

MLDS 2017 Spring HW3 - Generative Adversarial Networks

B03901056 孫凡耕 B03901070 羅啟心 B03901032 郭子生 B03901003 許晉嘉

1. Environment (1%)

OS	CPU	CPU Memory	GPU	GPU Memory
Arch linux 4.10	i7 3.6 GHz	32 GB	GTX 1080 Ti	11 GB

使用 Library: Tensorflow, GloVe, Theano, NumPy, SciPy, nltk, skipthoughts, scikit-image, h5py

2. Model Description (3%)

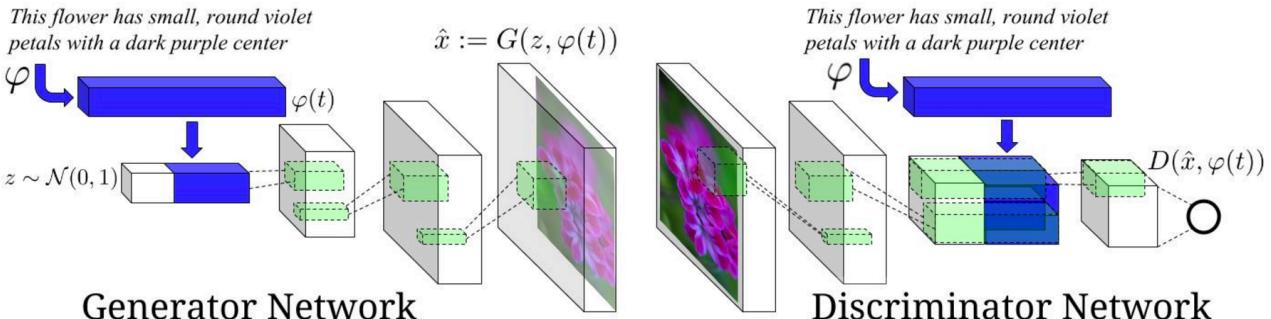


Figure 1. Image Source from Generative Adversarial Text-to-Image Synthesis Paper

我們實做的是 Conditional GAN 的架構，其中 GAN 使用的是 DCGAN，並使用多種不同產生 vector 的方式來作為 input 的 vector。DCGAN 的架構基本上是和 Paper 相同（如上圖 Figure 1. 所示），其中較詳細的參數可參考下面的使用參數 section。

- 使用參數 Parameter :

`z_dim` : 100、`batch_size` : 64、`image_size` : 64×64、`gf_dim` : 64 (第一層convolution layer的output)、`df_dim` : 64 (第一層deconvolution layer的input)、`gfc_dim` : 64 (fully connected layer的output)、`beta1` : 0.7 (Adam optimizer 的momentum)、`epochs` : 320、`learning rate` : 0.0002

- Objective Function :

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))].$$

3. Improvement (5%)

- Vectors

我們原本考慮使用 one-hot 來做 caption 的 word embedding，然而在考慮 performance 之後我們主要嘗試了另外兩種 vectors，分別為 GloVe 以及 Skipthoughts。GloVe 包含 50、100、200、300 四種維度，Skipthoughts 則包含 uni-skip、bi-skip、以及combine-skip。其中 uni-skip 使用的是前 2400 維，bi-skip 使用的是後 2400 綴、combine-skip 則是將兩者 cascade 共 4800 綴。下圖 Figure 3. 為各個 vectors 的 output images。另外右圖 Figure 2. 顯示的則是使用不同vector時 generator 以及 discriminator 隨著epoch所產生的變化。

