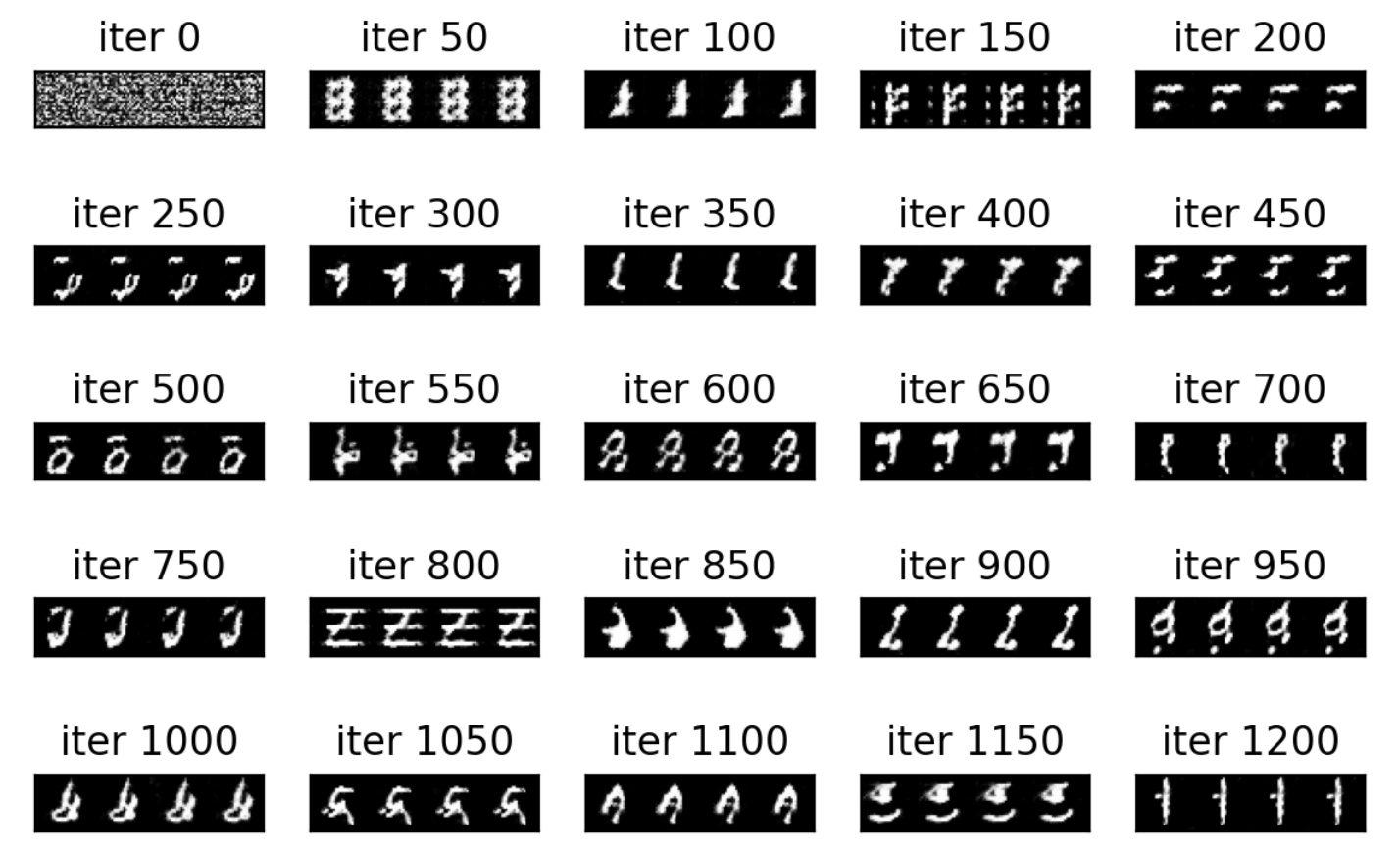
# 使用Jetson Nano生成手寫數字

# 前言

遇到生成的問題一定是要找現在最流行的GAN啦，GAN，那今天我們除了要讓大家了解GAN是什麼之外，我們也要來挑戰Jetson Nano的極限，上次已經用Nano跑貓狗分類發現對於它來說已經相當的艱辛了，這次要給它更艱難的任務，我們要讓電腦學習如何生成手寫數字的圖片！

