

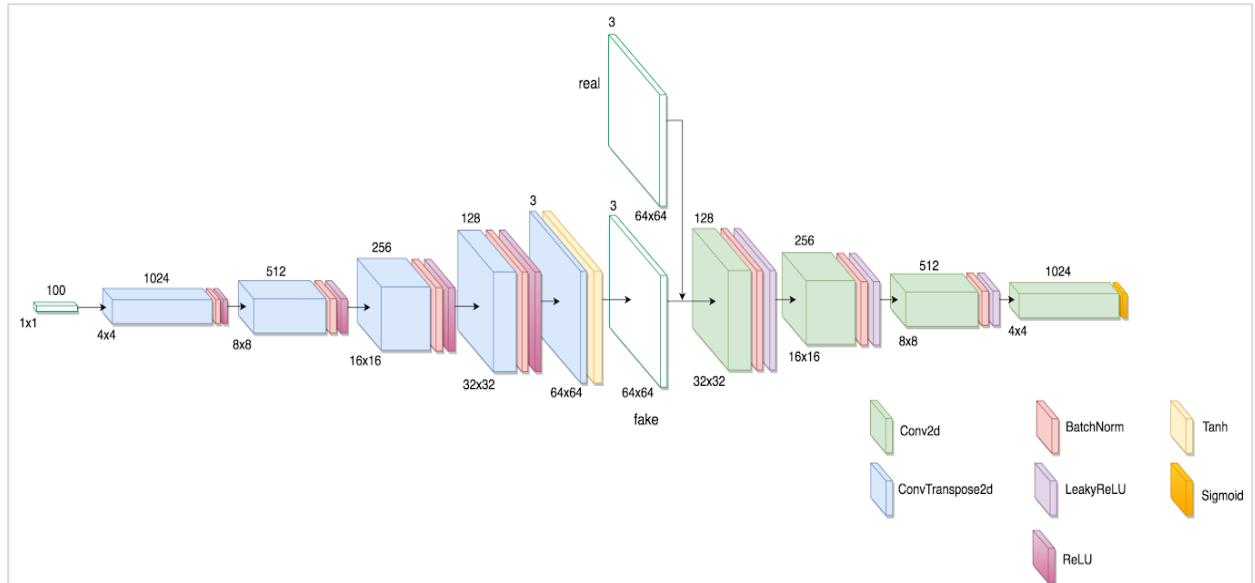
MLDS HW3 Report

組員：b04901060 電機三 黃文璿
b04901003 電機三 許傑盛
b04901096 電機三 蔡昕宇

分工：code: WGAN-GP, report: 1, 3
code: CGAN, data report: 2
code: DCGAN, report: 1, 4

