HW4 Report

資管碩一 R06725053 韓宏光

1. Model Description

(1) Model Structure

Model主要由以下5個構件所組成：

(a) discriminator(real)—判別真實圖片是否符合tag描述

輸入的image為真實圖片，caption為正確的tag描述

(以下layer皆用leaky relu作為activation function)

—輸入(64, 64, 64, 3)的image

—多次使用stride為2*2的2D convolution layer，按不同output dimension將其維度接續轉為(64, 32, 32, 64)、(64, 16, 16, 128)、(64, 8, 8, 256)、(64, 4, 4, 512)

—輸入(64, 2400)的caption vectors使用linear layer將其維度轉為(64, 256)再增擴為(64, 1, 1, 256)

—輸入(64, 64, 64, 3)的image組合與(64, 2400)的caption vectors

—將上述(64, 4, 4, 512)與(64, 1, 1, 256)疊為(64, 4, 4, 768)

—使用stride為2*2的2D convolution layer，將其維度轉為(64, 4, 4, 512)再攤平為(64, 8192)

—使用mini-batch layer(參考第二點說明)將其維度轉為(64, 1)，即為result

輸出sigmoid(result)與result

(b) discriminator(fake)—判別模型產生圖片是否符合tag描述

架構同上，但輸入的image為模型產生，caption為正確的tag描述

(c) discriminator(wrong)—判別真實圖片是否符合tag描述

架構同上，但輸入的image為真實圖片，caption為錯誤的tag描述

(d) generator (輸入為隨機產生的noise以及正確的tag描述)

(以下layer皆用leaky relu作為activation function)

—輸入(64, 2400)的caption vectors並使用linear layer將其維度轉為(64, 256)

—輸入(64, 100)的noise並與上述(64, 256)疊加為(64, 356)

—使用linear layer將其維度轉為(64, 8192)

—reshape並搭配linear layer將其維度為(64, 4, 4, 512)

—多次使用stride為2*2得2D deconvolution layer，按不同output dimension將其維度接續轉為(64, 8, 8, 256)、(64, 16, 16, 128)、(64, 32, 32, 64)、(64, 64, 64, 3)

—此維度為(64, 64, 64, 3)的matrix即為result

輸出tanh(result/2 + 0.2)(使其數值介於[0.5, 1]之間)與result

(e) loss function

—discriminator loss(real): 輸入真實圖片與正確tag描述。輸出扣除1後加總取平方(least-square)

—discriminator loss(fake): 輸入模型產生的圖片與正確tag描述，輸出扣除0後加總取平方

—discriminator loss(wrong): 輸入真實圖片與錯誤tag描述，輸出扣除0後加總取平方

—generator loss: 取得discriminator loss(fake)輸出的result(未sigmoid的)，扣除1後加總取平方

上述loss加總+l2_loss(discriminator的參數平方，參考第二點說明)即discriminator total loss；

最後再分別最小化discriminator total loss以及generator loss來訓練模型。

(2) objective function for G

```
g_loss = tf.reduce_mean((disc_fake_image_logits - tf.ones_like(disc_fake_image)) ** 2)
```

g_loss = 加總(discriminator對假image與正確tag描述的評分)後取平方

(3) objective function for D

discriminator loss(real): 加總(discriminator對真image與正確tag描述的評分 -1)後取平方

discriminator loss(fake): 加總(discriminator對假image與正確tag描述的評分)後取平方

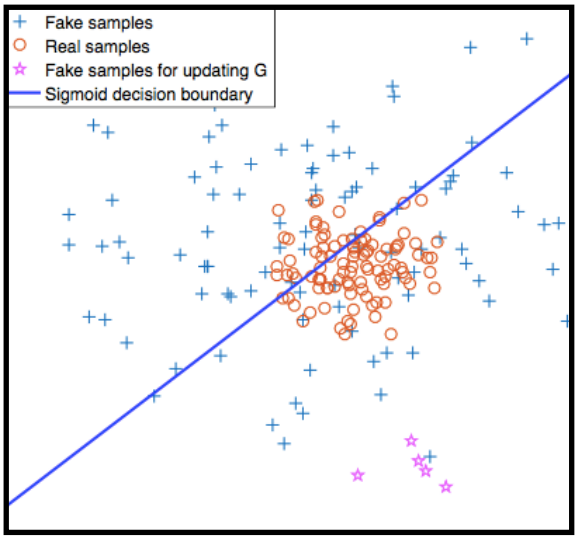
discriminator loss(wrong): 加總(discriminator對真image與錯tag描述的評分)後取平方