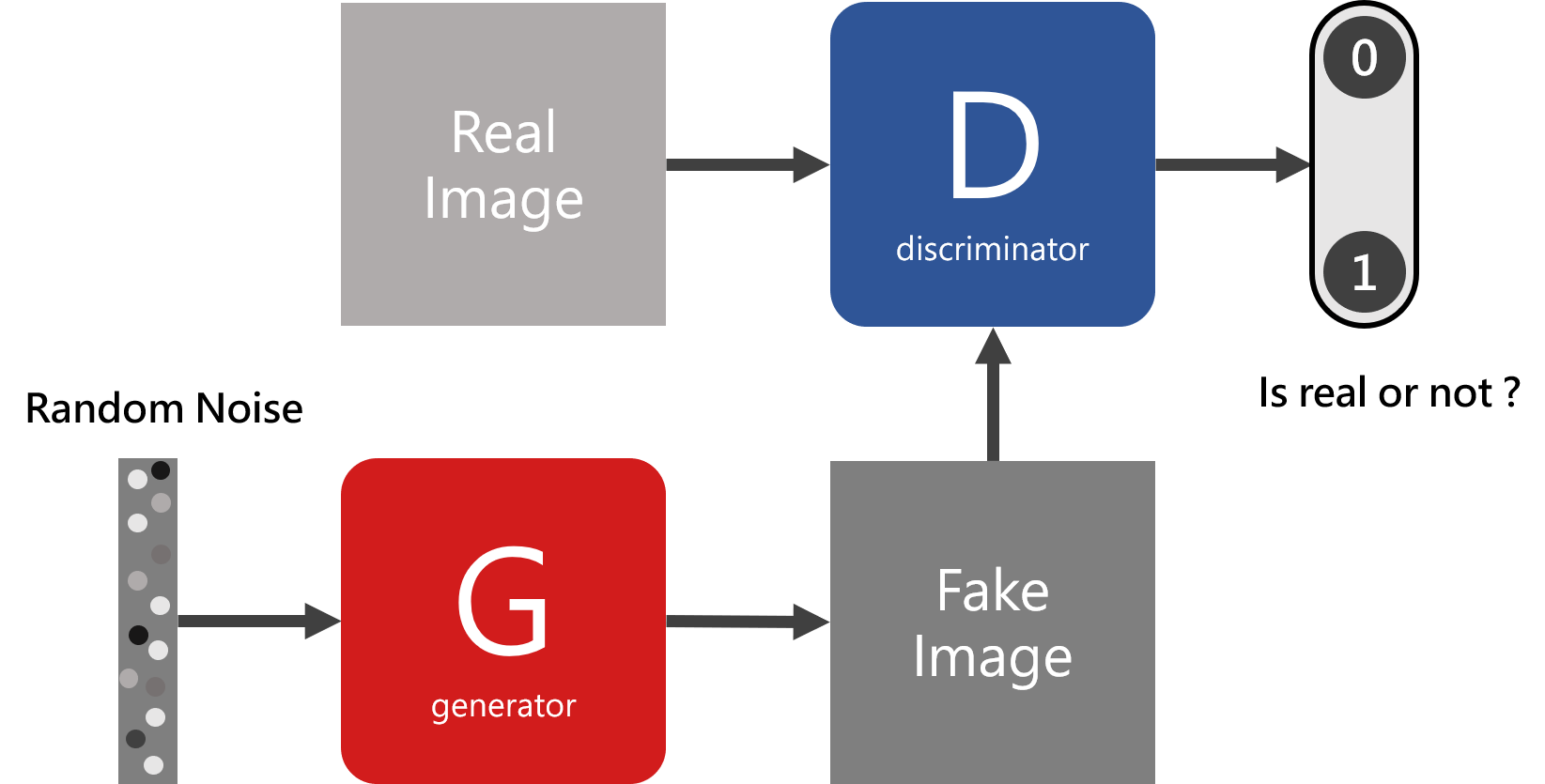


# GAN 基本觀念

GAN 是生成對抗網路 (generative adversarial network, GAN) 是一個相當有名的神經網路模型也是一個人見人怕的模型，原因是因為它非常的難訓練、耗費的時間也很久，它常常被用於深度學習中的生成任務，不管是圖像還是聲音都可以。

|  |
| --- |
| Deep Fake |
| 讓臉書忍不住出手封鎖的Deepfake 影片，是什麼技術？ - 台灣人工智慧學校  Image Source : <https://aiacademy.tw/what-is-deepfake/> |
| CycleGAN |
| Image Source : <https://github.com/junyanz/CycleGAN> |

核心觀念可以這樣想像，GAN就像「收藏家」與「畫假畫的人」，G是畫假畫的，D是收藏家；一開始畫假畫的人技術還不成熟，所以一眼就被收藏家發現問題，所以G就回去苦練畫功慢慢開始能騙過D，這時候D被告知買的都是假畫所以他也開始進步越來越能發現假畫的瑕疵，就這樣反覆的交手成長，等到最後互相僵持不下的時候就達到我們的目的了 - 獲得一個很會畫假畫的G，或者稱為很會生成圖片的G。