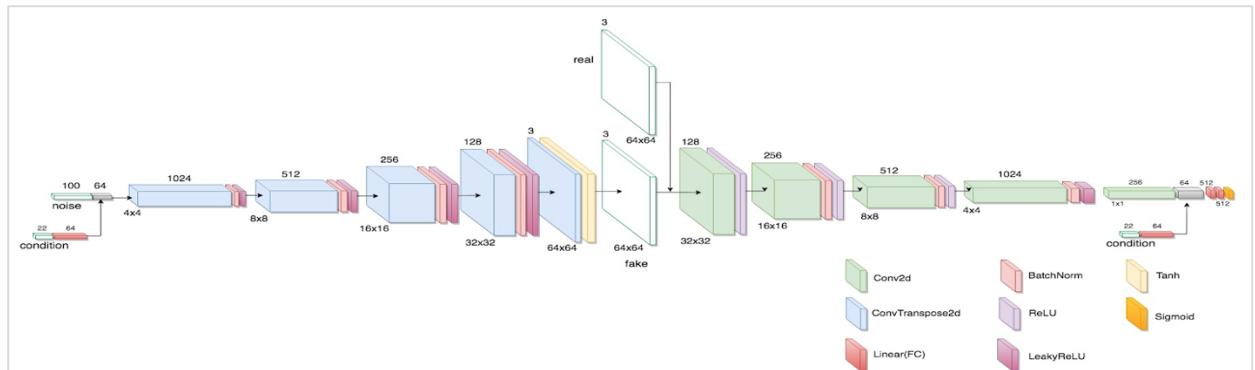
1. Model Description

1.1. Image Generation



這次用於 image generation 的模型和原本 DCGAN 論文中的設定相同，如上圖，其中我們也使用了論文中提到的訓練方法：loss 為 binary cross entropy，在 Generator 和 Discriminator 中都使用 Batch Normalization，optimizer 為 Adam，其中 Adam 的參數為：
 $\text{lr} = 0.0002$ ， $\text{beta1} = 0.5$ ， $\text{beta2} = 0.999$ 。Batch size 為 128，input noise 的維度為 100。另外 Generator 和 Discriminator 訓練的次數相同為 1:1，而不像 WGAN-GP 中 G:D 的訓練次數為 1:5。

1.2. Text-to-Image Generation

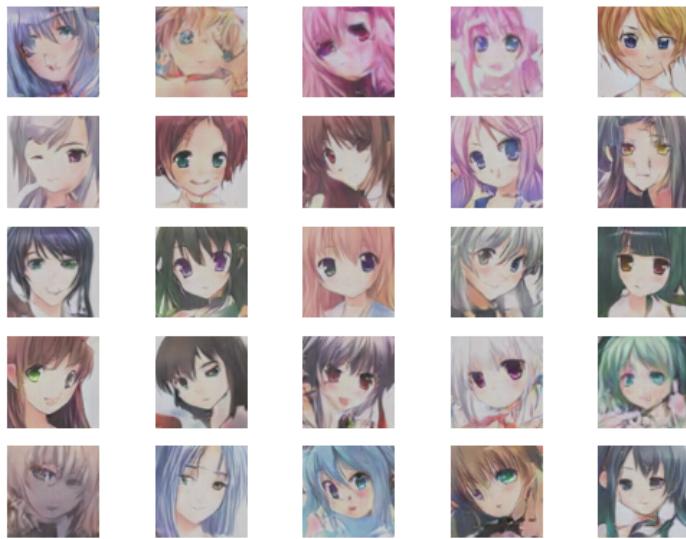


本題的 Conditional GAN 實作中先將 tag 轉換為 22 維的 one-hot 向量，再將這個向量通過 Fully-connected 進行 embedding，這個部分我們會在後文中討論 embedding layer 的效果。在 Generator 中 embedding 後的 tag 和 noise concatenate 在一起，而在 Discriminator 中 tag embedding 和最後 FC 的 feature concatenate 在一起，再通過兩層 Fully-connected，最後同樣為 Sigmoid。各層的參數可參考上圖，和 DCGAN 大致相同。在 CGAN 中我們使用的 loss 和 DCGAN 相同為 binary cross entropy，optimizer 的部分也相同，不過我們還另外使用了一些技巧，如在 D 的 input image 加上 noise，隨著 epoch decay。

