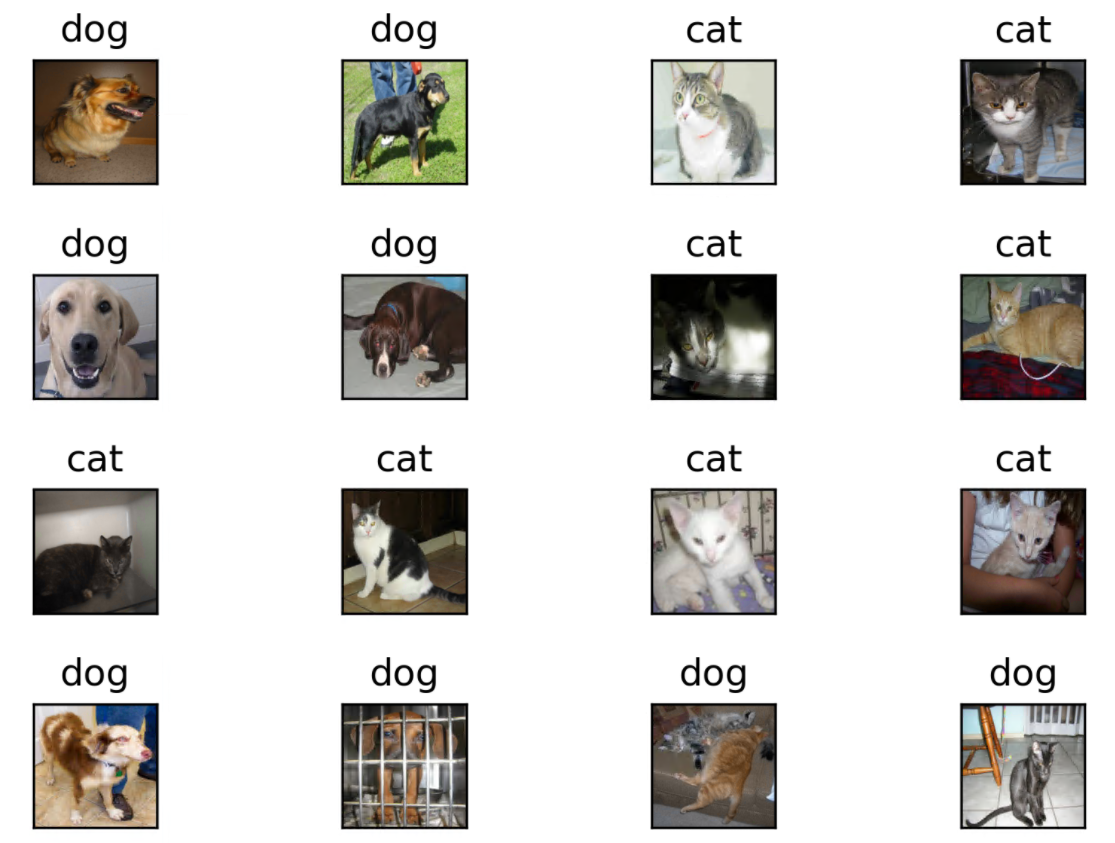