在GAN當中，我們可以輸入一組 雜訊 (Noise) 或稱 潛在空間 (Latent Space)，然後生成器 (Generator) 會將那組雜訊轉換成圖片，再經由鑑別器 (Discriminator) 分辨是否是真實的圖片，鑑別器將會輸出0或1，其中 0 是 fake，1 是 real 。

# DCGAN 架構介紹

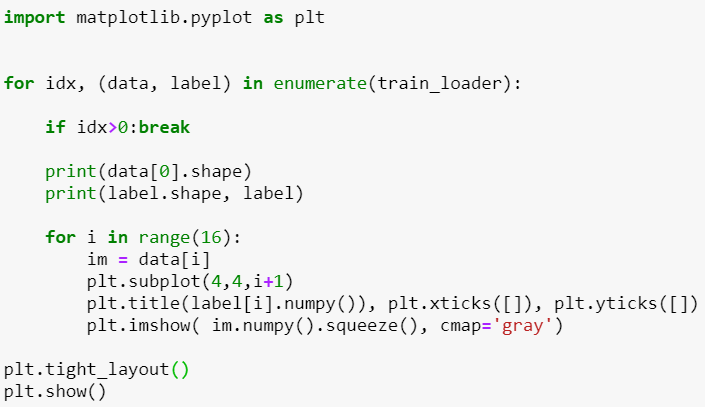
一般的GAN都是 FC ( linear )，需要將圖片reshape成一維做訓練，以 mnist 來看的話就是 1\*28\*28 變成 1\*784，而今天我們要介紹的是DCGAN，是利用捲積的方式來建構生成器與鑑別器，它跟一般的CNN有些許不同，作者有列出幾個要改變的點：

1. 取消所有pooling層。G使用轉置卷積（transposed convolutional layer）進行上取樣而D用加入stride的卷積代替pooling。
2. 在 D 和 G 中均使用 batch normalization。
3. 去掉 FC 層，使網路變為 全卷積網路 (Conv2D)。
4. G使用 ReLU 作為激勵函數，最後一層須使用 tanh。
5. D使用 LeakyReLU 作為激勵函數。