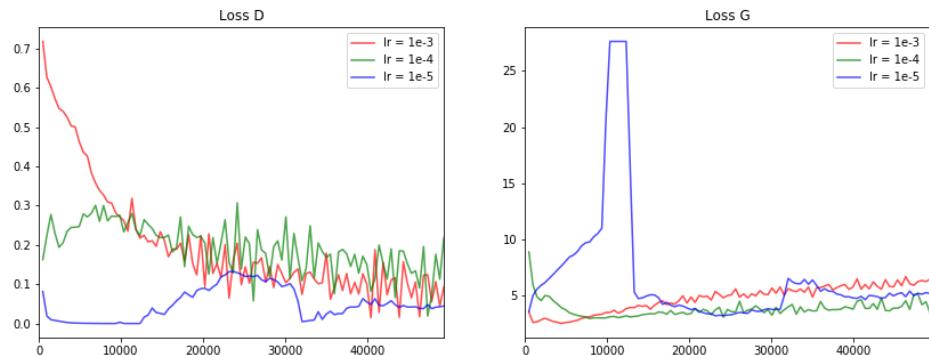
2. Experiment settings and observation

2.1. Image Generation

本次 DCGAN 所生成的圖片如下：



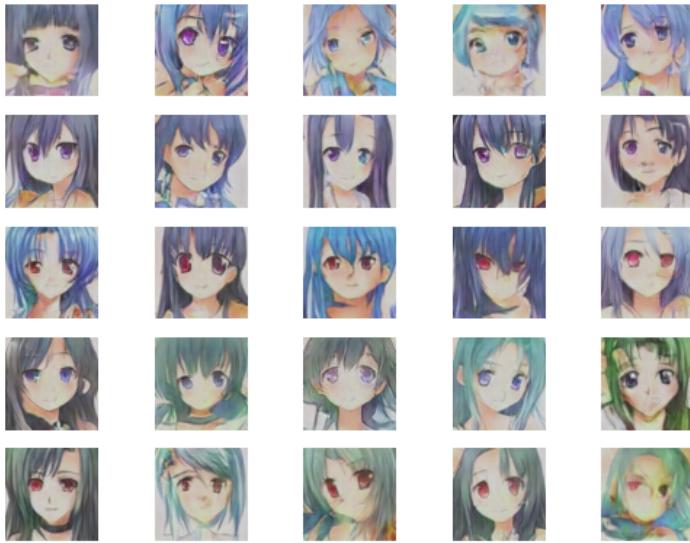
在 GAN 的訓練過程中，每次的 **training** 都非常震盪、比較沒有一個收斂的情況，因此我們想觀察 **optimizer** 在不同情況下的對訓練的影響。由於在 tips 中已經有一項使用 Adam 作為 **optimizer**，因此我們特別觀察 Adam 在不同 learning rate 底下，對 DCGAN 的訓練所造成的影响。下圖是我們分別使用 $\text{learning rate} = 1\text{e-}3, 1\text{e-}4, 1\text{e-}5$ 的 training 過程。



此外我們也嘗試過 $\text{learning rate} = 1\text{e-}2$ 做實驗，不過該情況會發生 Generator 的 loss 一直是 0，完全無法訓練到 Generator，產生的圖片也只是全部灰色的馬賽克。由上圖的曲線我們可以推論以下幾點，第一是 GAN 的 error surface 相當的崎嶇，當 learning rate = $1\text{e-}5$ 時，我們甚至能看到 loss g 有跑到一個 loss 在不同數量級的地區，這樣的情況可能可以解釋到為什麼 GAN 的訓練會是相當不穩定的。第二點，我們可以發現到在 $\text{lr} = 1\text{e-}4, 1\text{e-}3$ 的訓練前期，loss g 處於一個相對平穩的狀態，另外由 $\text{lr} = 1\text{e-}2$ 時 loss g 幾乎 = 0 的情況，我們可以推論，對 Generator 來說，產生出讓 Discriminator 能夠輕鬆分辨真假的圖片的 loss 是相當平穩的，而到後面 Generator 要產生讓 Discriminator 難以分辨圖片真假時，loss 會處於一個相當崎嶇的地區。另外觀察 loss d 與 loss g 的相對關係，我們能夠發現互補的關係，也就是說當 loss d 有顯著下降時，loss g 會有明顯的提昇，這樣的情況闡釋了在功能上 Generator 與 Discriminator 的相對關係。

2.2. Text-to-Image Generation

本次 CGAN 所生成的圖片如下：



由上到下每個橫排分別為：

(hair, eyes): (blue, blue), (blue, green), (blue, red), (green, blue), (green, red)

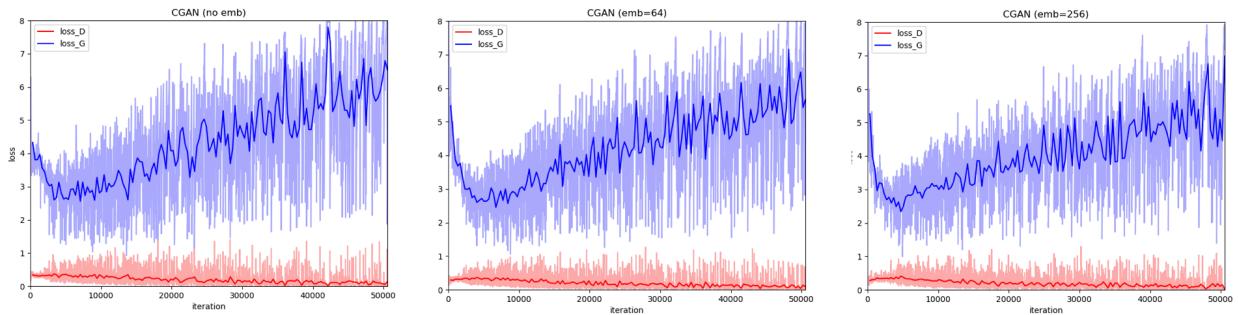
在 CGAN 中最重要的 feature 就是輸入 model 的 tags，最基本的作法是在原本的 noise 後面再接上經過 one hot encoding 過的 tags。不過這樣的作法會讓 generator 的 CNN 除了生成圖片之外，還需要具備做 tags embedding 的能力以理解 tags 的代表意義，而這點比較不是 CNN 所擅長，而是 FC。因此我們讓 one hot encoding 的 tags 先經過 FC 的 embedding 再接回 noise，如此一來可能會有較好的 performance，我們將實驗做在不同的 embedding 維度，討論不同維度下，對 CGAN 的訓練所造成的影響。