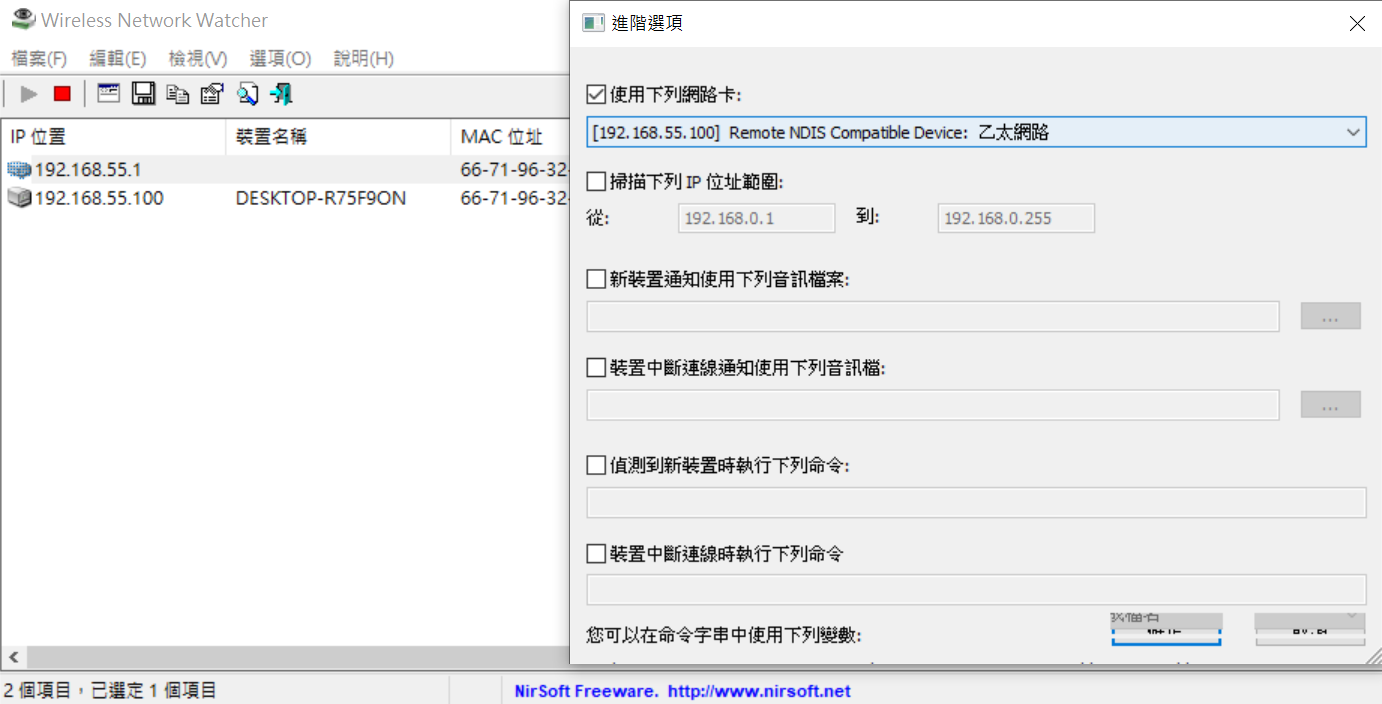
# 貓狗分類器，透過PyTorch建置