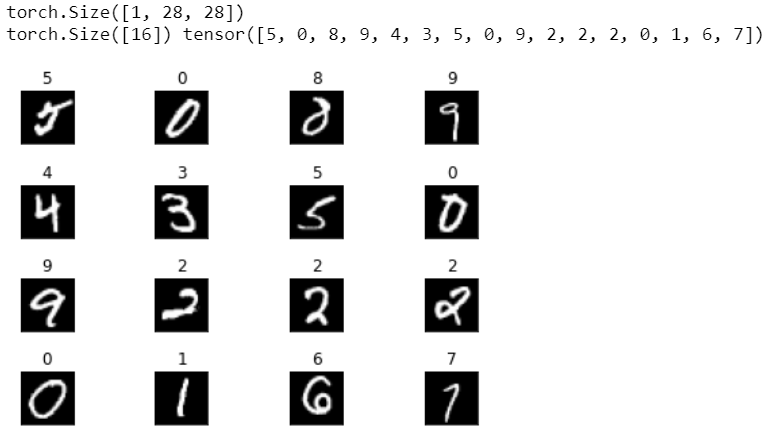
等等實作程式的時候可以再回來觀察是不是都符合需求了，那基礎觀念的部分先帶到這我們直接開始寫code吧！

# 準備數據集

今天我們要嘗試生成手寫數字，最常使用的就是MNIST，由於它太常用了可以直接在torch中就下載到，在torchvision.datasets裡面，由於它已經是數據集了所以可以先宣告transform套入，接著就可以打包進DataLoader裡面進行批次訓練



可以注意到圖片已經是灰階圖並且大小為 ( 1, 28, 28)



# 建構鑑別器

鑑別器的架構就是CNN只差要取消全連階層的部分，所以要想辦法將28\*28的圖片捲積成 1\*1輸出

最後因為輸出的label是介於 [ 0 , 1 ] 所以最後要透過Sigmoid來收斂。這邊直接使用官方提供的架構可以發現跟以往不同的地方是利用nn.Sequential將所有層在initial的時候就連接起來，你會發現更以前相比簡潔很多；此外我還使用了torchsummary的函式庫視覺化網路架構：