以上圖片是由 `testing_tags.txt` 所產生的，由上圖我們可以發現到當 `embedding` 的維度越大時，模型可以在很早的時候便學習到圖片對應 `tag` 的關係。

可以明顯的觀察到，256 維的 `embedding` 在第十個 epoch 就可以學習到 `tag` 的資訊（前三橫排的 `tag` 為 blue hair，後兩排為 green hair），而 64 維到第 50 個 epoch 也開始有些效果，若我們直接將 `embedding` 移除，則可以發現到第 150 個 epoch 模型仍然可以學習到 `tag` 資訊，但在 50 epoch 以前則完全看不出來。綜合以上結果可以得知較高的 `embedding` 維度有機會加快 CGAN 的收斂。

這樣的情況在觀察 `training` 時的 `loss` 情況或許能得到一點解答，我們發現在維度比較高的 `embedding` 會在 `training` 初期到達一個相對較低的 `loss`。下圖由左至右分別是 `no_emb`, `emb_64`, `emb_256`。



3. Compare your model with WGAN-GP

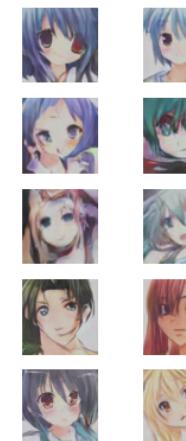
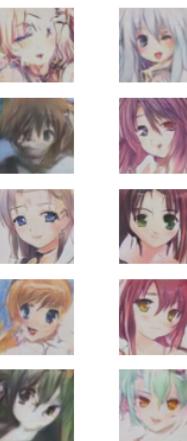
3.1. Model Description of the chosen model

在本題中我們使用 WGAN-GP 和 DCGAN 進行比較。WGAN-GP 的架構基本上和 DCGAN 相同，只不過最後的 sigmoid layer 被移除，以及 D 中沒有 Batch Normalization。最大的改變是在於 loss function 的設計。意思上 WGAN-GP 依然是要使 D 將 real 和 fake image 分越開越好，G 則是試圖根據讓輸出更接近 real image。不過 WGAN-GP 中不像 DCGAN 限制 real 為 1，fake 為 0，而是加上 gradient penalty 來避免發散。WGAN-GP 的 loss 如下：

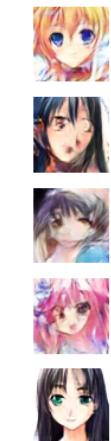
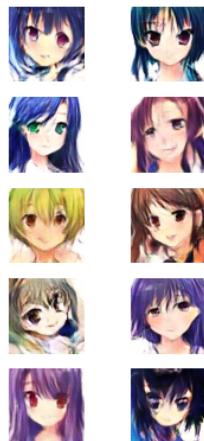
$$L = \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g} [D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

