

MLDS HW3 GAN -- Report

物理三 B03202047 王昱翔 | 物理三 B03202017 李漪廷

1 Environment

OS	Ubuntu 16.04.2 LTS	CPU	Xeon E5-2630V3
Memory	12G	GPU	NVIDIA Tesla K80
Libraries	Tensorflow r1.0, numpy 1.12.0, CUDA8.0, scipy 0.19.0, skipthoughts, skimage, Theano 0.8.2		

2 Model Descriptions

2.1 預處理

- 刪掉 hair 前面只出現 long、short、pubic 等沒有顏色意義的 tags 後，只使用同時有 hair、eyes 兩個 tags 的圖片。若同一部位有多種顏色，只取票數最高的，因此每張圖片只有頭髮、眼睛各一個顏色的特徵。(ex: “blue hair blue eyes”)
- 整理後共 10966 張圖片，對應頭髮 12 種顏色，眼睛 10 種顏色。
- skipthoughts 將特徵文字投影到 4800 維，再通過一層 fully-connected layer (fc) 和 relu 降到 256 維令成 text。
- 串接 text 跟 z (noise from uniform random, 100 維) 當 feature 輸入 Generator。

2.2 Generator of conditional-DCGAN

- 輸入 feature (256+100 維) → 通過 fc [input = 356, output = (64*4*4*8)] → reshape 成 [4,4,128] → batch-norm → relu activation。
註：所有 batch_norm 的 hyperpara [epsilon, momentum] = [1e-5, 0.9]
- 接續上一層，通過 4 層 deconv layer (kernel_size 都是 5*5, stride 都是 2, filters 各自為 [256, 128, 64, 3])，每通過一層 deconv 都經過 batch_norm → relu 再傳到下一層。

2.3 Discriminator of conditional-DCGAN

- 兩個 feature：image 和 text。image 維度 = (64,64,3)，分三類：real (符合文字敘述的圖)、wrong (不符合文字敘述的圖)、fake (G 對應文字產生的圖)；text = 256 維。
- 圖片經過 4 層 conv layer (kernel_size 都是 5*5, stride 都是 2, filters 各自為 [64, 128, 256, 512])，並都經過 leaky relu ($\max(x, 0.2 \cdot x)$)，其中後三層的 conv layer 通過 leaky relu 前都會經過與 batch_norm (hyperpara 與 generator 中的相同)。

2.4 Least Square GAN

- D 的 output 不通過 sigmoid。
- d_loss_real 修改為 $0.5 * E_{x \sim p_{data}(x)} [D(x) - 1]^2$, d_loss_fake 及 d_loss_wrong 修改為 $0.5 * E_{z \sim p_z(z)} [D(G(z))]^2$

- g_loss 修改為 $0.5 * E_{z \sim p_z(z)} [D(G(z)) - 1]^2$

2.5 WGAN

- 依照上課投影片，更改 con-GAN：(1) 去掉 D 輸出的 sigmoid。(2) 去掉 loss 中的 log。(3) 在 D 加上 weight-clipping。(4) 改成 RMSProp optimizer。

2.6 Improved-WGAN (Gradient-Penalty)

- 將 WGAN 的 d_loss 多加上 $\lambda \times mean[(slopes - 1)^2]$ ， $slopes$ 定義如下：
 $inter \leftarrow$ 在 G 的輸出和真實圖片的連線，uniform 隨機挑選的一點

$$gradients = \frac{d(Discriminator(inter))}{d(inter)} \rightarrow slopes = \sqrt{sum(gradients^2)}$$

- 改回 Adam optimizer

3 How to Improve Performance

3.1 先用沒有 conditional 的 GAN pretrain model

由於前期階段（前 20 個 epoch）用 con-GAN 產生的圖片都很模糊，甚至連人臉的輪廓都看不到（如 Fig1）。後來發現先用不考慮 condition 的 GAN（就是 d_loss 不加入 d_loss_wrong ）訓練 5 個 epoch，再加上 d_loss2_wrong ，con-GAN 就會根據原本的輪廓繼續 train（如 Fig2），會較清楚。