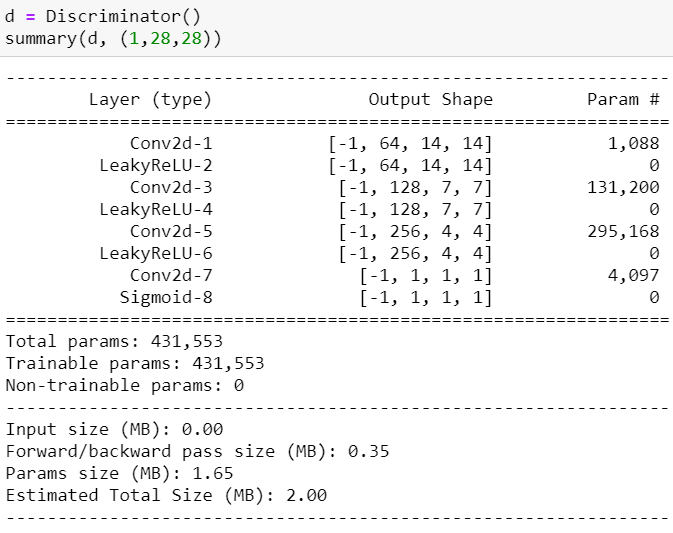


使用torchsummary之前需要先安裝套件

|  |
| --- |
| !pip3 install torchsummary |

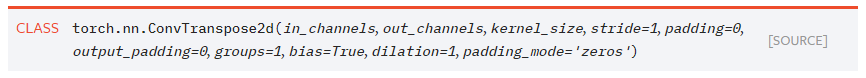
然後記得要先導入

|  |
| --- |
| from torchsummary import summary |



# 建構生成器

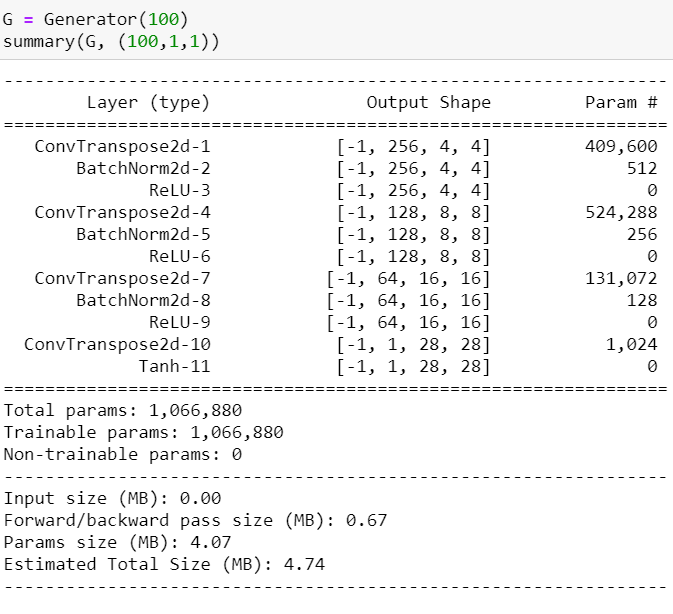
生成器一樣要全用捲積的方式，但是正如前面所說我們要將一組雜訊轉換成一張圖片，用一般的捲積方式只會越來越小，所以我們必須使用反捲積 ( ConvTranspose2d )，可以想像就是把捲積反過來操作就好，大小一樣要自己算記得最後輸出應該跟輸入圖片一樣大，公式如下：



建置生成器的架構與CNN顛倒，Kernel的數量要從大到小，然後我們不使用MaxPooling改用ConvTranspose，每層都要有BatchNorm並且除了最後一層的激勵函數是Tanh之外其他都是ReLU。



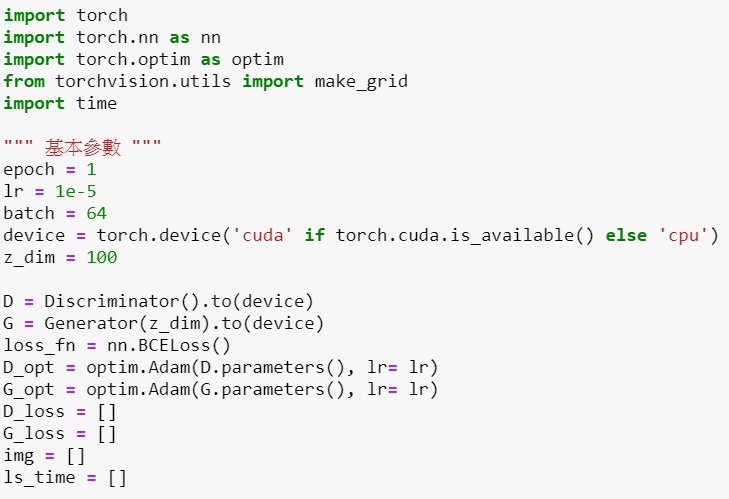
我們一樣使用 torchsummary 來可視化網路架構：



# 開始訓練ＧＡＮ

其實訓練GAN的方法不難，只要想像成是訓練兩個神經網路即可。GAN的訓練方式較常見的會先訓練鑑別器，讓鑑別器具有一定的判斷能力後再開始訓練生成器；首先一樣先建立基本的參數。

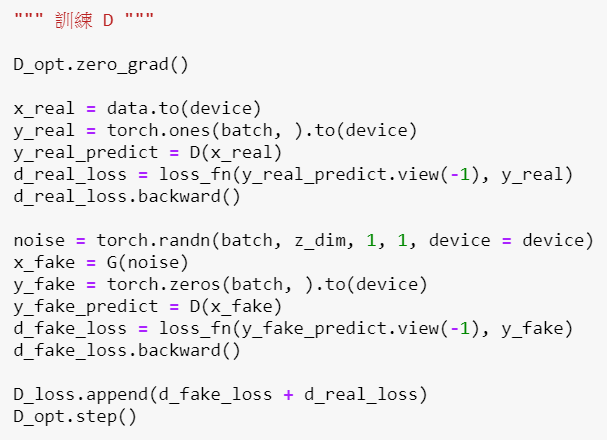
由於GAN的訓練較複雜，需要的迭代次數也越高，這次我們直接訓練個100次看成效如何，此外這次的損失函數都用BCELoss 因為主要是二元分類問題，對於想要了解更深入的可以去看台大李弘毅教授GAN的教學影片。