3.2. Result of the model

DCGAN



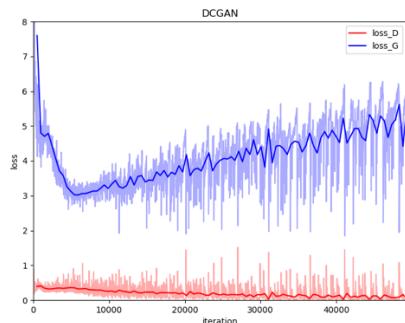
WGAN-GP



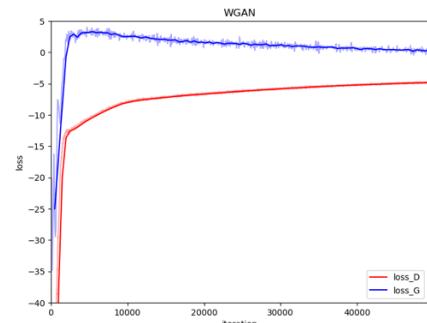
3.3. Comparison

首先我們從兩個 model 訓練過程的 loss 中觀察出明顯的不同穩定性：

DCGAN



WGAN-GP



可以注意到 DCGAN 的訓練過程中，iteration loss (淺紅、淺藍) 的變動非常劇烈，但在 WGAN-GP 中則是相對非常穩定，類似 WGAN-GP 原文中宣稱更穩定的訓練過程。

另外觀察的是生成的品質：

DCGAN



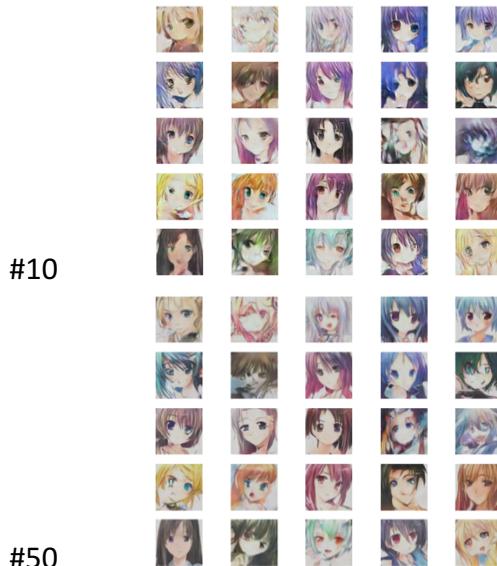
WGAN-GP



可以觀察到 DCGAN 所生成的圖看起來較灰，也就是對比度或飽和度比較低，而 WGAN-GP 則很明顯的有更亮更鮮艷的圖片，雖然從正確性來說，DCGAN 和 WGAN-GP 都還會生成出一些較不真實的臉，但整體來說品質相對接近。

接著比較的是 DCGAN 和 WGAN-GP 的收斂速度。前面的圖是在 100 epoch 的比較，故接下來我們分別在 10、50 epoch 觀察兩個模型的生成結果，如下：

DCGAN



WGAN-GP



從 Epoch 10 的結果來看，DCGAN 中已經有一些品質不錯的圖片生成，但 WGAN-GP 的圖大部分還是有許多雜訊，到了 Epoch 50 時 WGAN-GP 的品質有逐漸變好，但仍然有不少扭曲的圖片，故可以猜測 WGAN-GP 的訓練過程需要更多時間。

4. Training tips for improvement

4.1. Pick three tips, which tip & implement details (3% each 1%)

在本題中，我們實作中的三個 training tips，分別為

1. modified loss function (tip 1)
2. batch normalization (tip 4)
3. dropouts in G (tip 17)

而我們在前者的 model 中，有使用 batch normalization，因此在這裡比較是否使用 batch normalization 的結果作為比較。其他的在之前的 model 皆沒有使用，因此在這部分的實驗加上這幾個 tips 與之前的 model 作為比較。

而基本的 implementation 設定與之前的 model 相同

optimizer: Adam(lr= 0.0002, betas=(0.5, 0.999))