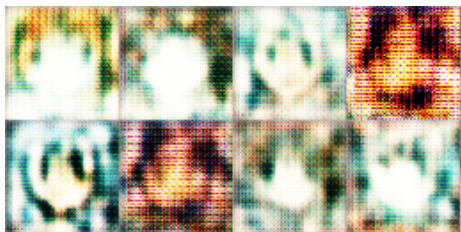


Fig1

Con- GAN with 21 epoch



Fig2 Con-GAN with 5 epoch

after training No-Con-GAN

3.2 如何選擇 wrong-image

為了使生成圖片能符合給定文字，我們需要輸入特定文字和不匹配圖片 `wrong_image`，選定過程如下：令資料能切割成共 n 個 batch，且 `right_text` 是第 k 個 batch，則 random 從 $[1, n-1]$ (included) 挑一個整數 x ，則 `wrong_image` 是第 $(x+k)\%n$ 個 batch 的真實圖片。

3.3 Text-dimension 設定為 256

測試 `text_embedd` 投影到 128 和 256 維的結果。發現輸入 256 維的 feature 在第二個 epoch 就可以產生對的顏色，若為 128 維，G 要到約第 13 個 epoch 才能產生符合文字的顏色。

3.4 調整 `g_optim` 和 `d_optim` 執行的次數比例

以 con-GAN 為例：訓練後期，D 變得非常強，容易分辨出真實和產生的圖片（ d_loss

很小，在 0.5 以下），相比之下 g_loss 大很多（約 = 1~5）。把 $d:g$ 比例從 1:2 調高到 1:5 後， g_loss 有明顯下降（降到 0.7~2），然而圖片並沒有顯著的變清晰；不過若把 D 的 $lr^*=0.2$ ，減慢 D 的訓練速度，可使我們得到最清晰的圖片。


以 WGAN-gp 為例：在論文中 d 和 g 訓練比例是 5:1，但隨訓練過程會發現 d_loss 下降比 g_loss 快很多，到 100 多個 epoch 時， g_loss 甚至大到幾十萬，這時圖片會突然變很模糊。經過調整後，發現 d 和 g 的比例在 4:3 時， d_loss 和 g_loss 能穩定地保持在同一數量級，可使輸出的圖片也穩定的變清晰。

3.5 使用 LSGAN 取代一般的 GAN

LSGAN 相較於一般的 GAN 效果較好收斂也較快，測試後發現相同 epoch 下收斂的速度的確很快，在第二個 epoch 就可以看出人臉了，但每個 epoch 需要花的時間較長，且最終似乎還是沒辦法產生出很清晰的輪廓。

4 Experiments Settings and Observation

4.1 各個 model 比較

Model	con-DCGAN	LSGAN	WGAN	WGNA-gp
Learning Rate	2e-4	2e-4	5e-5	1e-4
收斂效果 for 1 epoch	第二好	最好	慢	慢
Time / 1epoch (170 batch)	300s	800s	200s	200s
收斂所花時間	最短	第二短	最長	普通
穩定度（隨訓練 過程）	偶爾會有圖片 從清晰變回模 糊的情況	比 con-GAN 穩 定，但仍有清楚 變模糊的情況	幾乎不會有圖片從清晰 變回模糊的情況	
d:g 訓練比例	1:2~1:5	1:1	2:1	2:1~4:3
範例圖片經過 的 epoch 數	150	200	300	150
範例圖片 金髮 aqua 眼				

紅髮紅眼				
紫髮藍眼				
金髮紅眼				
這一橫排不是 同一文字特徵				

由於此次訓練的主題（人臉）並沒有很複雜，我們也沒有嘗試結構不良的 model，因此 con-DCGAN 的不穩定度並沒有特別明顯，並且，就算圖片從清晰變模糊，也能在可忍受的時間內再重新回復到清晰狀態。