## 訓練Discriminator

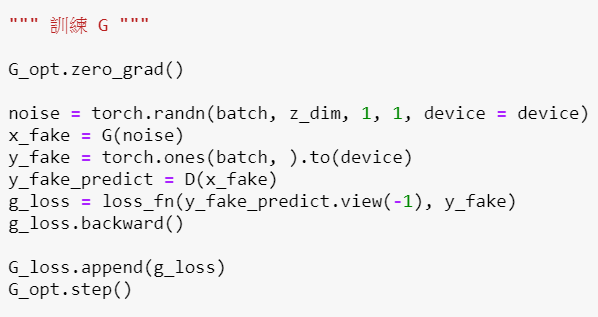
訓練鑑別器的目的是讓自己能分辨真實圖片跟造假圖片，首先，先將優化器的梯度清除避免重複計算，接者將真實圖片丟進GPU並給予標籤 ( 1 )，將其丟進D去預測結果並計算Loss，最後丟進倒傳遞中；假的圖片也是一樣的部分，差別只在於假圖片需要由生成器生成出來，所以要先定義一組雜訊並丟入生成器產生圖片，因為要讓鑑別器知道這是假的所以要給予標籤 ( 0 )，接著一樣將假圖片丟進鑑別器並計算Loss。經過反覆的訓練鑑別器就越來越能判斷真假照片了，但是同時生成器也會訓練，所以當鑑別器越強的時候，生成器產生的圖片也會越好！

這個部分要注意的地方是經過鑑別器訓練出來的答案維度是 [ batch ,1, 1, 1]，所以需要過一個view(-1)來將維度便形成 [batch, ] 一維大小。

