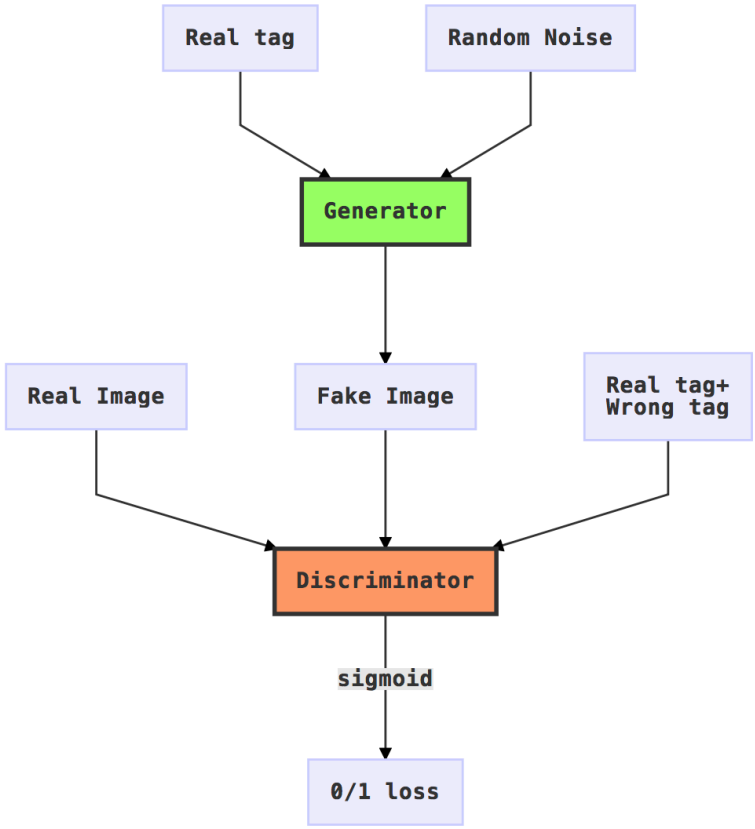
ADL hw4 Report

B04902004 王佑安

Model Description

Generator structure:

- Dense*1
- Conv2d*4
- Activation: RELU
- BatchNorm=True



Discriminator structure:

- Conv2d*4
- Dense*2
- Activation: RELU
- BatchNorm=True

Epochs: 150

Learning rate: 0.0002

Batch size: 64

Noise dim: 100

Embed dim: 25

Optimizer: Adam

Loss Function:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Improvement

1. Wrong tag sampling:

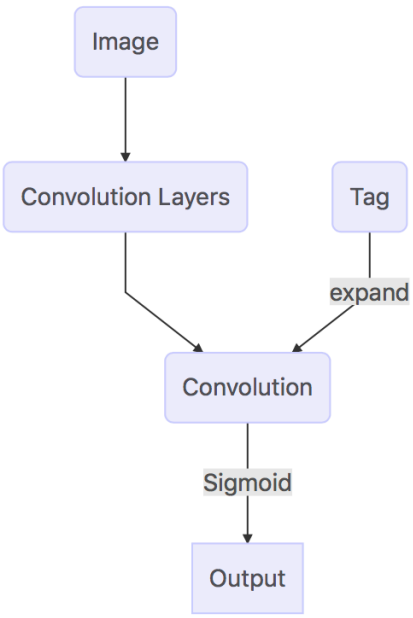
由於原本 data 中 tag 的分布非常不平均，有將近一半的圖片都沒有任何髮色跟眼睛顏色的 tag，在 train discriminator 時如果單存純從原本的 data 中 random sample 錯的 tag，很容易讓 discriminator 學到的只有如何分辨沒有 tag 的圖片，而不是圖片的 tag 正不正確。由於我將 tag embedding 的方式是直接將髮色跟眼睛顏色 one-hot，因此可以直接 random data 中有的顏色在轉 one-hot，就能讓產生的 wrong tag 分布平均。