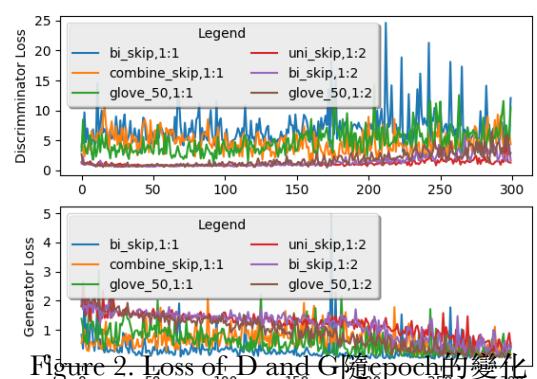


Figure 2. Loss of D and G隨epoch的變化

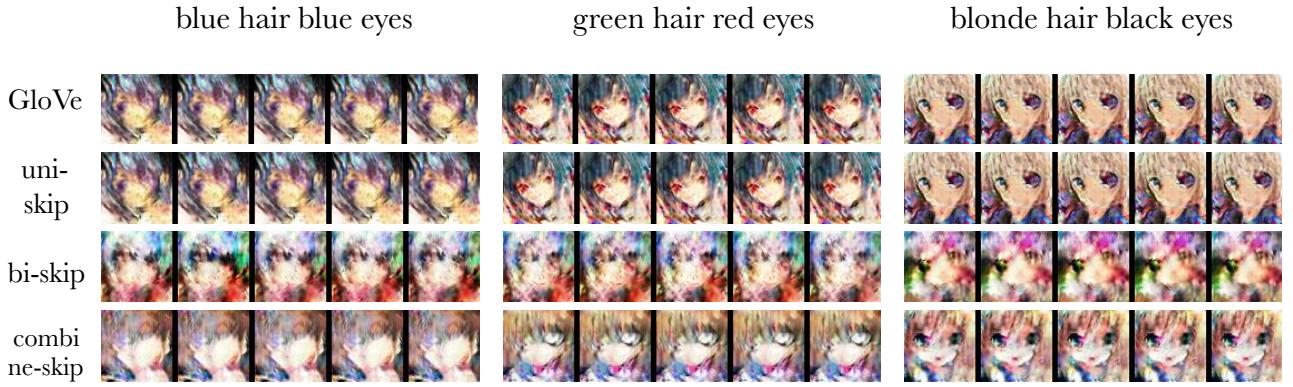


Figure 3. 使用不同 vector 所產生的 output image

- WGAN

我們除了 Basic 的 DCGAN 之外，也嘗試寫 WGAN，兩者的差別主要有四個：1. discriminator 的 output 不使用 sigmoid。2. clip discriminator 的 weight。3. 用 RMSProp 取代 Adam Optimizer。4. discriminator 的 training 次數多於 generator (Paper使用五次)。然而礙於時間限制最後我們並沒有使用 WGAN 作為我們使用的 Model 。

- Update Rate

都市傳說表示 discriminator 的 iteration 次數要小於 generator。因此我們將嘗試找出較好的比例作為 improvement。下圖 Figure 4. 使用 skipthoughts 以及 GloVe vector 作為 input vector，觀察 discriminator 及 generator loss 隨著 epoch 數目的變化。結論證實大約在 1:2 左右會有較好的 output 結果。

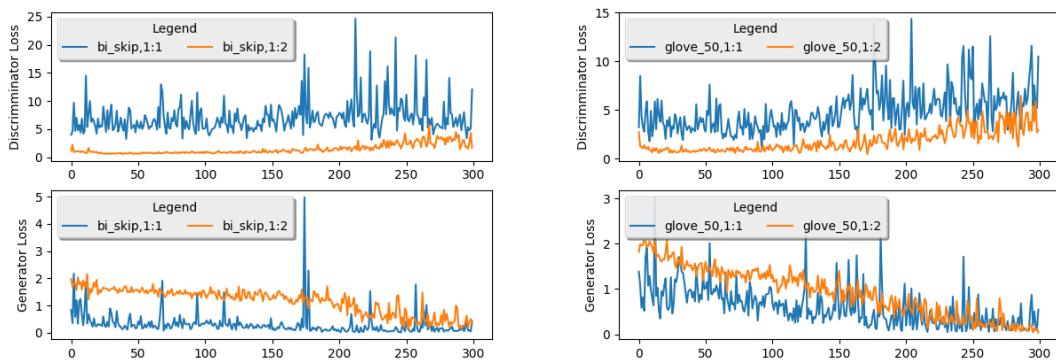


Figure 4. 不同 D & G iteration 次數的 loss 隨 epoch 增加之變化

- Discriminator Output

在我們的 Model 中，我們有使用助教所提供的 training tips，也就是將 fake image, right text、real image, wrong text、以及 wrong image, right text 皆讓 discriminator 判斷成錯誤的結果。我們的做法是把這三個的 loss 相加起來成為一個 loss，再統一由一個 optimizer 來 reduce 這個 loss 。