本篇文章將使用Jetson Nano做出一個非常簡單的貓狗分類器，其中用到PyTorch的ImageFolder做數據集並且用DataLoader將數據集載入的技術，並且學會用自己建置的CNN來訓練，最後取出測試圖片來做預測 ( 如上圖 )。

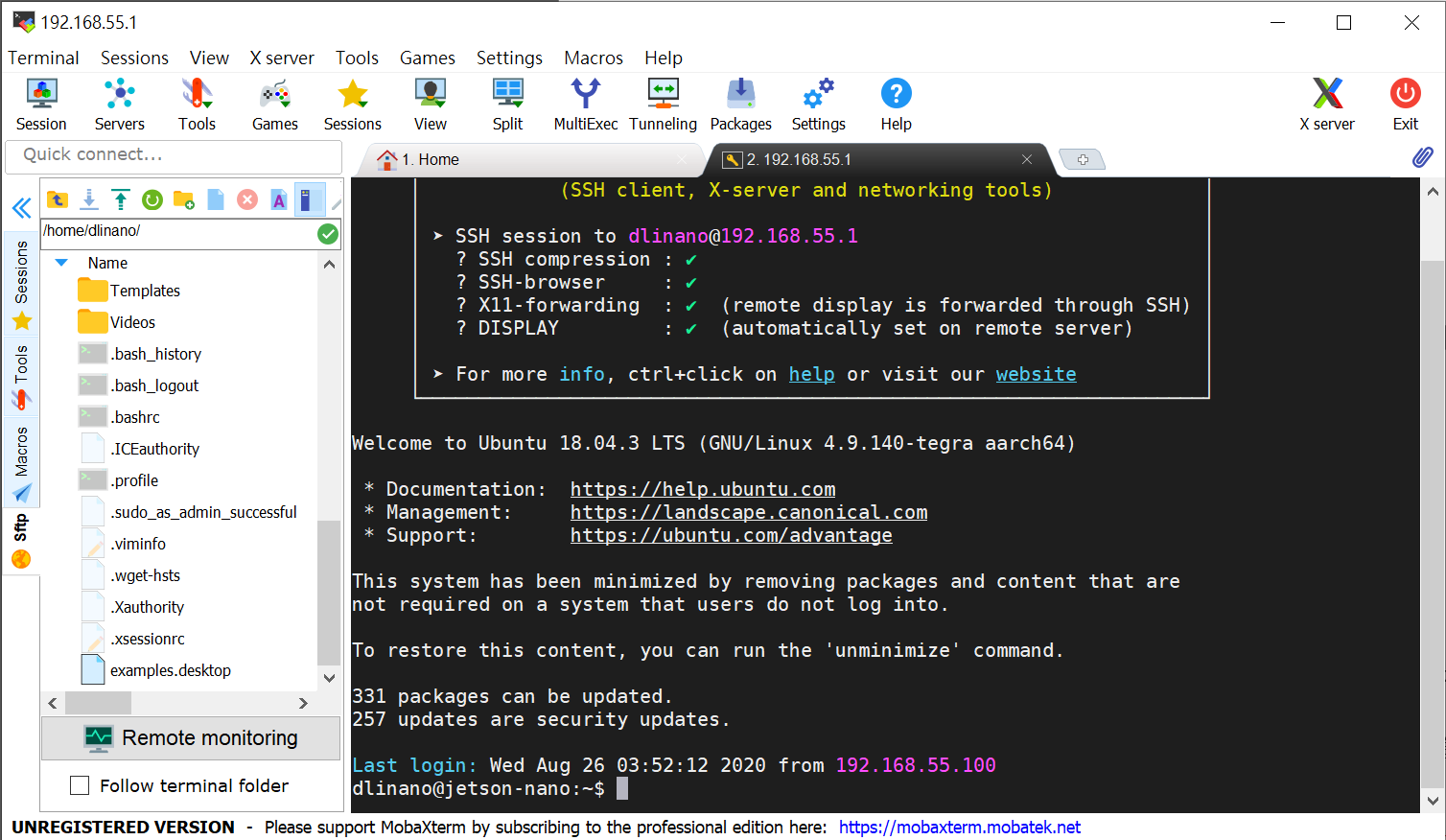
# Jetson Nano遠端、確認環境

今天我們要在Jetson nano這塊開發版上運作，而我們公司已經有許多Jetson nano的相關介紹、應用，所以就不多作介紹了，連結放在這裡給大家參考<https://blog.cavedu.com/tag/jetson-nano/>，遠端連線的部份我們使用Wireless Network Watcher查看Nano的IP位址，在選項 > 進階選項 > 使用下列網路卡 > 選擇乙太網路，點擊確定進行掃描就可以找到Jetson nano的虛擬IP

