

MLDS HW3 Report

組別：我才是真的Baseline

組員：f03942038鍾佳豪 / r05942102王冠驊 / d05921018林家慶 / d05921027張鈞閔

Environment

OS: Windows 10

CPU: Intel Xeon CPU E5-1630 v3 @ 3.7 GHz with 64GB RAM

GPU: GeForce GTX TITAN X (Pascal)

Python version: 3.4.2

Libraries: tensorflow 1.0.0, numpy 1.12.0, scipy, skimage, skipthoughts, matplotlib, argparse

Data Preprocessing

A. Image Processing

我們將全部 33431 張圖片利用 skimage 縮小成 64x64，存在 faces 資料夾。

B. Tags Processing

我們從 tags_clean.csv 擷取出包含 'hair' 及 'eye' 關鍵字的 tags，有三種情況：

(a) 只使用有完整 tag 敘述的圖片，並將所有 tags 銜接成一句描述，如 'blonde hair blue eyes'，捨棄完全沒有提到 'hair' 或 'eyes' tags 的圖片後，剩下 **18175** 張；

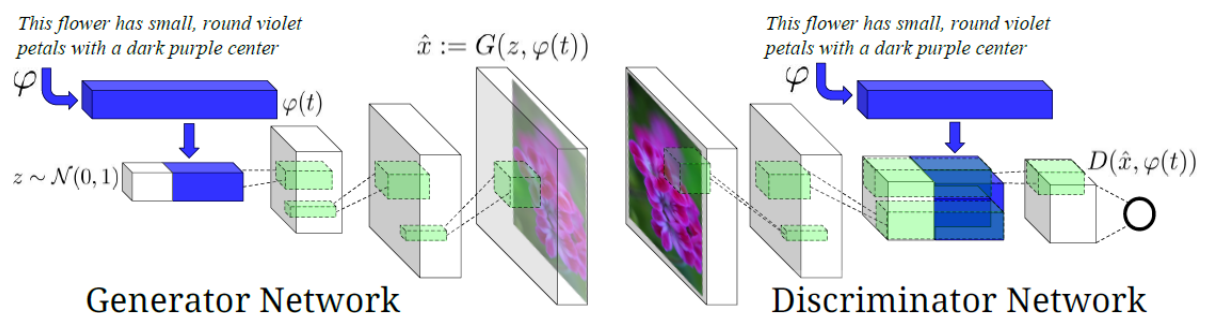
(b) 只保留 'hair' 和 'eyes' 都只包含一種顏色的圖片，使每張圖片都對應到唯一的頭髮顏色 (例如 'blonde') 和眼睛顏色 (例如 'blue')，剩下 **11569** 張；另外，不將 tags 銜接成一句將 tags 的描述，拆成 'hair' 的描述和 'eyes' 的描述。

(c) 使用全部 **33431** 張圖片，將未知顏色的 tags 補上 **unknown**；如 (b)，不將 tags 銜接成一句描述，而將其拆成 'hair' 的描述和 'eyes' 的描述。

我們接著利用 skipthought model 將敘述轉成 4800 維的向量，存成 dictionary (key 為圖片檔名，value 為 embedding vector)，寫入 vec_hair_eyes.pkl (**case a**)，以及 vec_hair_short.pkl 和 vec_eyes_short.pkl (**case b,c**)，其中 **case (b,c)** 每一張圖片有 4800 維頭髮顏色向量和 4800 維眼睛顏色向量，一共 9600 維。

Model

我們依照 Generative Adversarial Text to Image Synthesis 這篇 paper 提出的模型完成實作，model 架構如下：



在 Generator 當中，我們先將文字敘述 (case a 有 4800 維，case b,c 有 9600 維) 用一層 linear layer 投射到 256 維，再接上 100 維的 random noise。接著通過一連串的 de-convolution，依序得到：512 個 4x4 feature maps、256 個 8x8 feature maps，128 個 16x16 feature maps，64 個 32x32 feature maps，最後得到 3 個 channels (RGB) 的 64x64 輸出圖片。

在 Discriminator 當中，我們直接將輸入的圖片 (64x64x3) 通過一連串的 convolution，依序得到：64 個 32x32 feature maps、128 個 16x16 feature maps，256 個 8x8 feature maps，512 個 4x4 feature maps。此時按照 paper 提出的方法，將 text features (case a 有 4800 維，case b,c 有 9600 維) 也用一層 linear layer 投射到 256 維，再轉成 4x4x16 的 tensor，接在 4x4x512 feature maps 的後面。接著依序通過 1x1 及 4x4 的 convolution 得到輸出 score。