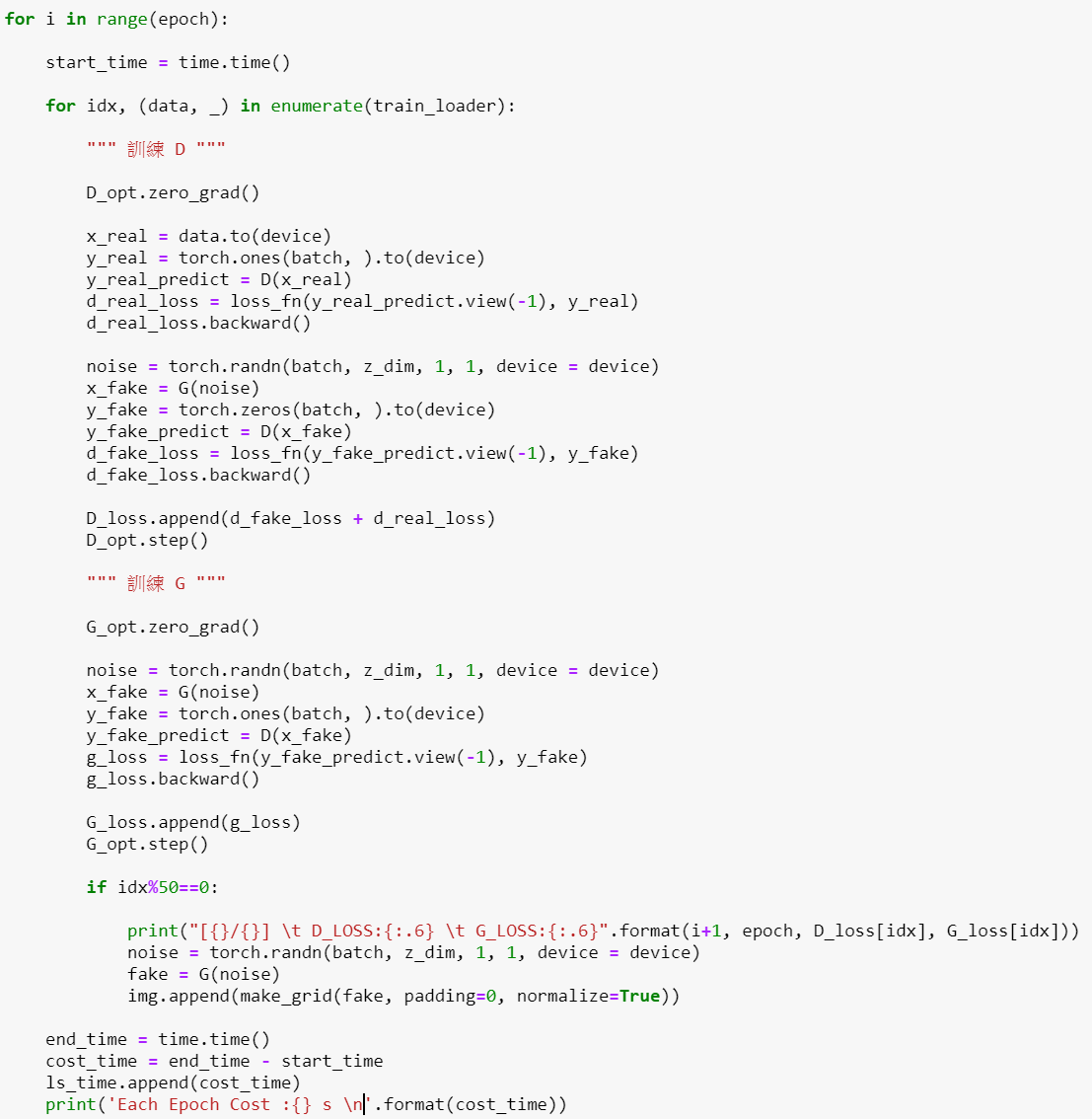


## 訓練生成器

如果已經看懂鑑別器的訓練方式了，那生成器對你來說就不是個問題！我們一樣要先將梯度歸零，亂數產生雜訊並且透過生成器產生圖片，這邊要注意的是因為生成器的目標是要騙過鑑別器，所以我們要給予真實的標籤 ( 1 )，這樣做的目的是，生成器一開始產生的圖片很差所以鑑別器得出的結果都接近於0，這時候如果標籤是1的話，計算Loss數值將會很大，神經網路就會知道這樣不是我們想要的，它會再想辦法生成更好的圖片讓Loss越來越小。