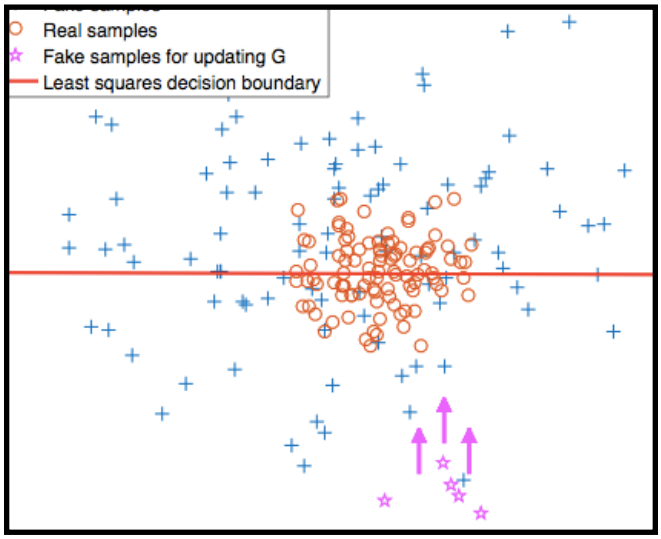
```
d_loss1 = tf.reduce_mean((disc_real_image_logits - tf.ones_like(disc_real_image)) ** 2)
d_loss2 = tf.reduce_mean((disc_wrong_image_logits - tf.zeros_like(disc_wrong_image)) ** 2)
d_loss3 = tf.reduce_mean((disc_fake_image_logits - tf.zeros_like(disc_fake_image)) ** 2)
```

2. How do you improve your performance

(1) Least-square loss

若使用sigmoid_cross_entropy來評量錯誤標準的話，會因為sigmoid的特性(只在中間浮動較大，其他皆貼近0與1)使得提供generator改進的資訊量很少(discriminator根本不在乎假圖片錯多少，他只要判斷有錯就好了，如下方左圖所示)，在generator來不及改進前，discriminator便已收斂完成。
而least-square多給了generator那些假圖片"錯多少"(如下方右圖的紫色箭頭)。使得generator可以更快速且有效的學習。





(a) 實作：原相減項加總後取平方

```
d_loss1 = tf.reduce_mean((disc_real_image_logits - tf.ones_like(disc_real_image)) ** 2)
```

(2) Mini-batch

由於普通的GAN在train到最後，將會有著mode collapse(generator發現一個A分佈可以有效提高分數，discriminator發現後generator躲到B分佈，再被發現後躲回A，來回跳動的現象)的問題。也因此到最後所有generator產生的圖片都會非常類似，為了解決這些問題，我們需要鼓勵模型產出的圖片要具有多樣性：透過增加給generator產生圖片彼此相似度的loss，讓模型避免產出同樣的圖片。

(a) 透過自己相減得到diffs，加總後取絕對值後得到abs_diffs

(b) abs_diffs取負號後得對數值，再以exponential為基數得真數值(minibatch_features)

```
diffs = tf.expand_dims(activation, 3) - tf.expand_dims(tf.transpose(activation, [1, 2, 0]), 0)
abs_diffs = tf.reduce_sum(tf.abs(diffs), reduction_indices=[2])
minibatch_features = tf.reduce_sum(tf.exp(-abs_diffs), reduction_indices=[2])
```

