

Deep Learning (Homework 3)

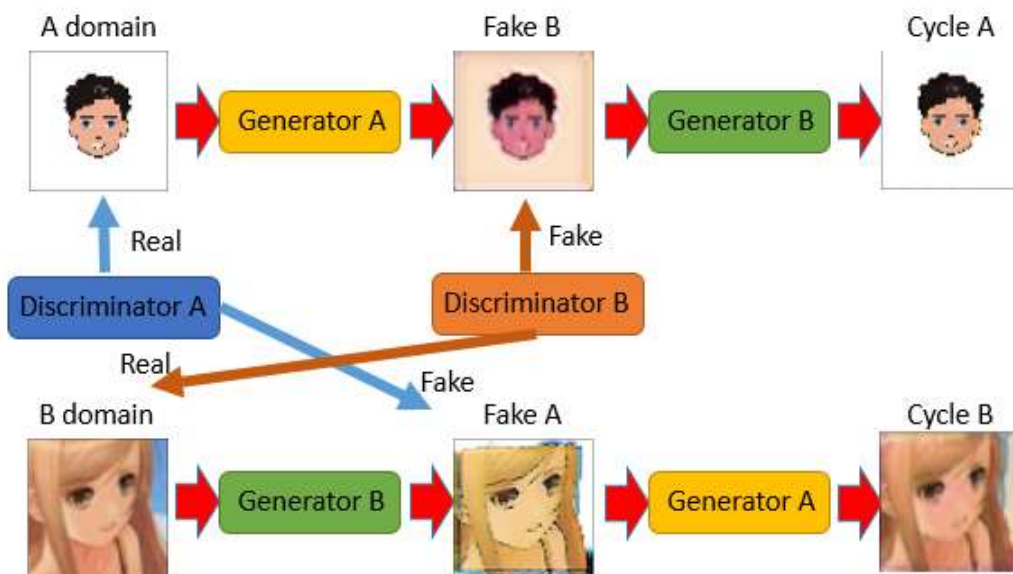
Prob2

這次的圖片資料有動畫(animation)與卡通(cartoon)，目標是要讓動畫圖片經過轉換後可以擁有卡通的風格，而卡通圖片經過轉換後可以擁有動畫的風格，為了達成此目的，我們使用了 Cycle GAN。

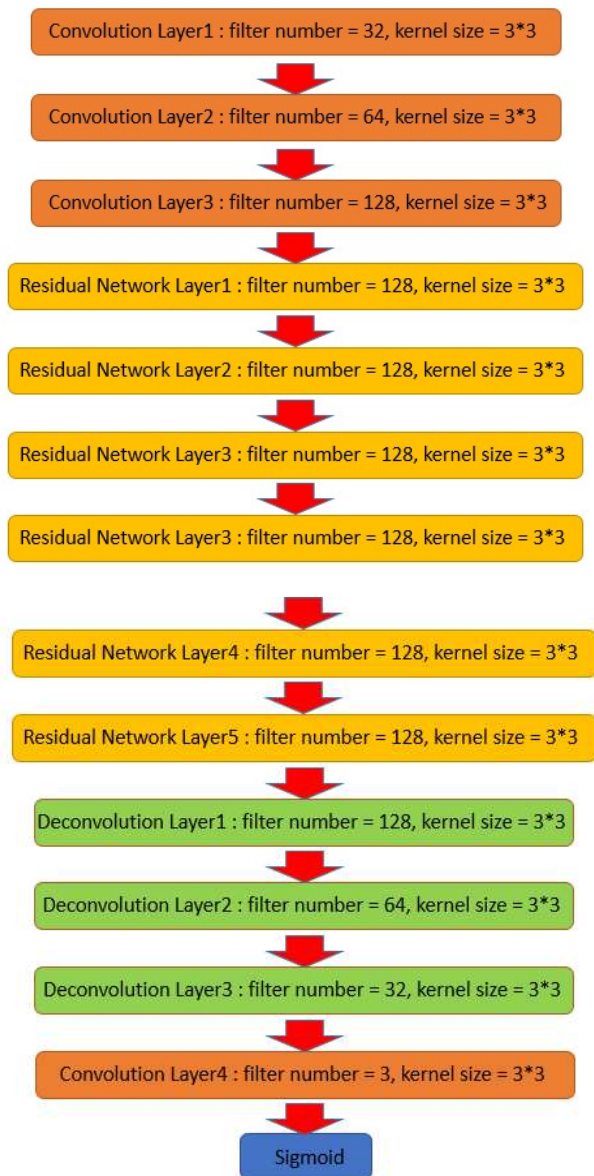
令卡通的 data 是屬於 A domain 的圖片、動畫的 data 是屬於 B domain 的圖片，我要建立 2 個 Generator 與 2 個 Discriminator，A domain 的圖片經過 Generator A 可以轉換成接近 B domain 的圖片，但其實這是張假的 B domain 圖片，轉換後的圖片會經過 Discriminator B 來判斷這張圖片的真假，我的 Generator A 會希望自己製作出來的 Fake B 圖片很像真的 B domain，能夠讓 Discriminator B 判斷成它是真的 B domain(輸出接近 1)，然而我的 Discriminator B 並不希望被 Generator A 產生的圖片給欺騙，所以希望自己能判斷出這是張假的 B domain 的圖片(輸出接近 0)，與此同時，我不會希望 Generator A 為了能夠騙過 Discriminator B，而不擇手段，創造了一張完全跟原本圖片(A domain 的圖片)毫不相干的圖，所以我會再把 Fake B 圖片通過 Generator B 轉回 A domain，稱它為 cycle A。我會希望我的 Generator 能能夠還原回原本圖片的樣子，所以要設法讓 cycle A 與原圖片(A domain 的圖片)越接近越好。

