b03901057 詹晉誠

ADLxMLDS HW4

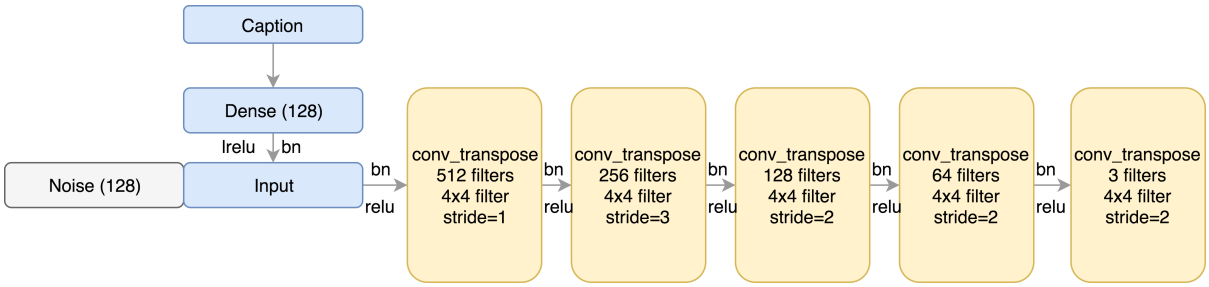
Platform: MacOS High Sierra
Python version: 3.6
Tensorflow version: 1.3
Skimage version: 0.13.1

• Model Description

1. Model Structure

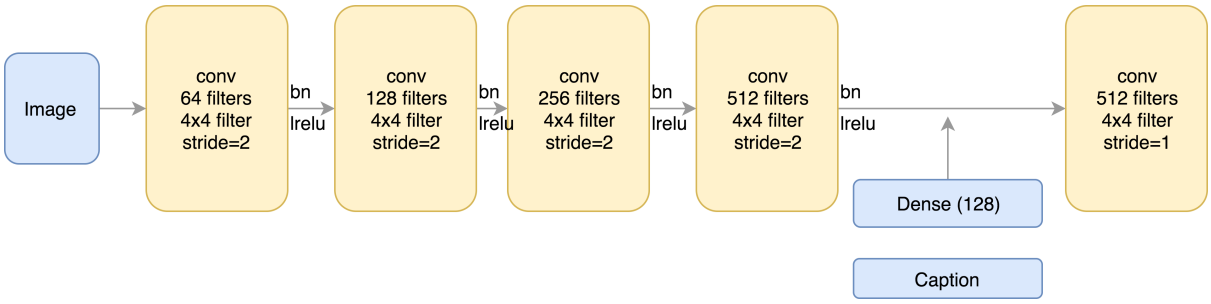
在我這次的作業中，model的架構主要是參考tex2image這篇論文中的model架構並加上自己的一些改進。以下會跟別針對generator以及discriminator的兩個部分來說明model的架構。

Generator的架構如下所示：



圖中括號中的數字（如dense(128)）代表那層layer或是vector的大小，箭頭旁的bn代表batch-normalization, relu代表relu activation，lrelu 代表leak relu activation。首先caption會先經過一個大小為128的dense layer將caption vector做embedding，接著會與noise接起來並作為CNN的input。之後會經過五層的de-convolution layer把原本只有一個pixel大小的input變成96x96x32的彩色圖像。每層convolution layer的參數都寫在圖中。

Discriminator的架構如下所示：



Discriminator的input是 一張image，然後會經過四層的convolution layers，每層的filter數量分別是從64以二為倍數增加，然後每次的stride是2，因此每過一層layer，

image size就會減半。第四的conv. layer的output會與caption的embedding (128維）接起來，然後最後在經過最後一層的convolution layer。

此外，在這次作業中Caption都會先經過skip-thought vector 處理過變成一個128維的vector，再丟進generator/discriminator中。skip-thought vector 我是使用tensorflow gitbhub是的pretrain model來處理的。

2. Generator objective function

這份作業中我使用的generator loss有別於一般的DCGAN，我有進行一些修改讓產生的圖更像真實的，以及讓產生的image更像文字所述。仔細來說，我的generator loss是由下列三項構成：