Discriminator : Generator update ratio = 1 : 1

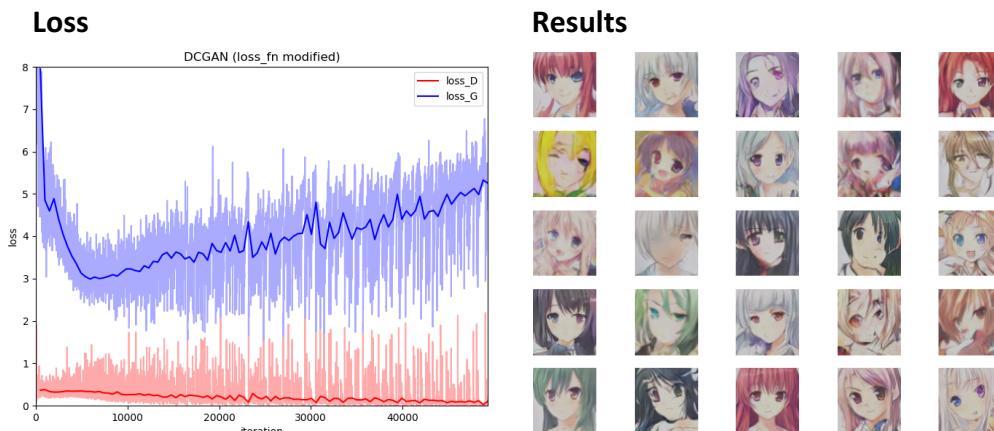
ReLU in Generator, LeakyReLU in Discriminator

根據上述這樣的架構加入以下要討論的 tips。

4.2. Result (image or loss...etc.) and Analysis (3% each 1%)

4.2.1. Modified loss function

在原始的 paper 中得到 generator 的 loss function 為 $\min \log(1 - D)$ ，但實作時卻用 $\max \log D$ ，在這個 tip 中比較兩者差別。

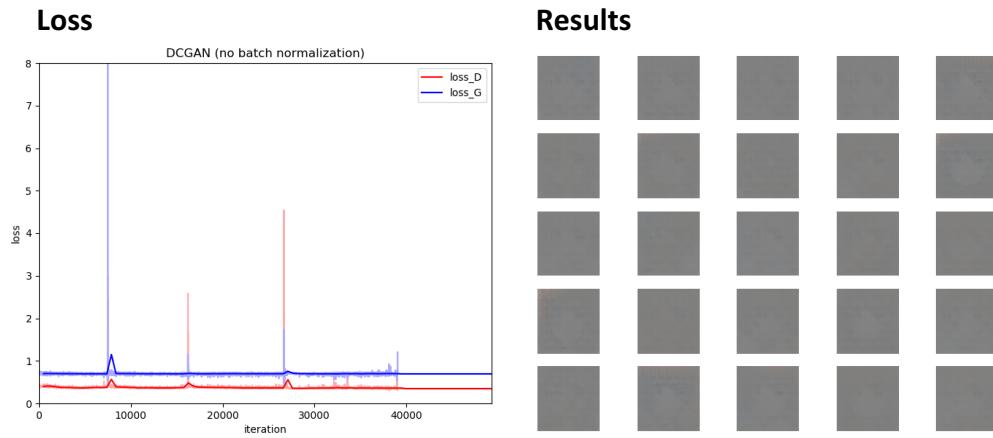


可以看到 loss 相較起來變化沒有那麼大，趨勢也如同前述的 model 的 loss 一樣，在大概幾個 epochs 之後 generator 的 loss 都會到一個最低點的曲線。再者觀察生成的結果如右圖，還是能生成不錯的圖片。基本上本次實驗中 implement 在兩個 objective function 上的結果其實差不多。

而從這次實驗的學習速度來看，使用這種 loss 並沒有很明顯的加速效果。觀察兩種訓練方式在前幾 epoch 的表現很接近，故較難給出哪個 loss 比較好的結論。

4.2.2. Batch normalization

因為在先前使用的 model 有 batch normalization，所以在這裡使用沒有 batch normalization 的狀況。



在這個情況之下，我們將原本 model 中在 Generator 與 Discriminator 的 batch normalization 移除，得到以上這樣的結果。Training loss 顯然沒那麼高，維持在一個定值，但我們同時也看到生成的圖片都是灰色的，皆生成雜訊。從圖中很明顯的看到 Generator 只會生成灰色的圖片，同時 Discriminator 並沒有分別出這兩者的差異，原因可能是因為 Discriminator 在沒有 batch normalization 的時候收斂的速度相對不夠快，不足以分辨 Generator 生成的圖片，才會導致 Generator 生成不出像是真實的影像。常常看到 Generator 的 loss 變為 0，更能確定是 Discriminator 沒有分辨出來的結果。