而動畫的 data 為 B domain，同樣的，B domain 的圖片經過 Generator B 可以轉換成接近 A domain 的圖片，但其實這是張假的 A domain 圖片，轉換後的圖片會經過 Discriminator A 來判斷這張圖片的真假，我的 Generator B 會希望自己製作出來的 Fake A 圖片很像真的 A domain，能夠讓 Discriminator A 判斷成它是真的 A domain(輸出接近 1)，然而我的 Discriminator A 並不希望被 Generator B 產生的圖片給欺騙，所以希望自己能判斷出這是張假的 A domain 的圖片(輸出接近 0)，與此同時，我不會希望 Generator B 為了能夠騙過 Discriminator A，而不擇手段，創造了一張完全跟原本圖片(B domain 的圖片)毫不相干的圖，所以我會再把 Fake A 圖片通過 Generator A 轉回 B domain，稱它為 cycle B。我會希望我的 Generator 能能夠還原回原本圖片的樣子，所以要設法讓 cycle B 與原圖片(B domain 的圖片)越接近越好。

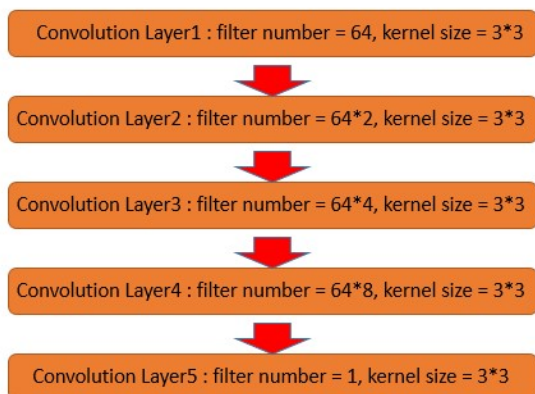
以上為 Cycle GAN 架構與 Loss function 決定的方式，下圖為此次的網路架構：



以下為我的 Generator 架構：



以下為我的 Discriminator 架構：



我設置了存放 fake data 的 pool，分別存取 fake A 與 fake B 提各 100 筆，供 Discriminator 做訓練，如果已經產生了超過 100 筆的 fake data，我會隨機挑出幾筆舊的 fake data 丟掉，然後用新的 fake data 取代，這樣可以讓辨識難度更高，同時限制 pool 大小，避免佔用過多空間。

以下為此次我 Training 的參數：

Learning rate= $4.999999999999996 \times 10^{-6} \rightarrow 7.737125245533638 \times 10^{-9}$

Epoch times= 30

Batch size=5

Data number=10000 (cartoon and animation both have 10000 images)

