ADL x MLDS Fall 2017

Homework 4 Report

Student: CSIE R06922068 Yu-Jing Lin 林裕景

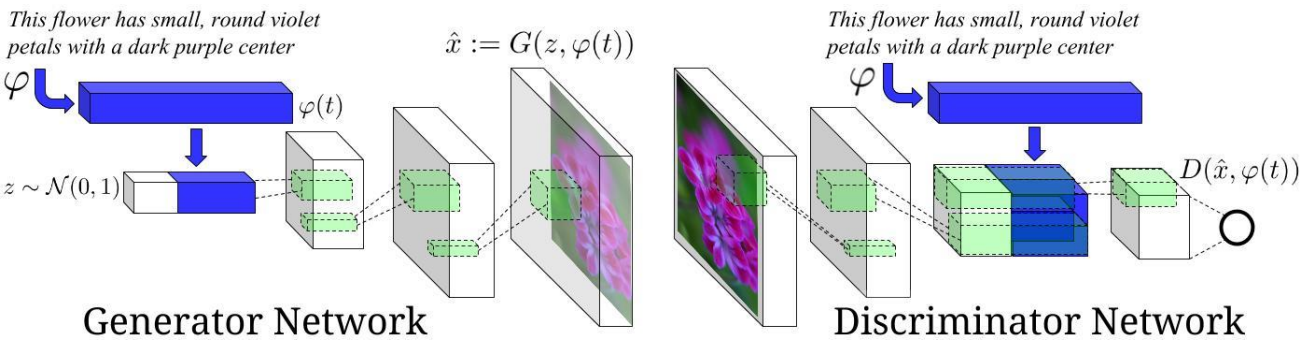
Homework: text2image

Model Description

我在這次作業 HW4 實作了兩種生成對抗網路 GAN。

一開始，聽說 WGAN-GP 與 skip-thoughts 很厲害，因此用別人的 WGAN-GP 修改，加上把 tags 串接在一起經過 skip-thoughts 轉換出的 condition（2400 維），打算做出既穩定又能根據條件生成圖案的 GAN。架構上的差別主要是：在原本 WGAN-GP 的 generator 中，將 condition 也作為 input，透過 dense（成 256 維）和 relu 後與 noise（100 維）串接；而在 discriminator 中，則是將 condition 過同一組 dense 和 relu，再與倒數第二個 conv2d 疊合，接著通過最後一個 conv2d 以及 dense，輸出 scalar。而 loss 的部分則增加為四種 loss：(real data, right label), (fake data, right label), (real data, wrong label), (wrong data, right label)，並保留 gradient policy 的部分。然而，訓練出來的網路雖然可以產生動漫人物臉，但是卻沒有學到 condition，似乎把 condition 當成雜訊一樣，因此我後來轉到另一條路。

第二個（如下圖），才是我這次成功的網路，助教在投影片中提供的 text2image open source，是一個 conditional CDGAN，我將它搭配 one-hot encoding condition，成功的產生出條件性的動漫人物臉。首先，我將每個動漫人物的 tags 先做過濾，取出作業指定顏色的頭髮和眼睛的 tags，然後對兩種類別分別做 one-hot encoding（先全設零，只要有出現就設 1），再串接起來成為一個 23 維度的 vector（頭髮 12 維、眼睛 11 維）。再來，用 Keras 實作了一遍 text2image 的架構，基於 DCGAN（主體為 CNN 連接而成）的網路，用前一段寫的方式插入 condition。然後定義四種 loss：(real data, right label), (fake data, right label), (real data, wrong label), (wrong data, right label)，discriminator 的 loss 是四個 loss 的總和，而 generator 的 loss 則是第二個 (fake data, right label) 加上負號。使用 AdamOptimizer。



網路的架構如下：　　（左邊是 discriminator；右邊則為 generator）

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 64, 64, 3)	0	
conv2d_1 (Conv2D)	(None, 32, 32, 64)	4864	input_1[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 16, 16, 128)	204928	leaky_re_lu_1[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 128)	0	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 8, 8, 256)	819456	leaky_re_lu_2[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 8, 8, 256)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 4, 4, 512)	3277312	leaky_re_lu_3[0][0]
input_2 (InputLayer)	(None, 23)	0	
leaky_re_lu_4 (LeakyReLU)	(None, 4, 4, 512)	0	conv2d_4[0][0]
lambda_1 (Lambda)	(None, 4, 4, 23)	0	input_2[0][0]
concatenate_1 (Concatenate)	(None, 4, 4, 535)	0	leaky_re_lu_4[0][0] lambda_1[0][0]
conv2d_5 (Conv2D)	(None, 4, 4, 512)	274432	concatenate_1[0][0]
leaky_re_lu_5 (LeakyReLU)	(None, 4, 4, 512)	0	conv2d_5[0][0]
flatten_1 (Flatten)	(None, 8192)	0	leaky_re_lu_5[0][0]
dense_1 (Dense)	(None, 1)	8193	flatten_1[0][0]