Generator 的 activation function 使用 relu，最後一層為 tanh。Discriminator 的 activation function 使用 leaky-relu (leak = 0.2)，最後一層視情況直接輸出或使用 sigmoid。兩者都使用 batch normalization (epsilon=1e-5, momentum = 0.9)。

至於 Discriminator loss function 的定義及整體 model 的 optimization 方式，我們實做了 **DCGAN**，**WGAN** (weight clipping)，及 **improved WGAN** (gradient penalty)，分別敘述如下：

A. DCGAN

如 paper 所提出的，Discriminator loss function 包含一種 positive case ({real image, right text}) 以及兩種 negative cases ({real image, wrong text} 和 {fake image, right text})。我們直接將三種 cross entropy 做加權平均。Optimizer 使用 Adam (learning rate = 0.0002，momentum = 0.5)，每一個 batch 會先 update Discriminator 參數 1 次，再 update Generator 參數 2 次。

```
# Objective function of Discriminator:
## D_logits_real, D_logits_fake, and D_logits_wrong: Discriminator outputs
## of one positive case and two negative cases, respectively
d_loss_real = tf.reduce_mean(
    tf.nn.sigmoid_cross_entropy_with_logits(
        D_logits_real, tf.ones_like(D_logits_real)))
d_loss_fake = tf.reduce_mean(
    tf.nn.sigmoid_cross_entropy_with_logits(
        D_logits_fake, tf.zeros_like(D_logits_fake)))
d_loss_wrong = tf.reduce_mean(
    tf.nn.sigmoid_cross_entropy_with_logits(
        D_logits_wrong, tf.zeros_like(D_logits_wrong)))
d_loss = d_loss_real + d_loss_fake + d_loss_wrong

# Objective function of Generator:
g_loss = tf.reduce_mean(
    tf.nn.sigmoid_cross_entropy_with_logits(
        D_logits_fake, tf.ones_like(D_logits_fake)))
```

B. WGAN

將 Discriminator 最後輸出的 sigmoid 拿掉，直接輸出最後一層 convolution layer 的 output 做為 score。並加上 weight clipping 的 operation (threshold = ± 0.01)。Discriminator loss function 和 DCGAN 類似，包含一種 positive case 以及兩種 negative cases，但是並非 cross entropy，而是直接將 positive case 的 scores 減去 negative cases 的 scores。Optimizer 使用 RMSProp (learning rate = 0.0002)，每一個 batch 先 update Discriminator 參數 5 次之後才 update Generator 參數 1 次。

```
# Objective function of Discriminator:
## D_real, D_fake, and D_wrong: Discriminator outputs
## of one positive case and two negative cases, respectively
d_loss_real = tf.reduce_mean(D_real)
d_loss_fake = tf.reduce_mean(D_fake)
d_loss_wrong = tf.reduce_mean(D_wrong)
d_loss = d_loss_real - d_loss_fake - d_loss_wrong

# Objective function of Generator:
g_loss = -tf.reduce_mean(D_fake)
```

C. Improved WGAN

Discriminator 的輸出和 loss function 的計算方式和 WGAN 相同，只是會再加上 penalty term (scale = 10.0)。Optimizer 使用 Adam (learning rate = 0.0002，momentum = 0.5)，並且也是每一個 batch 先 update Discriminator 參數 5 次之後才 update Generator 參數 1 次。


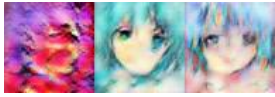



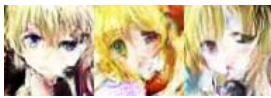


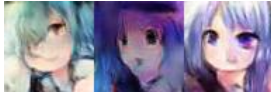
```
# Objective function of Discriminator:
## D_real, D_fake, and D_wrong: Discriminator outputs
## of one positive case and two negative cases, respectively
d_loss_real = tf.reduce_mean(D_real)
d_loss_fake = tf.reduce_mean(D_fake)
d_loss_wrong = tf.reduce_mean(D_wrong)
d_loss = d_loss_real - d_loss_fake - d_loss_wrong

# Add gradient penalty:
## inputs are real data, G are generated data, y are text embedding vectors
epsilon = tf.random_uniform([], 0.0, 1.0)
x_hat = epsilon * inputs + (1 - epsilon) * G
d_hat = discriminator(x_hat, y, reuse=True)
ddx = tf.gradients(d_hat, x_hat)[0]
ddx = tf.sqrt(tf.reduce_sum(tf.square(ddx), axis=1))
ddx = tf.reduce_mean(tf.square(ddx - 1.0) * scale)
d_loss = d_loss - ddx

# Objective function of Generator:
g_loss = -tf.reduce_mean(D_fake)
```