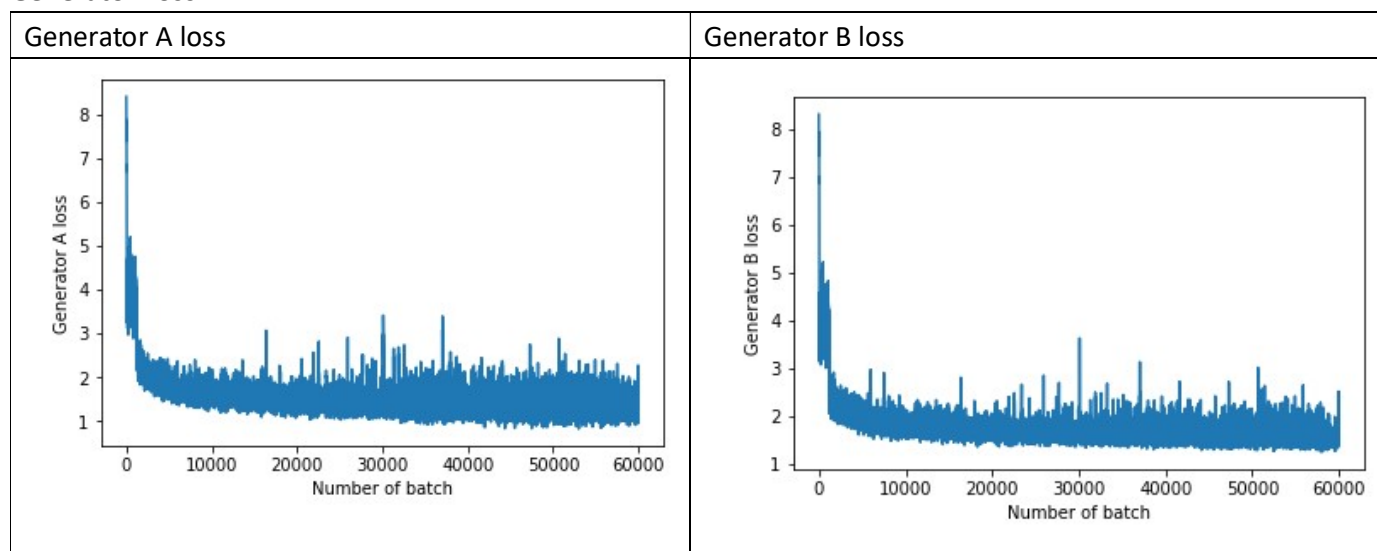
Result:

Generator A loss = 1.5438408, Generator B loss = 1.8068442

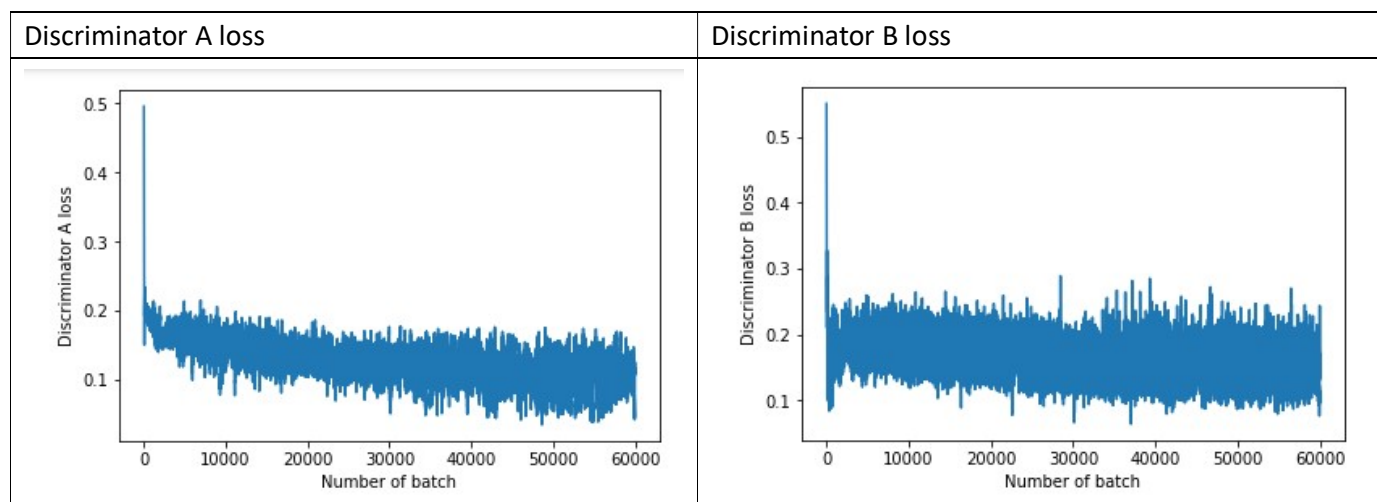
Discriminator A loss = 0.11698594, Discriminator B loss = 0.16998392