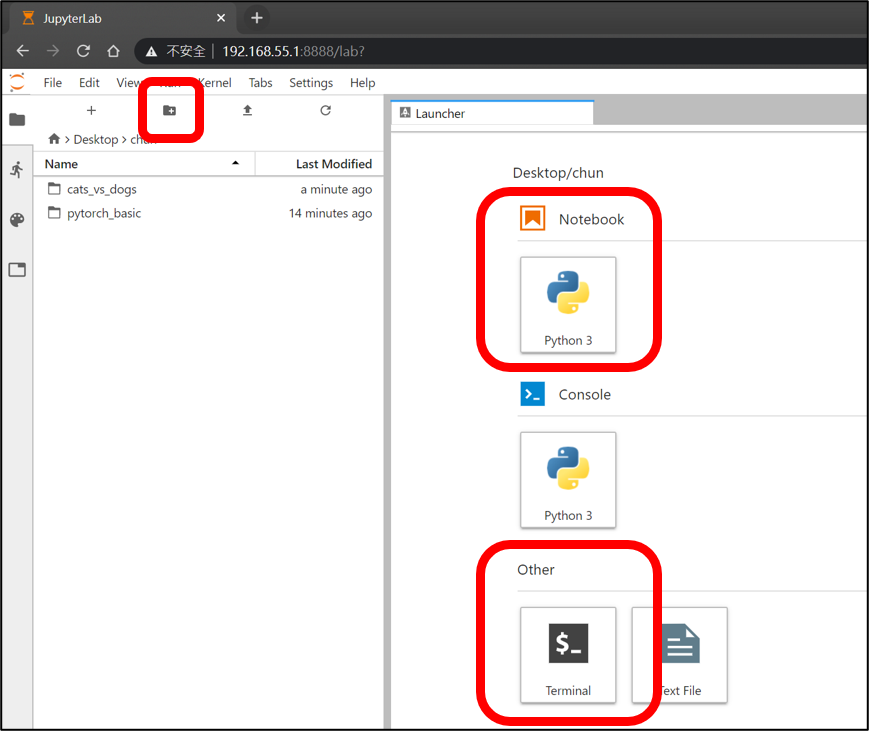


接著可使用MobaXterm或Jupyter Notebook進行內網的遠端，兩者都有檔案系統可以直接進行檔案傳輸，相當好用！在MobaXterm的部份，直接上網安裝點擊Session輸入IP位址、使用者帳號、密碼即可使用；而Jupyter Notebook的部份，Jetson Nano已經有幫我們設定好Notebook的遠端功能，所以只要在PC端開啟瀏覽器輸入http://{你的JetsonNano的IP位址}:8888/，輸入密碼即可登入；本次範例將使用Jupyter Notebook來運作。

以下為使用MobaXterm操作的介面：



以下為使用Jupyter Notebook操作的介面，建議先新增一個資料夾當作工作區，新增後可以開始一個Terminal跟Jupyter Notebook；Termial用來安裝套件、Jupyter用來撰寫與執行程式：