4. Experiment (5%)

- Epoch

我們嘗試觀察不同 training epoch 下 Generator 所產生的 output 圖案。以下 Figure 5. 中所顯示的即是不同 epoch 次數抽樣的訓練成果。可以發現當 epoch 等於 1 時，只有非常不明顯的眼睛和人臉輪廓。當 epoch 達到 100 時，開始有明顯的人臉出現，然而其中

並無太多的細節。當 epoch 來到 200 時，開始出現較明顯的細節，然而還是會有許多重複的圖案出現。當 epoch 超過 300 後，基本上圖片就以含有大量的細節，然而仍然有機會出現相同的圖片，這個問題即做 GAN training 時常遇到的 mode collapse，也就是Generator 會將不同的 input （加上noise） mapping 到同一張圖片。mode collapse 常見的解決做法包含：使用 minibatch 增加多樣性、加入 counterplay 如 minmax 等等。



Figure 5. 在不同Epoch下的實驗結果，各數字代表training的epoch次數

- Diversity

我們在 input 文字上做了如下 pseudo code 中的 iteration，output 的結果如 Figure 6. 所示。圖中橫軸為眼睛不同的顏色，縱軸則為頭髮不同的顏色，可以發現 output image 的確隨著不同的 input 文字而有明顯的改變。其中頭髮相對於眼睛而言較為明顯，我們猜測是由於頭髮在圖中的面積較大，因此 machine 較容易判斷所致。除此之外，也可以發現不同顏色的影響不盡相同，舉例來說綠色所造成的影響特別顯著。

Pseudo Code:

```

enum color = {brown, aqua, black, red, orange, yellow, green, purple, blue, gray, pink}
for i = 0 to 10
    for j = 0 to 10
        input = color[ i ] + hair + color[ j ] + eyes
        show image = Generator( input )
    end
end