Total params: 4,589,185
Trainable params: 4,589,185
Non-trainable params: 0

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 100)	0	
input_4 (InputLayer)	(None, 23)	0	
concatenate_2 (Concatenate)	(None, 123)	0	input_3[0][0] input_4[0][0]
dense_2 (Dense)	(None, 8192)	1015808	concatenate_2[0][0]
reshape_1 (Reshape)	(None, 4, 4, 512)	0	dense_2[0][0]
batch_normalization_1 (BatchNor	(None, 4, 4, 512)	2048	reshape_1[0][0]
activation_1 (Activation)	(None, 4, 4, 512)	0	batch_normalization_1[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 8, 8, 256)	3277056	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 8, 8, 256)	1024	conv2d_transpose_1[0][0]
activation_2 (Activation)	(None, 8, 8, 256)	0	batch_normalization_2[0][0]
conv2d_transpose_2 (Conv2DTrans	(None, 16, 16, 128)	819328	activation_2[0][0]
batch_normalization_3 (BatchNor	(None, 16, 16, 128)	512	conv2d_transpose_2[0][0]
activation_3 (Activation)	(None, 16, 16, 128)	0	batch_normalization_3[0][0]
conv2d_transpose_3 (Conv2DTrans	(None, 32, 32, 64)	204864	activation_3[0][0]
batch_normalization_4 (BatchNor	(None, 32, 32, 64)	256	conv2d_transpose_3[0][0]
activation_4 (Activation)	(None, 32, 32, 64)	0	batch_normalization_4[0][0]
conv2d_transpose_4 (Conv2DTrans	(None, 64, 64, 3)	4803	activation_4[0][0]
activation_5 (Activation)	(None, 64, 64, 3)	0	conv2d_transpose_4[0][0]

Total params: 5,325,699
Trainable params: 5,323,779
Non-trainable params: 1,920

How do you improve your performance?

在 performance 方面，可以分為 training 成效和 training 效率。

Training 成效的方面，首先我從 data 品質著手，眾所皆知這次的 anime faces dataset 有很多出問題的部分，觀察一下 data，發現有的圖片切到的不是人臉，有的原圖有很多人但只切出一個人的臉，因此 tags 有很多髮色和眼色。因此我打算用一個簡單的方法過濾不要的 data，把沒有出現任何指定 tags 的圖片，以及在 23-dim 的 encoding 中，出現超過兩個 1 的圖片（原圖應該有很多人物），都過濾掉，留下來的會是圖片與 tags 比較相符的 data。第二，由於一般的 GAN 適合生成小圖，64x64 的 size 接近極限了，所以我一開始生成 96x96 的成果最後出現歪扭曲斜的臉，後來將所有圖片讀取進來前先 resize 成 64x64，也只生成 64x64 的圖片。

Model 的部分我也做了很多嘗試與調整（大體上是 DCGAN 加 condition 的架構），我 survey 過網路上很多的 implementation，發現大多數人使用 strides=(2, 2)的 Conv2D 與 Conv2DTranspose 來 down-sampling 和 up-sampling，而不是用 MaxPooling2D 和 UpSampling 這兩個 layer，前面提到的效果較佳。而在 discriminator 中，非線性變換使用 Leaky ReLU，而 generator 裡則用一般的 ReLU，以及只有 generator 在經過非線性前會過 batch normalization，這些改動讓 training 的過程和結果更加穩定有效。

至於 training 效率的部分，一開始我都直接讀取所有會用到的 data，因此常常吃滿記憶體，後來發現每次只讀一個 batch 的量進來訓練，完後再讀取下個 batch，這個做法大大降低了記憶體的用量，在 training 的速度上也沒有變慢的感覺。

在 Optimizer 的部分我選擇了 Adam optimizer，而其 learning rate 在試過三種 0.0002, 0.0001, 0.00005 後，發現後者由於太小，網路的 loss 收斂不了，而前兩者都可以訓練出漂亮的人物臉孔，雖然 0.0002 會讓 loss 跳動稍大，但訓練的速度縮短許多，所以最後採用 0.0002。

Experiment settings and observation

最後我採用的參數是：

```
num_channel = 3
noise_dim = 100
label_dim = 23
num_hidden_G = 64
num_hidden_D = 64
image_size = 64
batch_size = 64
lr, beta_1, beta_2 = 2e-4, 0.5, 0.99 # Adam optimizer
```

Layer 權重初始化的方法如下：

```
conv2d_init = TruncatedNormal(mean=0.0, stddev=0.02, seed=None)
dense_init = TruncatedNormal(mean=0.0, stddev=0.02, seed=None)
gamma_init = RandomNormal(1., 0.02) # batch normalization
```