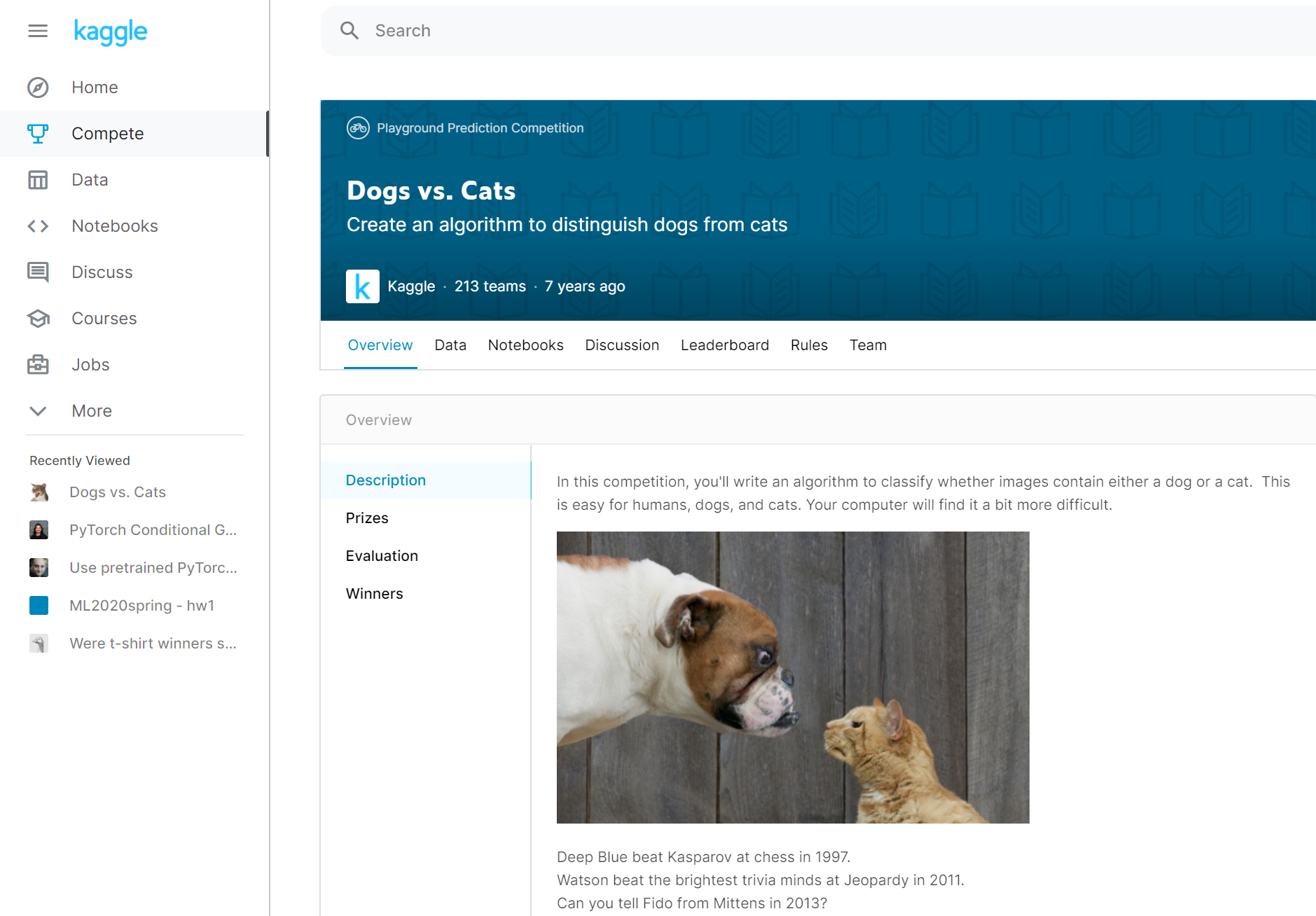


# Kaggle

## 介紹

今天使用的數據集會從Kaggle上面下載下來，這是一個數據建模和數據分析競賽平台，每年都會辦機器學習、深度學習的比賽，也有勵志成為機器學習工程師的人會反覆地在上面刷各種數據集取得好成績。