2. Discriminator tag concatenation:

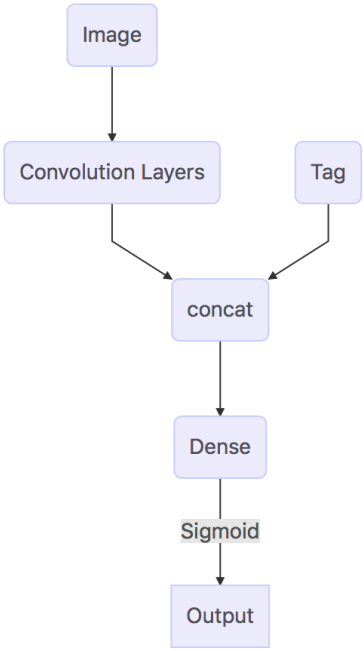
一般的 DCGAN 在 discriminator 將 tag 跟 image concat 時通常會將圖片通過 convolution layers，再將 tag expand 成跟圖片一樣的的大小接在一起當成新的 channel 再過一層 convolution。但這樣 tag 的資訊會被放大，導致原本 image 分辨的能力會變弱，generator 只要生出顏色正確的圖就能騙過 discriminator，而影響圖片其他資訊的 noise 變得不重要導致 generator collapse，產生的圖片都長得差不多。

為了解決這個問題，我在 discriminator 的架構作了修改，原本 expand tag 的部分改成直接跟通過 convolution 的 image concat 在過 dense，沒有經過 expand 的 tag 的資訊不會太強，修改後 generator 就不會 collapse 了。

(tag expand)



(tag concat)



Experiment & Observation

1. Noise 對生成圖片的影響：

   為了測試 Noise 是否真的讓 generator 產生不同圖片而不是 over fit training data，測試了給定相同 tag 不同 noise

   (tag: blonde hair aqua eyes)

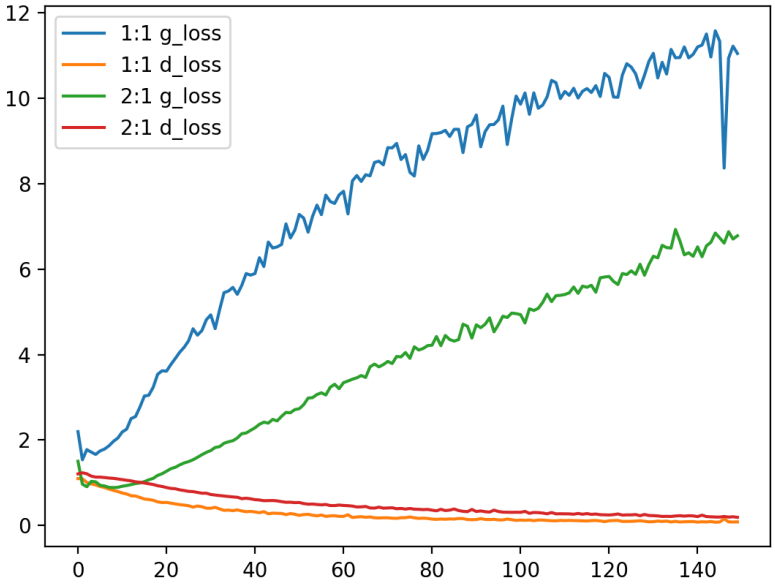
   可以發現 generator 生成 5 張符合條件的不同圖片。但我們還是不知道這 5 張不同的圖片是否是 over fit training data 中 5 張不同圖片的結果，因此做了第二個測試，給定相同 noise 不同 tag。

   (tag: 12 種不同 hair color + random eyes color)