訓練的過程：

學習順序：先學色調，再學頭髮（眼睛為黑洞），最後學臉型，看起來越來越像動漫人物。

Bonus: Style Transfer

Model Description

Bonus 的部分，考慮到現有的一個 domain 是動漫人物的臉，還不確定另一個 domain 要用哪種 data，因此我使用可以學習 un-paired data transfer 的 CycleGAN，來架構實作這部分的 style transfer。

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, None, None, 3)	0
conv2d_1 (Conv2D)	(None, None, None, 64)	3136
leaky_re_lu_1 (LeakyReLU)	(None, None, None, 64)	0
conv2d_2 (Conv2D)	(None, None, None, 128)	131072
batch_normalization_1 (Batch	(None, None, None, 128)	512
leaky_re_lu_2 (LeakyReLU)	(None, None, None, 128)	0
conv2d_3 (Conv2D)	(None, None, None, 256)	524288
batch_normalization_2 (Batch	(None, None, None, 256)	1024
leaky_re_lu_3 (LeakyReLU)	(None, None, None, 256)	0
zero_padding2d_1 (ZeroPaddin	(None, None, None, 256)	0
conv2d_4 (Conv2D)	(None, None, None, 512)	2097152
batch_normalization_3 (Batch	(None, None, None, 512)	2048
leaky_re_lu_4 (LeakyReLU)	(None, None, None, 512)	0
zero_padding2d_2 (ZeroPaddin	(None, None, None, 512)	0
final (Conv2D)	(None, None, None, 1)	8193

Total params: 2,767,425
Trainable params: 2,765,633
Non-trainable params: 1,792

其 discriminator 為一個多層 CNN 串接而成的網路（如右圖），基本上與 DCGAN 的 discriminator 幾乎一樣，差別只在於在這裡有用到兩層的 zero-padding，最後依然是輸出一個 scalar，表示此 discriminator 認為 input data 有多大的機會是張真的圖片。

而 generator，則是用 U-Net 架構，由於在課堂上聽老師說 U-Net 是目前公認效果比較好的架構因而採用它。其架相似於 anto-encoder，然而 U-Net 還用了 residual 的概念，將每次 down-sampling 前的圖直接接到之後對應的 up-sampling 後的那層輸出，避免重要的資訊在 encode-decode 的時候跑掉。由於其架構之 summary 過於複雜，這裡就不放了。

CycleGAN 和一般 GAN 訓練的時候有一點很不一樣，就是如果要經過 batch normalization，一次只能給少量的圖片，這裡我用 batch size 4。

Experiment result

我用了幾組 dataset 做測試，有髮色轉換，動漫真人臉轉換，幫動漫臉上初音色，三種配置，其中效果最好的是髮色轉換。在底下的結果中，第一排是原始的圖片，第二排是由 Gab 和 Gba 轉到另一個 domain 的圖片，第三排是將產生的圖片再透過 Gba 和 Gab 轉回原本 domain 的圖片。

Hair Color

Dataset A: anime faces with red hair

Dataset B: anime faces with blonde hair

將含有 red hair tag 的圖片拿出來當 domain A，含有 blonde hair tag 的圖片作為 domain B，餵給 CycleGAN 訓練，很快就能得到上面的結果。可以發現到雖然原始資料中不存在第二排的動漫人物，但看起來就跟真的一樣，毫無瑕疵。

Miku

Dataset A: grayscale anime faces　　Dataset B: anime faces of Hatsune Miku

最後，我嘗試做初音化，也就是把所有圖片都塗上初音的顏色，做法是先把圖片轉黑白作為 domain A，帶有初音 tag 的圖片過濾出來作為 domain B，然而最後大部分的角色都能成功轉過去，而由於有些初音是粉紅色頭髮或切到別的角色，因而造成 output 時頭髮出現漂染的現象，十分特殊。

Anime2Real

Dataset A: anime faces　　Dataset B: CelebFaces Attributes Dataset (CelebA)

從左邊六張圖來看，雖然有轉出真實人物的臉，但看起來很像硬把動漫臉的頭髮中間挖空，塞進一個真人臉的感覺，而從右邊六張圖則可以看得出雖然是動漫人物的臉，但表情不太搭，就結果而言是失敗的例子。令人欣慰的是，最後一張手有成功被畫出來，也看起來像動漫人物的手。

我推測，兩邊 domain 人臉佔畫面的比例不同是訓練失敗最大的主因，動漫人物多數是大臉佔滿畫面，少有背景，而真實人物則有留邊，這樣的差異會使得 CycleGAN 在學習的時候誤把動漫人物的頭髮以為是真實人物的背景，在前處理的時候將圖片裁切一致應該可以做出較好的結果，但是由於時間不夠沒多做嘗試。

Reference

1. Text2image (https://github.com/emansim/text2image)
2. WGAN (https://github.com/tjwei/GANotebooks/blob/master/wgan2-keras.ipynb)
3. CycleGAN (https://github.com/tjwei/GANotebooks/blob/master/CycleGAN-keras.ipynb)