的確可以增進 Generator 的效能。

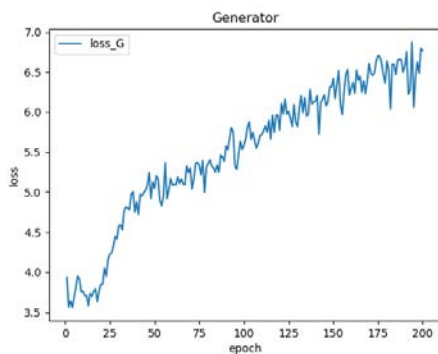
Dataset (1) Generator loss



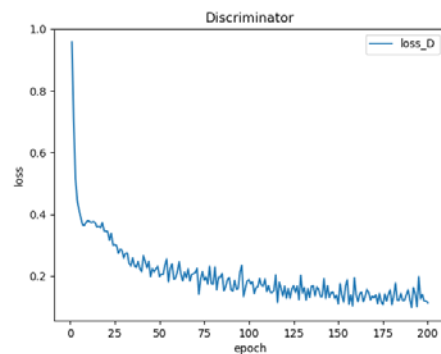
Dataset (1) Discriminator



Dataset (2) Generator loss







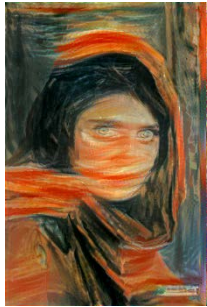


Dataset (2) Discriminator



很明顯的 Dataset (1) 的 Generator 在訓練過程中 loss 震盪較劇烈，而這也直接反映在圖片的生成方面，可以看到 Dataset (1)產生的結果有大小眼、臉型也大多不對稱，較不穩定，成果較差。而 Discriminator 與 Generator 互相影響，所以 Discriminator 在 Dataset (1) 震盪也較劇烈，而不穩定的 Discriminator 也會造成判斷上誤差大，與 Generator 互相影響，難以訓練出較好的

V. HW 3-3 Style Transform

1. Show the result

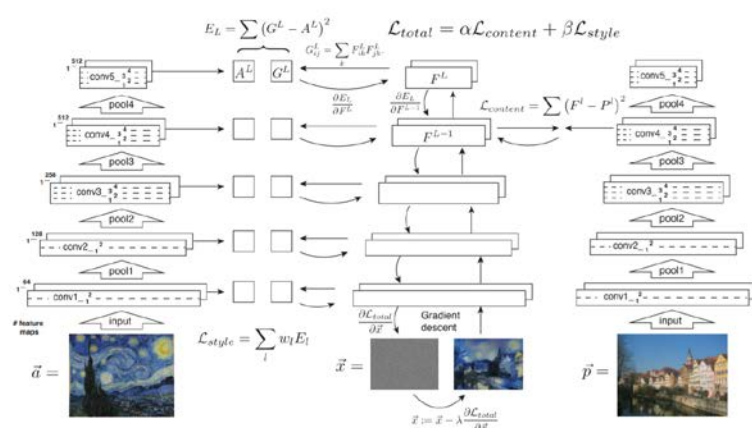
input	style	output
	 <i>The Starry Night</i> by Vincent van Gogh	
	 <i>Der Schrei</i> by Edvard Munch	
	 <i>Woman with a Hat</i> by Henri Matisse	

2. Analysis

模型架構：

這個 style transfer 的演算法使用了 VGG-verydeep-19 的 pre-trained network，此模型包含 16 層的 convolution layers（with 3*3 filter）和 5 層 pooling layer，深度達 19 層，通常用於 object recognition 和 localization，這裡我們利用它來抽取圖片的 features，進一步做到 style transfer 的效果。

此演算法的想法與架構如下：



我們利用上述模型抽取出三張圖片的 features，分別是欲改變風格的圖片(content)、欲轉換的風格(style)和產生的圖片(generated image)。

Generated image 一開始以全白的圖片初始化其值，訓練過程中，分別計算 generated image 的 feature 和 content feature、style feature 相差的 mean squared error，加權後當作 loss（此次實驗中，content feature: style feature=5: 1000），並使用 L-BFGS optimizer 優化圖片，generated image 的像素值隨著 epoch 逐漸被訓練成不同風格的圖片。

值得一提的是，我們只抽取較高層的 convolution layer 所抽取的 content features，因為做為一個用於 object recognition 的 CNN，較高層的 feature 比較能代表圖片的實際內容，而不只是表示各個像素的資訊，所以生成的圖片既能表達圖片的內容，又不拘泥於像素的資訊。

觀察結果：

此演算法生成出的結果非常好，效果轉換的很成功，甚至比一些 cycle gan 得到的結果更好，可以看出它的輪廓清晰，也因為使用的是用於 object recognition 的 pre-trained model，所以各個物體之間，像是臉、頭巾等，可以看出明顯的差異，而不會模糊成一塊。而且它 training 的時間和所需的 training data 遠遠低於 gan。

雖然這個演算法在畫風轉換上蠻成功的，甚至超越有些 gan 的效果，但是應用範圍遠小於 gan，無法在 zebra 和 horse 之間做轉換，也無法生出各式各樣的人臉表情。

VI. 分工表

hw3-1	B03901101 楊其昇
hw3-2	B03901145 郭恆成
hw3-3	B03901065 林宣竹