Learning curve per batch 如下：

Generator Loss

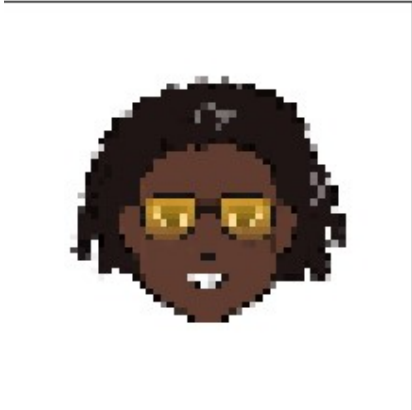










Discriminator Loss



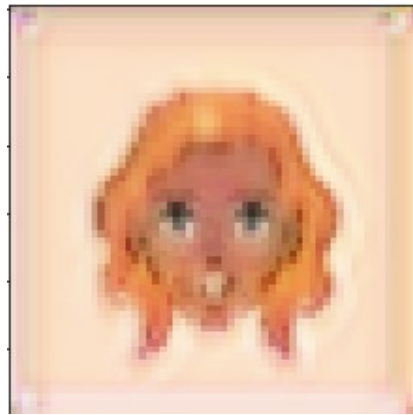
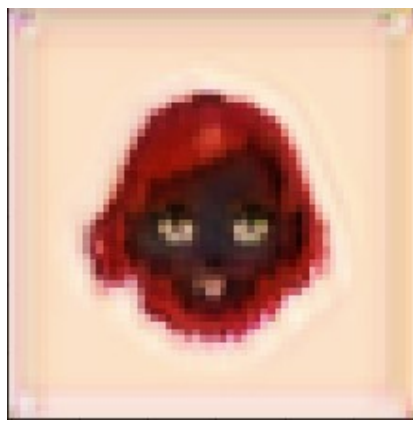
我分別丟入 15 筆來自 cartoon 與 animation 的 data 做測試，以下為結果：

丟入 cartoon 測試：










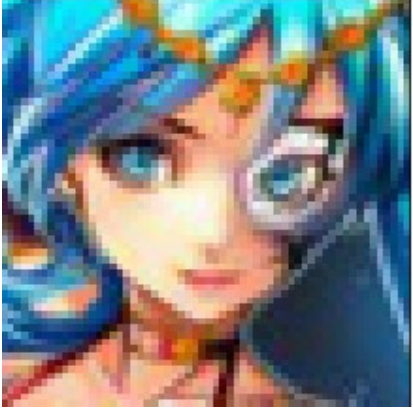

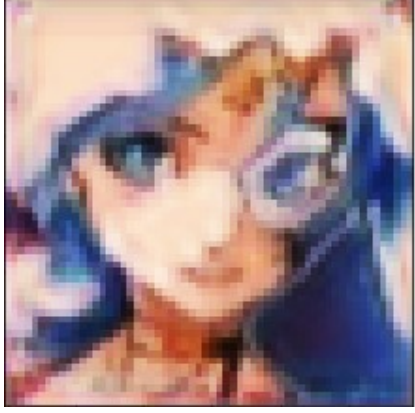
原 Cartoon 圖	經 Generator A 變 Animation	再經 Generator B 變回 Cartoon
		
		
		