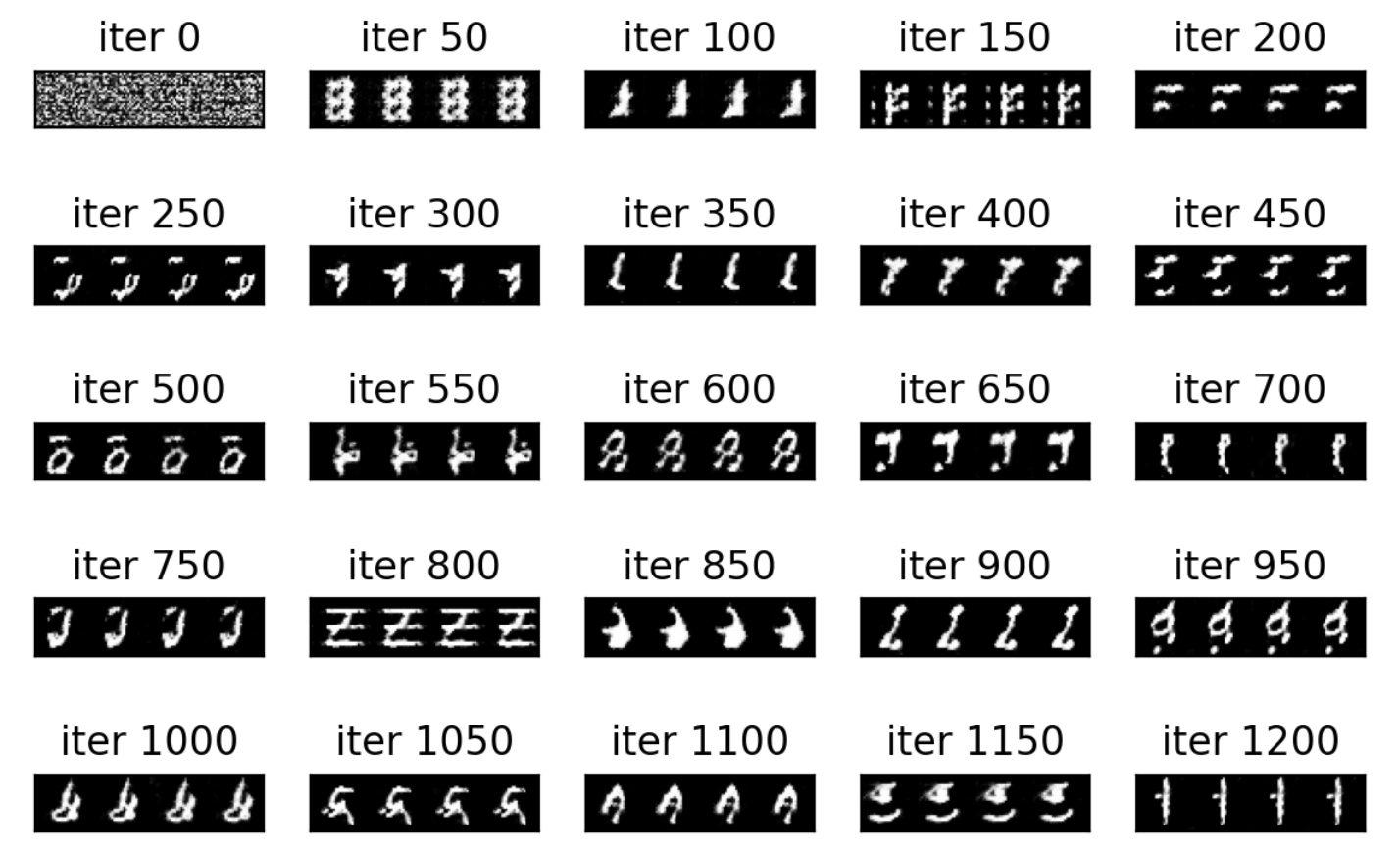


所以完整的訓練如下，先訓練Ｄ再訓練Ｇ，有的人會讓D先訓練個幾次再訓練G，聽說效率比較高；而我為了測試jetson Nano的速度所以每一個epoch都有紀錄時間，就連我自己的GPU( 1080 ) 都會爆顯存更不用說用Jetson Nano來跑了！如果遇到顯存爆炸的問題可以嘗試先將batch size調小。



# 成果

你可以注意到它慢慢能轉換成數字了，轉換的速度其實很快但要更細節的紋路就需要更多時間來訓練。



# 訓練時間比較

使用GPU 1080 訓練，每一個epoch約耗時310秒左右；而JetsonNano開啟cuda來跑大概每一個epoch約耗時1030秒左右。所以其實要在Nano上面運行GAN也是可行的，速度還算可以。

|  |  |
| --- | --- |
| 桌上型電腦 | Jetson Nano |
|  |  |

# 結語

這篇教大家如何建構DGAN，接下來我們將會在Jetson Nano上嚐試更多GAN相關的訓練，下一篇將讓電腦玩填色遊戲~