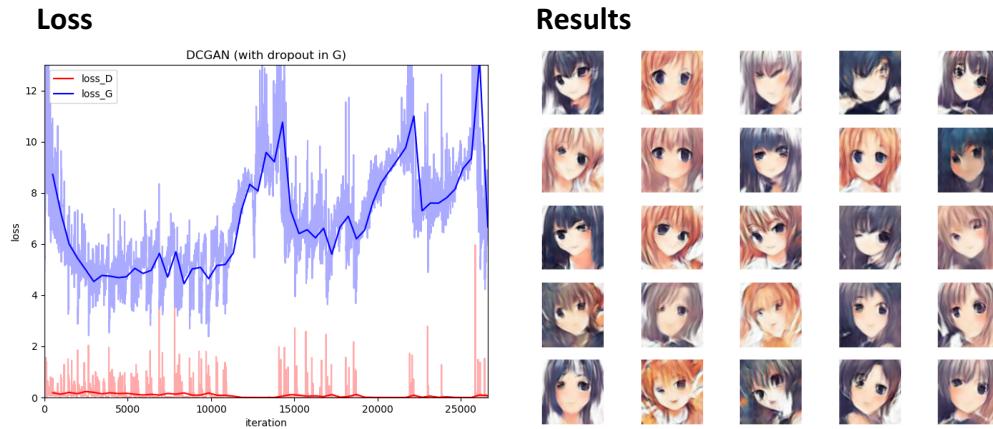
我們嘗試觀察 discriminator update 的過程，其中在前幾個 epoch 中，其實還仍然略有影像的生成如下圖。但是過了幾個 epoch 之後，人臉開始慢慢消失，又變回全部灰色的圖片。推測加上 batch normalization 的效果明顯較好。

下圖為第 4 個 epoch 所生成的圖，可以發現有一點臉部的輪廓：



4.2.3. Dropouts in G

前面的實作沒有加上 Dropout，以下為加上後的結果：



在 tip 中提到，利用 dropout 來製造 noise，用來減少 model 在 training 的過程中容易 memorize training data。但在 GAN 之中，加入 dropout 的效益感覺不大，因為本身通過 generator 的 input 也是一個 noise，model 在 training 上 dropout 有加與否似乎沒有那麼大的差異。由左圖來看，loss 似乎較比較不穩定，尤其在過了 20 個 epochs 的時後，看到 loss 在幾個區段會突然飆升，training 過程中其實沒有比較穩定，且當這個現象發生時，常常在之後出現 mode collapse 的情況。右圖為第 16 epoch 生成的圖，看起來生成的結果較好，然而，越到後面是 mode collapse 發生時所生成的圖，如下圖：



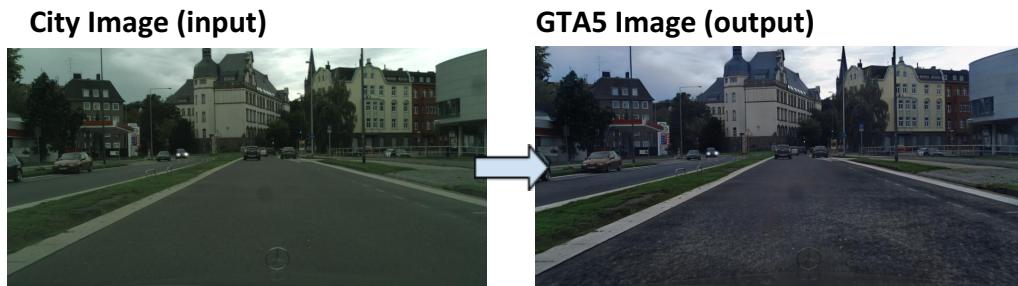
可以看到所生成的的圖片在 50 epoch 的時後其實就很相像，大概被分成黃色頭髮與綠色頭髮，但越到後面全部都變得一樣，結果不是很理想。因此我們在後來的實作上並沒有加上 dropout，反而效果較好。