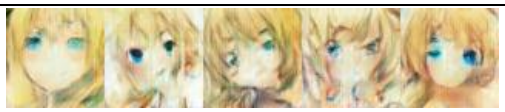
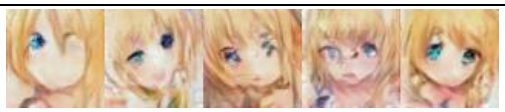
   除了髮色跟眼睛顏色，臉型表情幾乎一模一樣，因此可以得知 generator 確實有學到 tag 跟 noise 代表的意義。由於 tag 是由髮色跟眼睛顏色直接 one-hot 得到的，沒有其他資訊，因此 noise 就能代表其他全部資訊，而 generator 在相同 noise 的情況下能產出除了 tag 給的資訊外其他條件全部相同的圖片。

2. Generator/Discriminator 更新頻率

在調整 hyperparameters 時我發現 generator/discriminator 更新頻率對 model fitting 的速度影響很大，於是將 G:D=1:1 跟 2:1 的 loss 畫出來。



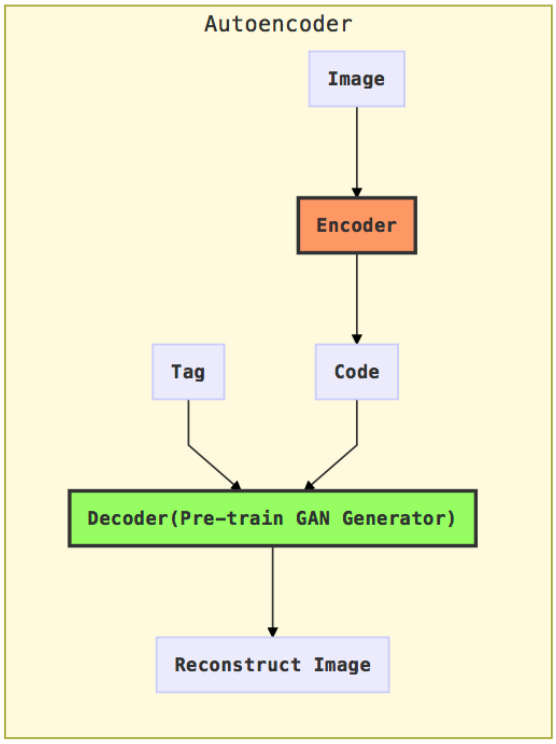
可以觀察到 1:1 的 g_loss 上升非常快大約 20 個 epoch 就超過 5 了。我將兩種 setting 在 training 過程的生成的圖片畫出來

Epoch	G:D=1:1	G:D=2:1
5		
20		
100		

可以發現 1:1 雖然確實跟 loss 下降的速度一樣很快的學會如何畫出正確的顏色，但 train 到後面 2:1 的結果卻比 1:1 清楚很多，五官位置也都比較正確比，1:1 的後面卻幾乎沒在進步。可以推論因為 1:1 的 generator 更新比較快，所以很快地就能學到如何騙過 discriminator，但由於 discriminator 的能力跟 generator 差太多，導致兩邊都學不到東西，而 2:1 的讓兩邊保持差不多能力互相對抗才能不斷進步，train 到最後就比 1:1 的好了。

Style Transfer

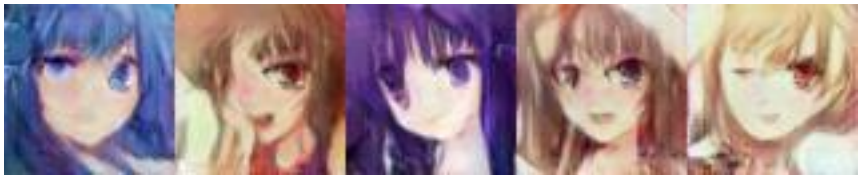
我做的是把這次作業的 training data 髮色/眼睛顏色的 transfer。先將 pre-train 好的 Generator 當作 decoder 直接 train 一個 auto encoder，之後只要替換 decoder 的 tag 就能將 input 轉換髮色\眼睛顏色。



由於 pre-train 的 generator 效果本來就沒有很好，直接用 autoencoder 做出來的 reconstruct image 還原度也沒辦法太高，但臉型跟髮型大致上還是有對到。

原始圖片

Reconstruct



Transfer (aqua hair aqua eyes)