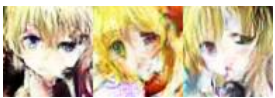

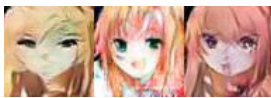

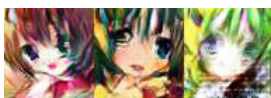
Experiments

Exp0: 比較三種 GAN，(A) DCGAN；(B) WGAN (weight clipping)；(C) Improved WGAN (gradient penalty)。Tags 處理方式採用 18175 張圖片。

GANs	(0-a) DCGAN	(0-b) WGAN	Improved WGAN
(1) blue hair blue eyes			
(2) blonde hair black eyes			
(3) white hair green eyes			

Discussion0: 對比 WGAN 和 Improved WGAN 的圖片，由於 WGAN 存在著訓練困難、收斂速度慢的問題，這可能是我們 WGAN 結果較差的原因，而 Improved WGAN 則是如預期般地有最佳的表現。DCGAN 會有 mode collapse 的問題，使得產生的圖片樣式都較類似相同。Exp0 的結果顯示 **Improved WGAN** 明顯優於其他兩種 GAN，因此接下來的實驗都只用 **Improved WGAN** 來進行。

Exp1: 比較三種 tags 處理方式，(a) 只使用包含 'hair' 或 'eyes' 關建字的 18175 張圖片，合併成一句描述；(b) 只使用同時包含 'hair' 和 'eyes'，且分別都只包含一種顏色的 11569 張圖片，將 'hair' 和 'eyes' 分別表示成 4800 維向量；(c) 使用全部 33431 張圖片，以 unknown 填補未知的 tags，將 'hair' 和 'eyes' 分別表示成 4800 維向量。

Tags processing	(1-a)	(1-b)	(1-c)
(1) blue hair blue eyes			
(2) blonde hair black eyes			
(3) white hair green eyes			





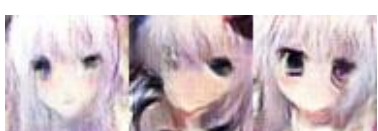

Discussion1:

(1-a) 將 'hair' 和 'eyes' tags 銜接成一句描述再轉成 4800 維的向量，會使得不同資訊被揉合在一起，如 (1-a-2) 金髮黑眼的第二張眼睛染上金色、(1-a-3) 白髮綠眼的第一張變成了綠髮，正確度較差。

(1-b) 將 'hair' 和 'eyes' 分開，各以 4800 維的向量表示，畫出來的圖不會有明顯的顏色錯誤，正確度較佳，但由於使用的圖片數量較少，真實度較差，產生出來的圖片比較模糊，如 (1-b-1) 第三張、(1-b-2) 第一張、(1-b-3) 第二張的五官。

(1-c) 同 (1-b) 將 'hair' 和 'eyes' 的描述分開表示，但將沒有顏色描述的 tags 補上 unknown 增加訓練的圖片數量。由於 (1-c) 使用較多的圖片訓練，真實度較佳，但因為加上許多 unknown tags，對訓練而言是很多的 noise 干擾，所以正確度稍差。

Exp2: 比較 Discriminator loss function 中，positive scores 與 negative scores 的合併方式 **(a)** 直接相加；**(b)** 將兩種 negative scores ({fake image, right text} 和 {real image, wrong text}) 都乘上 0.5，如 paper 的方法。Tags 處理方式採用 **(case 1-b)** 11569 張圖片。

Pos/Neg Loss	(2-a)	(2-b)
(1) blue hair blue eyes		
(2) blonde hair black eyes		
(3) white hair green eyes		

Discussion2: 若照 paper 的建議，兩種 negative scores 各乘上 0.5 以平衡 positive 和 negative scores，訓練後期整個 model 就壞掉了，反而直接將 positive 和 negative scores 相加的效果比較好。在訓練 GAN 時是否有平衡 positive 和 negative scores 的需要，這原因值得再多花心力研究。

Team Division

f03942038 鍾佳豪	Tag processing; Experiment 1
r05942102 王冠驊	Image processing; Experiment 2
d05921027 張鈞閔	Tag processing; Experiment 0; Report
d05921018 林家慶	DCGAN, WGAN, Improved WGAN implementation; Report; Experiment 0-2