5. Bonus

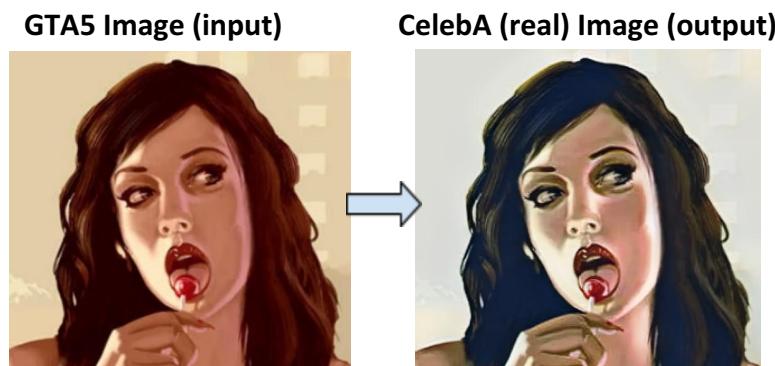
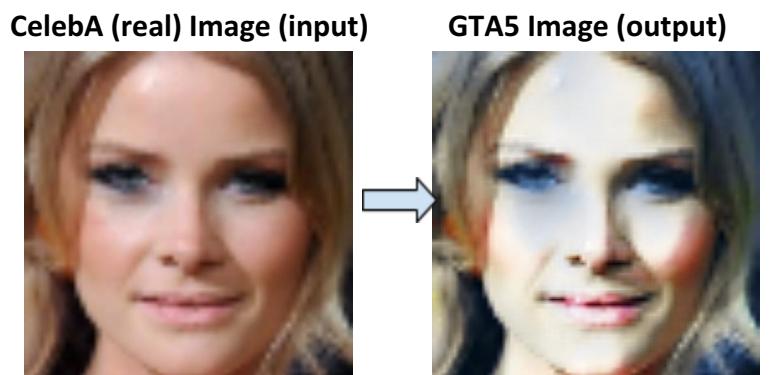
5.1. Show your results (1%)

我們使用 Unsupervised image-to-image transform network (UNIT) 來實作本題的 style transfer。而在網路上有原作者在 github 上基於 proposed 的 paper 已經有 pretrained 好的 model，以下的討論皆使用這個 pretrained model 來做觀察。

在這個 data 之中，將 real world 的街景與 GTA5 的虛擬場景作轉換。作者的 github 中還有很多例子，像是貓和狗的種類 transfer。而我們將這樣的 model 應用在 CelebA 的 dataset，把真實的人臉當作 real world 的 input，以下為 implement 結果。

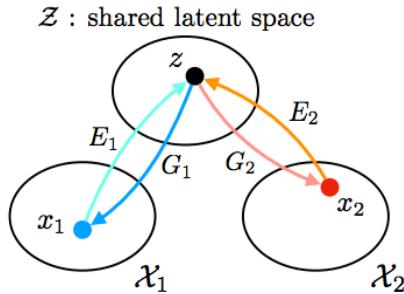


作者的 github 中還有很多例子，像是貓和狗的種類 transfer。而我們將這樣的 model 應用在 CelebA 的 dataset，把真實的人臉當作 real world 的 input，以下為 implement 結果。



5.2. Analysis (1%)

在原始的 paper 中的利用類似 VAE-GAN 的架構，且在他們的架構中，Encoder 所生成 latent 是 shared 的，亦即可能 input 同時可能是從兩個完全不同的 distribution，但是兩者皆壓到同一個 latent space 的假設，如下圖



因此將這個 latent space 給 generator 時，可以轉回原本的 distribution 或者是轉到對應的另一個 distribution 中。

從上面的 GTA5 的虛擬場景與現實景的轉換，可以看到 GTA 的場境看起來都會比較明亮。轉過去之後，場景裡面的物體相對看起來比較平面，較遠的物體看起來有點會有比較扁平的感覺，使得長得比較像虛擬環境裡面會出現的場景。

在人臉的部分，真實的 data 我們使用 CelebA 的 random data。但是 GTA 的 data 我們則是隨機網路上找的影像。從上面的轉換來看，發現其實這樣的 model 都在改變影像的光影，因為通常虛擬環境的物體因為光影不夠真實，才會肉眼看時感覺比較像假的。在 CelebA 的 data 轉過去之後，主體（人臉）轉為比較白的色調，就如同上面 city image 的天空的部分轉換也是轉換較白的色調。而我們 GTA image 做的反而較為不理想，因為可能原始的圖片比較平面，且更能確定這個 model 只有轉換色調而已。