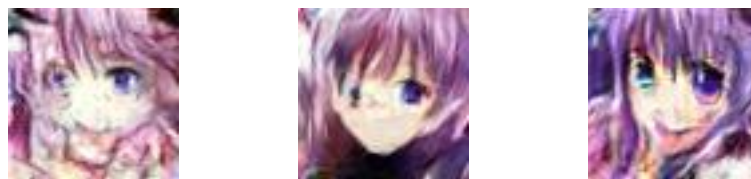
整體來說，con-GAN 也是收斂時間最短的（考慮每個 epoch 所花時間）、能在最短時間內看到清晰的人臉（我們計算資源有限，因此收斂時間是很大的考量），四個 model 中也是顏色與文字匹配度最高的（wgan-gp 雖然很清楚但顏色出錯率高，且不顯色），因此，我們選擇 con-DCGAN 作為繳交的 model。

4.2 Other Settings

- Batch size = 64
- z (noise) dim = 100
- text embedding (經過 fc layer) = 256
- beta1, beta2 for Adam optimizer = 0.5, 0.9
- weight clipping = 0.01 (for WGAN)
- lambda = 10 (constant of gradient penalty for WGAN-gp)

4.3 Observation – 演化

老師曾提過 GAN 是一個演化的過程。我們發現 training 的過程中有類似現象。Model 會先 train 好其中一顆眼睛，再頭髮，再是另一顆眼睛，如下圖由左至右。

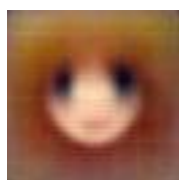


4.4 Model 可以知道產生的身體部位

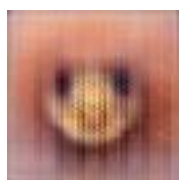
在訓練後期會發現有些不同 condition 產生的圖片，只是從同一個「模板」改變顏色。如 4.1 con-DCGAN 的第一二張範例圖片，input text 個別是金髮藍眼及紅髮紅眼，看得出來這兩張圖片除了頭髮和眼睛的顏色外幾乎一模一樣。在這個情況下，若 input text 相同，Generator 產生的圖片也幾乎一模一樣，也就是和 noise 是獨立的。若想解決這個問題，我們可對 noise train 一個 loss，讓不同 noise 有不同風格，理論上可解決這個問題。（參考助教附的論文內容）

4.5 額外增加 pre-train generator model

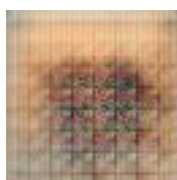
基於好奇我們想測試：如果預先將目標函數設定為減少 G 的輸出和真實圖片 pixel 間的 cross_entropy，使得在訓練一般 GAN 之前，G 已能生成較接近人臉的圖片，能不能使收斂時間縮短。結果發現，縱使 G 能輸出人臉，但在改回 minimize 一般的 GAN 中的 g_loss、d_loss 後，產生的圖片仍會先經歷變模糊的階段，再重新變清晰，因此先前的 pre-train 等於沒有實質功效，反而是多浪費時間。（下圖從左至用配合上述文字步驟）



Pre-train：直接
極小化和真實圖
片 pixel 的差距



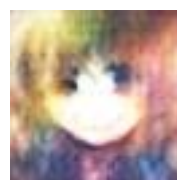
改回用一
般的 GAN
做訓練



圖片幾乎變
回和沒訓練
過一樣模糊



重新開始
出現人臉



人臉變清晰

4.6 其他

- noise 從 uniform 或 normal distribution 中 sample 出來，圖片的清晰度或真實度幾乎沒有任何差別。
- 看到許多人將 Adam 中的 beta1、beta2 設定為 0.5 和 0.9，然而和 default 值訓練出來的圖片，清晰度和真實度看起來也沒有差別。

5 Team Divisions

李漪廷：preprocess、model (WGAN、WGAN-gp)、report

王昱翔：model (con-DCGAN、LSGAN)、report、彙整並上傳