- discriminator看到generator產生的圖片與正確的caption後作出的決定，與1的cross entropy，這個就是傳統DCGAN中generator 的loss。
- discriminator看到正確的image 以及正確的caption後作出的決定與discriminator看到generator 產生的image與正確的caption後作出的決定的差值的L2 norm。加入這項的目的是要讓generator產生的圖更符合文字描述。
- generator產生的圖與真正的圖差值的L1 norm。加入這項的目的是要讓產生的圖更像真實的圖。

最後，這三個loss整合的方式是用weighted sum，上述三者的weight分別是1, 100, 以及50。這個加總後的loss即是generator loss。

3. Discriminator objective function

Discriminator的loss我則是參考助教投影片中所建議的方式設計，詳細來說Discriminator的loss包含下列四項：

- discriminator看到正確的image 以及正確的caption後作出的決定與1的cross entropy，不過我在這邊有做soft label（將1降低成0.9，0還是保留成0），這樣做可以讓GAN train起來更穩定。
- discriminator看到錯誤的image 以及正確的caption後作出的決定與0的cross entropy
- discriminator看到generator產生的image 以及正確的caption後作出的決定與0的cross entropy
- discriminator看到正確的image 以及錯誤的caption後作出的決定與0的cross entropy

最後這四個loss會被直接加起來，成為discriminator的loss。

- Performance improvement

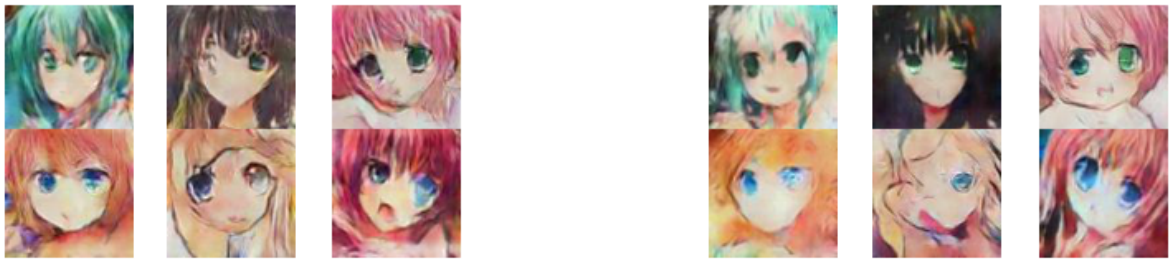
除了前述關於generator loss以及discriminator loss方面的改進外，我這

次的作業中我改進了以下的部分：

1. Noise 的取樣方式：

傳統的noise 取樣方式最常見的就從uniform distribution中做sample，但是用這種方式取出的sample常常出現極端值，也就是說常常sample到不常發生的情況，進而讓產生出的image品質變差。我參考了“Sampling Generative Networks”這篇論文的方式，將noise的distribution改為Gaussian。因為Gaussian distribution中越偏離極端值的發生機率會越低，因此sample到的noise會是比較常見的情況，圖片的品質也會比較好。以下是用uniform distribution跟Gaussian distribution兩種方式得到的結果比較：

Gaussian　　Uniform

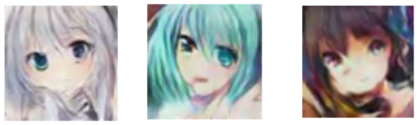


比較兩者，可以發現整體而言Gaussian會比Uniform好一些些。

2. Training 時image size的大小：

在這份作業中最常遇見一個困擾是產生的圖會模糊。會讓圖模糊的原因可能有很多，但是一個可以讓圖變的更加清楚的方式在training時不要將圖縮小成64x64，而是使用96x96的原圖。如果在training時就將圖縮小成64x64，那generator就會看不到很多細節，再加上generator本身沒有辦法學到那麼好，產生出來的圖就會模糊。如果在training時讓generator看大圖，generator就有機會學到一些比較細節的東西，即使最後有些還是沒學起來，產生的圖自然就會比較清楚一些。以下是使用64x64的image做training以及使用96x96的做training結果的比較：

96x96



64x64

從結果可以發現96x96的圖片細節比較多，也比較接近真實的圖。

• Experiment settings and observations

本次實驗使用的參數設定如下所示：

- batch size = 64, caption embedding dimension = 2400, learning rate = 2e-4
- optimizer: Adam with beta1 = 0.5