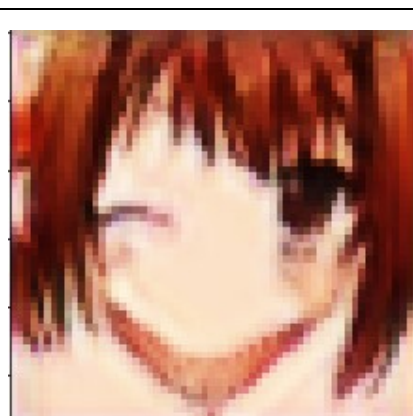
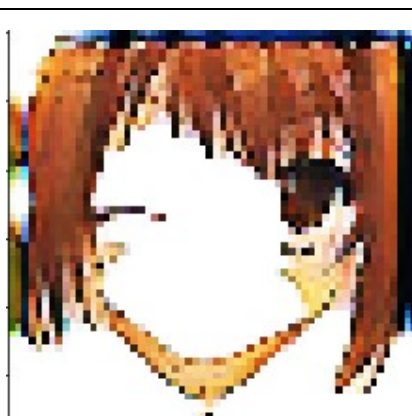
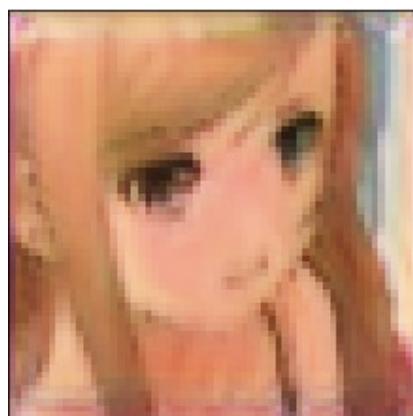