```

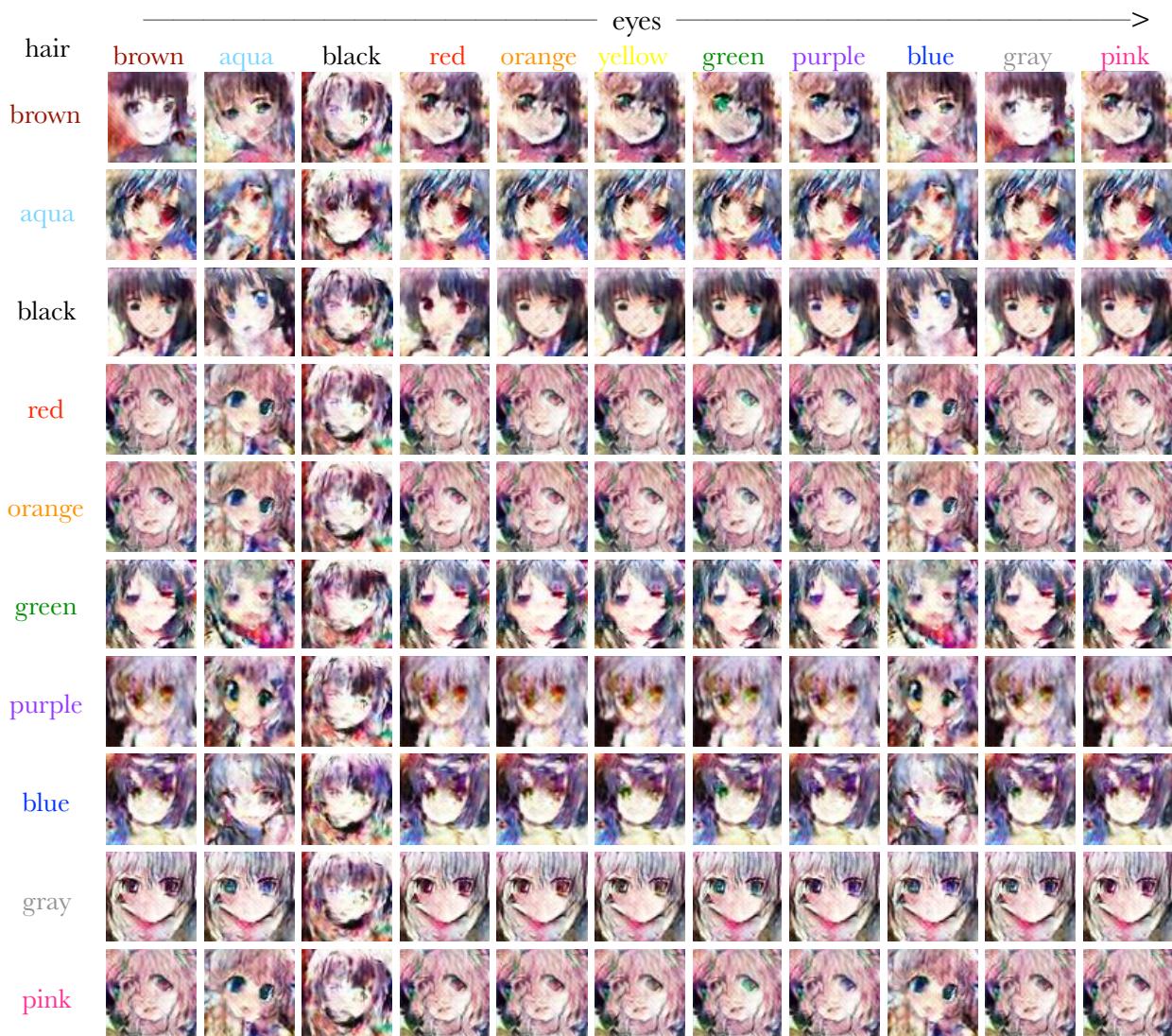


Figure 6. Generator Output Image

5. Bonus (10%)

- Glasses and Sun Glasses

在 Bonus 的部分，我們首先嘗試訓練 machine generate 出含有 Glasses 以及 Sun Glasses 的圖片。結果如 Figure 7. 所示。可以發現輸入 Glasses 的結果的確有眼鏡的出現，然而在 Sun Glasses 的 case 結果就沒有那麼順利。經過分析後我們推測原因是含有 Glasses tagging 圖片的數量約是 700 多張，然而 Sun Glasses 的圖片則只有十張左右。



Figure 7. Output Image with Tag ‘Glasses’ 和 ‘Sun Glasses’

- Hair Style

在訓練完眼鏡的部分之後，我們接著嘗試訓練不同的髮型，包含長髮、短髮、雙馬尾、以及長馬尾。即便 long hair 以及 short hair 在所有 Tag 的數量當中分別位居第一名以及第三名，然而這部分的訓練成果並沒有太顯著，成果如 Figure 8. 所示。我們認為原因是 faces 上的 data 已經將大部分的髮型裁切掉，因此事實上光就人臉的部分並無法完整呈現這些髮型的不同處。