(3) Discriminator regularization

透過設定正規化，增加discriminator訓練的難度，同時避免其overfitting。

(a) 對於每一個discriminator的weight作加總並乘上0.01

(b) 將原有的discriminator的loss加上上述的regularization項

```
for d_weight in d_vars:
  regularizers += tf.nn.l2_loss(d_weight)
d_loss = tf.reduce_mean(d_loss + 0.01 * regularizers)
```

(4) 標籤篩選 & 固定文法結構

(a) 僅選取含有eyes或hair字串的tags及相對應的圖片

```
[t for t in tags if ('eyes' in t or 'hair' in t)]
```

(b) 將tag與tag間用and隔開。舉例：blue hair red eyes -> blue hair **and** red eyes
雖然沒有實驗此種方式進步幅度的大小，但我認為結構分明的句型更容易產生好的caption vector。

```
skipthoughts.encode(model, [' and '.join(filtered_tags)])
```

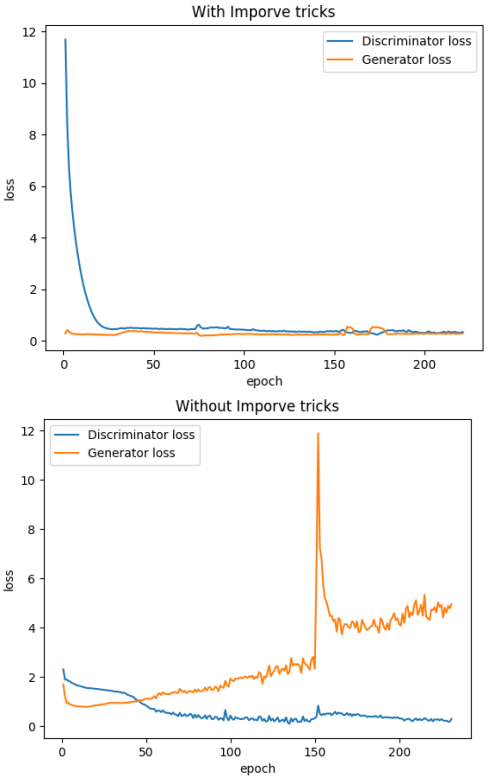
3. Experiment settings and observation

(1) Generator loss以及Discriminator loss

左圖表上下分別代表有使用improve technique以及沒有使用的loss比較。

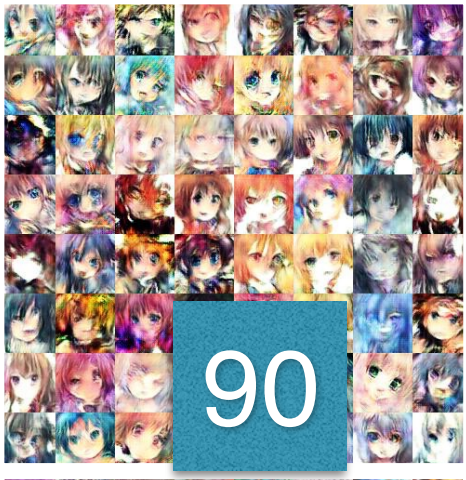
可以發現沒有使用improve techniques的generator loss從最開始幾個ep變開始變，而在約epoch-150的地方便急速的上升，造成mode collapse，與之相比有使用improve technique的模型從開始後便穩定下降，變動起伏小。

右圖片則為epoch-150的兩個模型(上方為使用了improve technique，下方則無)產生的假圖片：未使用improve technique確實造成了嚴重的mode collapse，圖片臉型幾乎一致。