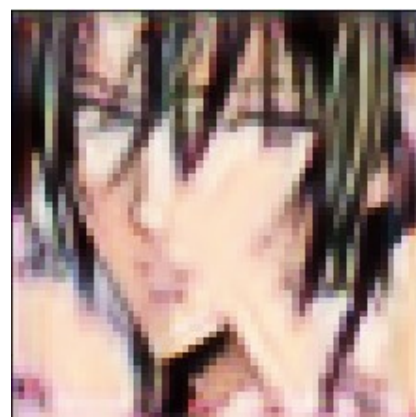


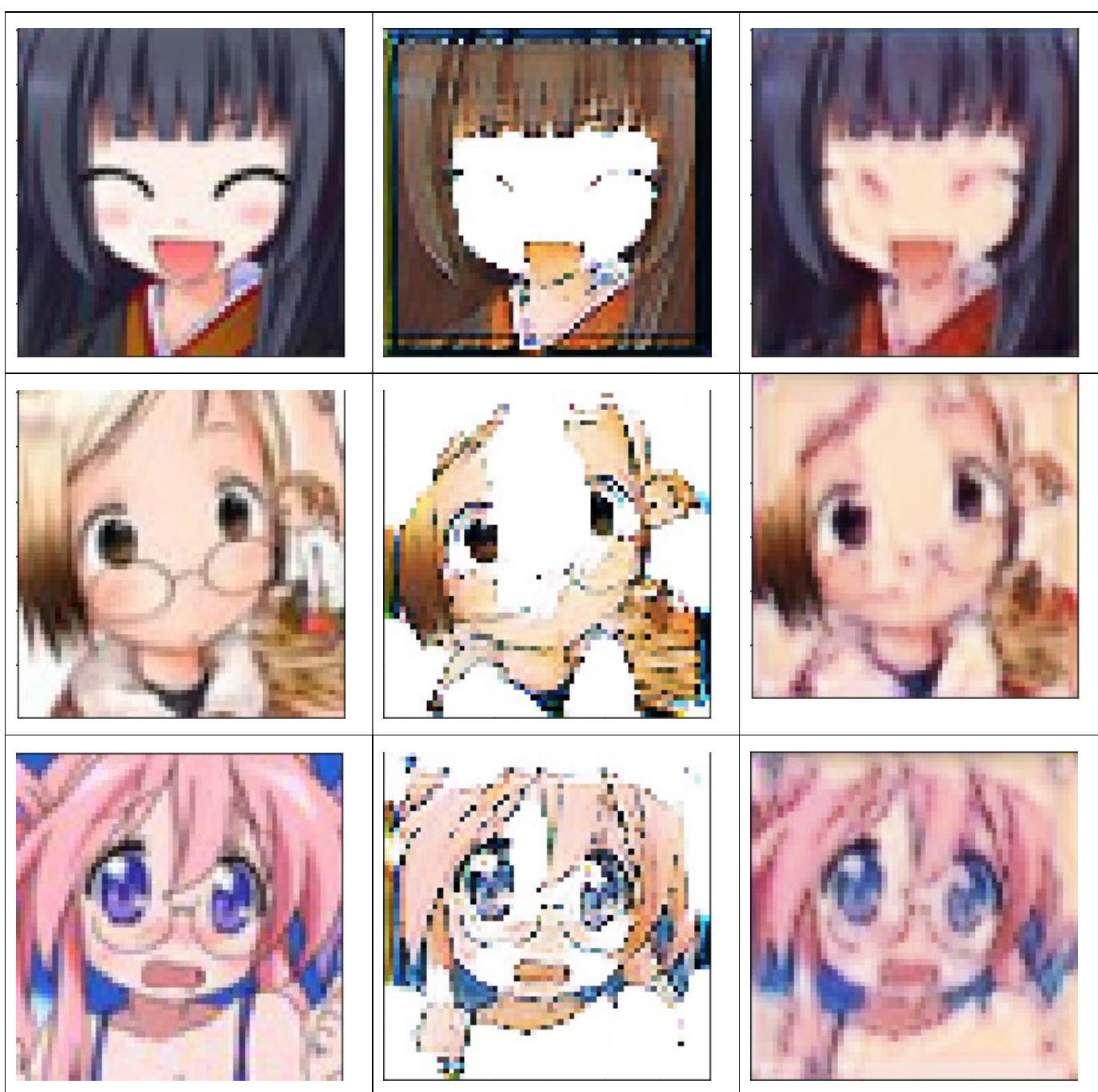


丟入 animation 測試:

原 Animation 圖	經 Generator B 變 Cartoon	再經 Generator A 變回 Animation
		
		
		
		







根據人眼的觀察，cartoon 與 animation 的差別便是：cartoon 使用的線條較粗糙，臉的輪廓會用黑色去分隔頭髮與臉，並且除了人物的頭像外，背景都是白色的。而在轉換的結果可以明顯看出這些特點。將 cartoon 轉為 animation 後，顏色的配置變得沒有明顯的分隔，較為連續，並且都自動產生了背景顏色。而將 animation 轉為 cartoon 後，畫風變得較為粗糙，並且臉的輪廓都用了黑色做分隔。如果原本的顏色太淡，還會被誤判為背景色，而直接轉為白色，如同倒數第 2 張圖，由於臉的額頭部分顏色過淡，直接被轉成白色了。並且原本 animation 的背景也被去除了。

GAN 之所以為對抗學習，便是利用 Generator 和 Discriminator 之間互相超越與優化來達成，理想上，Generator 產生讓 Discriminator 會辨識錯誤的結果，接著 Discriminator 優化自己，讓自己可以辨識出 Generator 生成的 data 與原 data 的差異，然後 Generator 為了再讓 Discriminator 辨識錯誤，再優化自己。但現實狀況往往會出現一方較強勢而一方覺弱小的狀況，無法取得平衡。如果 Discriminator 較強大，會造成梯度消失、如果 Generator 較強大，會造成模式崩潰。

Gradient vanish: 因 Generator 的梯度更新來自 Discriminator，剛開始訓練時，tensor 皆為隨機產生的，其生成的圖片必定不佳，使得 Discriminator 能輕易判斷出 Generator 產生的圖片是假的，故

Discriminator loss 趨近於 0，導致 Generator 沒有梯度訊息，無法去優化自己，即為梯度消失。

Mode collapse: 由於 Generator 太強，產生的圖片太像真的，使得 Discriminator 無法判斷出哪些是真的與生成出來的圖片。如果這時候其實 Generator 生成的圖片還不夠真實，Discriminator 又無法分辨真假，就會造成 Generator 不再進步，繼續用現在程度的照片丟給 Discriminator。最後生成出來的照片結果也會不佳，沒有達到能夠以假亂真的效果。

感覺生成出來的圖片並沒有非常接近該類別，比如 cartoon 通過 Generator A 變成的 animation 圖並沒有非常的像原本就屬於 animation 的圖片，頭的大小仍是小的，只是臉的顏色、背景風格變成 animation 版的。