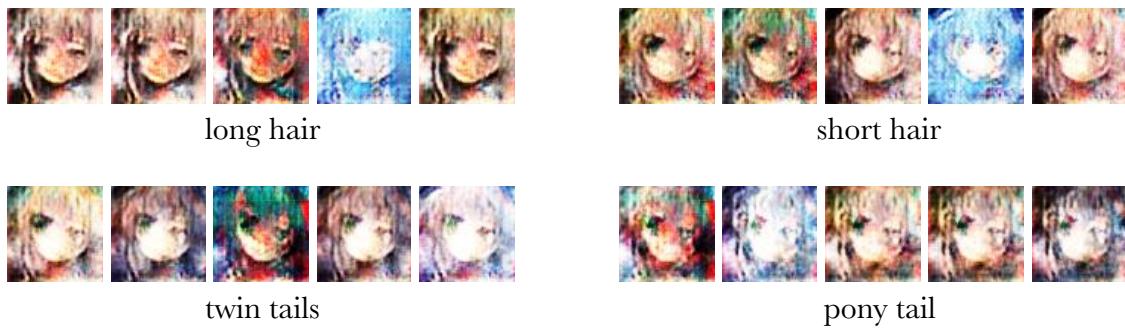


Figure 8. Output Image with 不同的髮型

- Full Image

最後一個 Bonus 我們嘗試在整張完整的圖片上做 training，我們首先把最常出現前 200 名的 Tag 取出來，然後只把含有這前兩百名 Tag 的圖片當成 training data。舉例來說假如圖片 X 有 Tag A、B、C。其中只有 Tag A 和 B 位於前 200 名，則我們就把 the girl has A and B 做為 caption input 轉換成 skipthoughts vector 傳進 model。圖片的話則把所有大小都 resize 成 64×64 做 training。Figure 9. 為部分 generator 所產生的結果，可以發現包含背景的成分，且圖片也大致符合 Tag 的主題，但由於解析度不夠的關係並無法表現出細節，除此之外也有些圖片看起來並沒有任何明顯的主題。

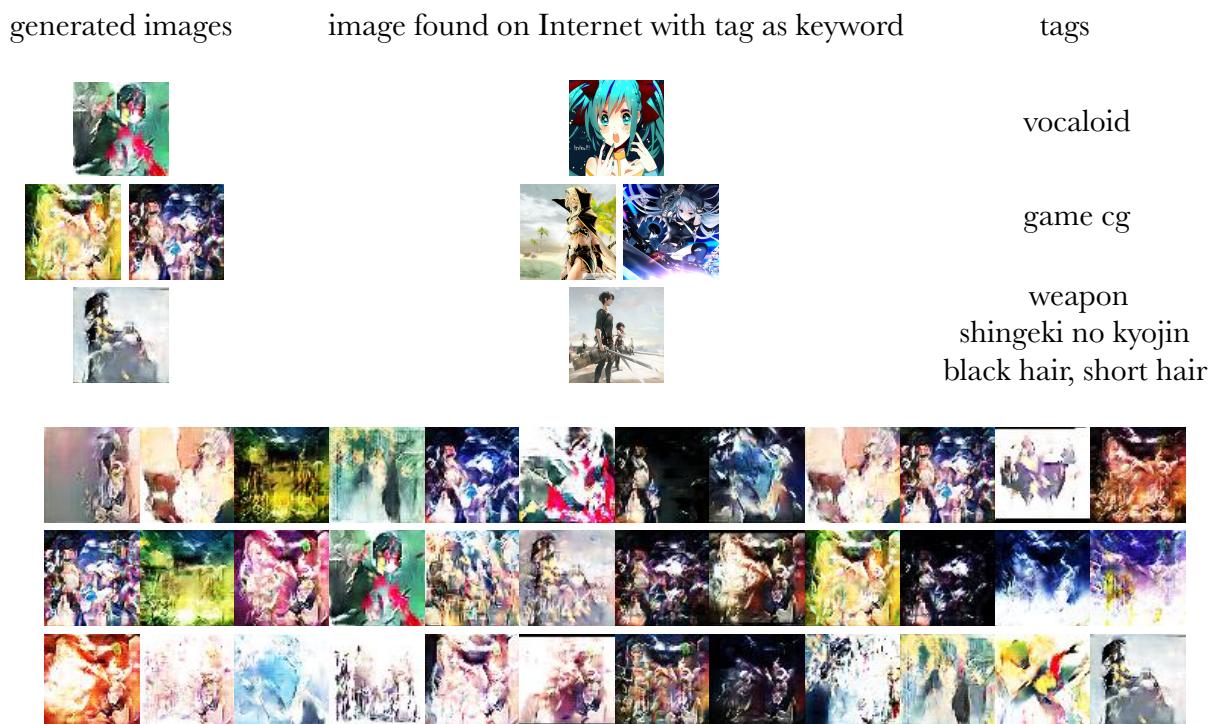


Figure 9. Bonus Full Images

6. Team Division (1%)

B03901056 孫凡耕：分配組內工作、教導組員、跑實驗、Bonus

B03901070 羅啟心：Conditional GAN、跑實驗

B03901032 郭子生：撰寫報告、Bonus

B03901003 許晉嘉：協助餘項事務