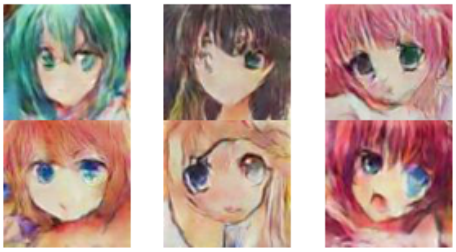
其餘的參數設定已於前述，此處便不在贅述。

這次的實驗我有改變一些參數，並觀察其對結果的影響，以下是我的觀察：

- Noise dimension的大小

我發現noise dimension如果大一些，產生出的圖可以比較多樣化一些，反之，如果太小，產生出來的圖會幾乎都一樣。但是如果noise dimension太大（例如比caption的dimension大）的時候，產生的圖反而會變得比較不好，例如比較模糊或是顏色比較不正確。這樣的結果是合理的，因為當noise的dimension太大時，caption對於輸出結果的影響可能會就比noise的影響還要小了，因此產生出的圖就會比較不好。以下是一些結果的比較

Small noise dimension



Large noise dimension

- Mode collapse

在我的實作中，我發現mode collapse的問題很嚴重，不只是train到後來會collapse，其實在training 一開始時就會發生collapse的問題了，如下圖所示：

關於這部分，我認為如果採用WGAN等等比較好的loss function也許就可以避免mode collapse的問題了，不過礙於時間因素，我沒有實作WGAN等等比較好的演算法，因此目前也只能推測而已。

- Batch-normalization testing mode

Batch-normalization有分成training mode 跟testing mode，理論上在testing時應該要讓batch-

normalization 在testing mode產生的圖片才會正常，不過有趣的是我發現在test階段batch-normalization也要設為training mode產生的圖片才是對的，不然會產生很暗的圖（雖然是有一些輪廓沒錯）

- **Bonus: CycleGAN**

在這次的bonus 中，我嘗試使用CycleGAN來達到style transfer的效果。CycleGAN的概念是訓練一個generator把source domain的東西轉成target domain 的圖（X -> Y)，然後discriminator要負責判斷generator產生出的圖是不是屬於target domain的圖。

這次的作業中我實作的CycleGAN是參考Github上的實作（https://github.com/xhujoy/CycleGAN-tensorflow）。Generator 的架構是使用resnet的設定，discriminator的架構基本上跟這次作業中使用的架構類似。

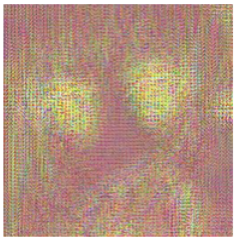
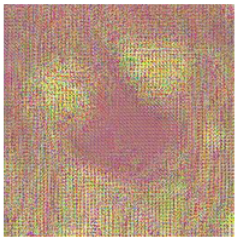
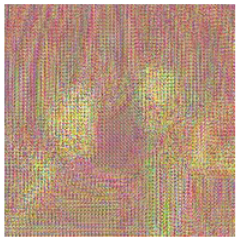
Generator分別會將real X翻成fake Y, 然後將fake Y翻回fake X，同時也會將real Y 翻成fake X，然後將fake X翻回fake Y。而generator的loss就是fake Y和1的cross entropy、fake X和1的cross entropy、翻回X的fake_X'和real X的L1 norm再加上翻回Y的fake_Y'和real Y的loss的總和。

Discriminator的loss也可以分為X, Y的兩個discriminator，X的discriminator的loss是real X跟1的cross entropy與fake A sample與0的cross entropy的平均。Y discriminator的loss則是X的對稱。最後將兩個loss加起來就是discriminator的loss。

我的CycleGAN是train在作業中的dataset與ukiyoe兩個datasets上。ukiyoe dataset的一些sample如下所示：



而將ukiyoe transfer到動漫人物的結果如下：

可以發現結果是還有一些動漫人物的輪廓，但是沒有 ukiyoe 的風格。我認為目前的問題在於 training 時間還不夠長，因此 generator 還沒學好，但是礙於時間因素，我無法在繼續 training。