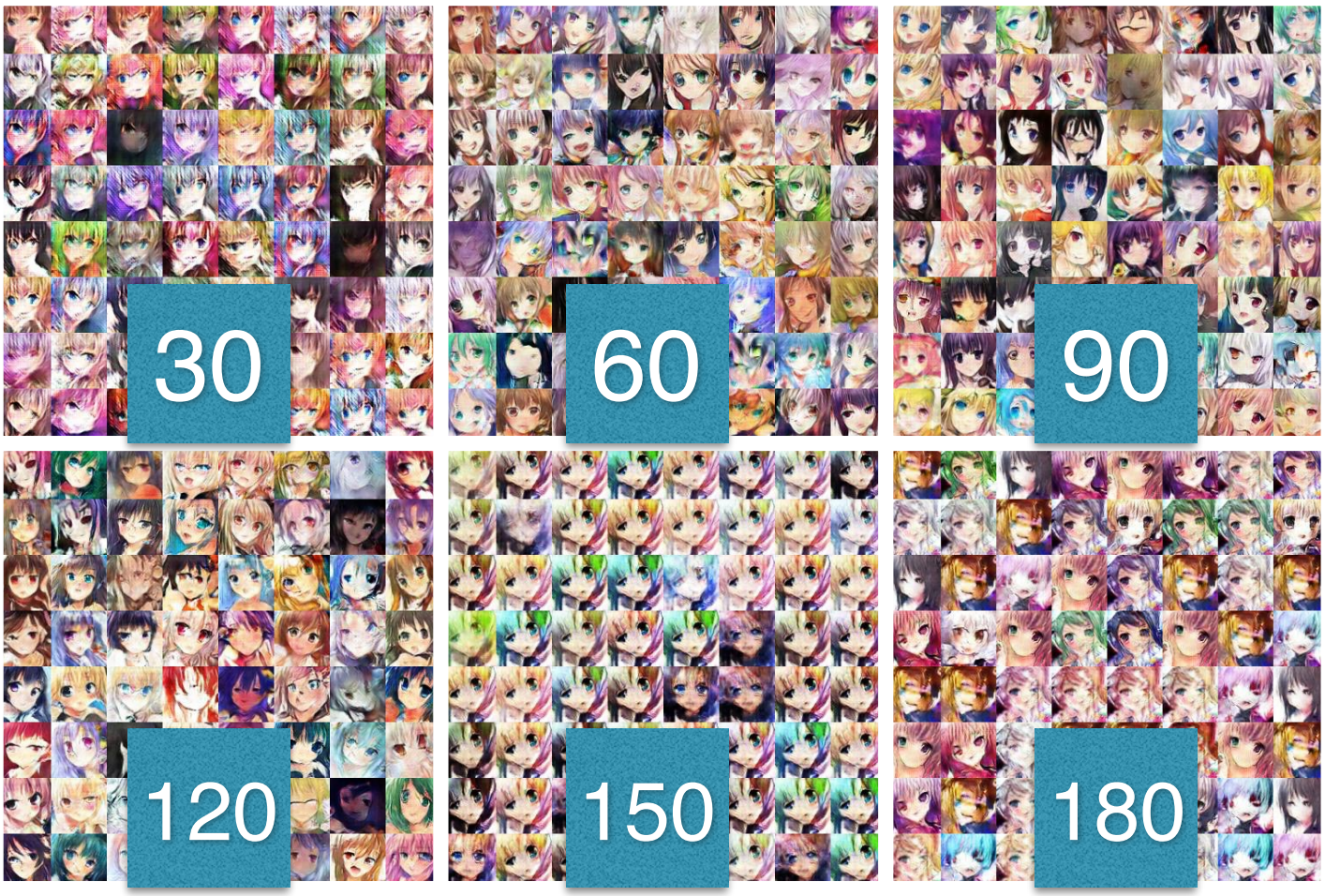


(2) 產出圖片比較

(使用improve technique)



(未使用improve technique)



由上述兩組圖可以看出使用了improve technique之後，儘管仍會有部分圖片臉型重複，但數量大幅降低，且其五官並非完全一致，而是有著明顯的差異。
也因為使用了improve technique(regularization, least-square)，discriminator在前期的訓練速度較慢，也因此同樣是30 epochs，未使用improve technique產出的圖片反而比較有模有樣；但是使用了improve technique的優點是顯而易見的：它幫助模型在長遠的表現更為出色。

若有更多時間，我會比較WGAN跟Least-square的差異，同時將圖片output size升為96*96，並搭配stack gan以產生更清楚的圖片，並使用標記質量較好的dataset。

4. Style Transfer

使用cyclegan將動畫人物的頭髮顏色一律轉為棕色。