數據下載的連結<https://www.kaggle.com/c/dogs-vs-cats/overview/description>



關於Kaggle數據集下載的方法有兩種：第一種是直接下載，第二種則是透過Kaggle提供的API進行下載，今天也教大家如何使用API下載。

## 使用API下載Kaggle數據集

### 第一步、安裝Kaggle API

可在Jupyter Notebook中執行：

|  |
| --- |
| !pip3 install -U -q kaggle |

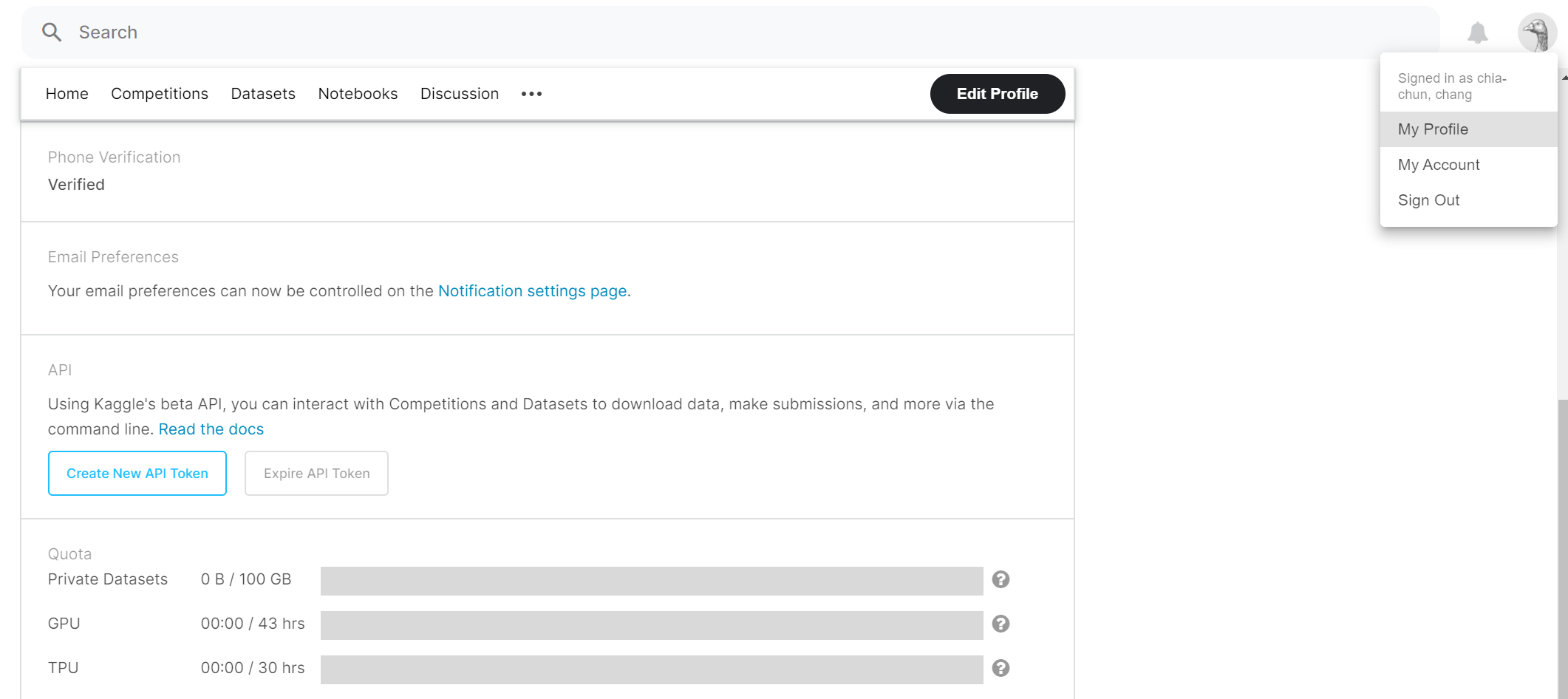
也可以在Terminal中執行

|  |
| --- |
| pip3 install -U -q kaggle |

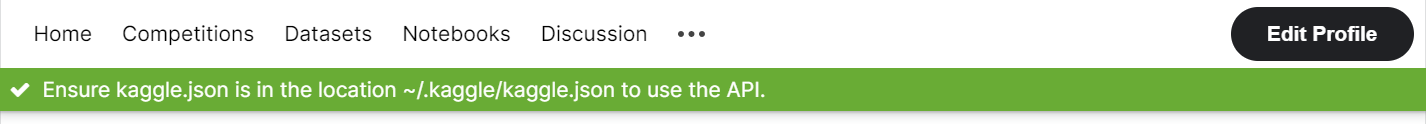
兩者差別只在於Jupyter中要使用命令需要加上驚嘆號來與程式碼作區別！

### 第二步、取得認證JSON檔

右上方大頭貼 > My Account > 下拉至API欄位並點擊 Create New API Token就可以取得認證的JSON檔



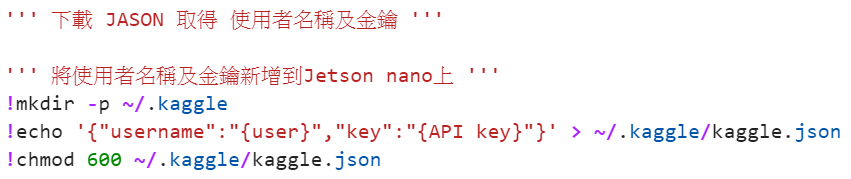
注意存放位置！



這邊下載後會提醒你要放到使用裝置的 ~/.kaggle 位置，等等新增到Jetson Nano中要特別注意！

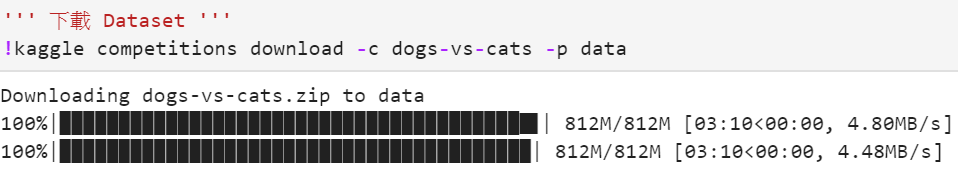
### 第三步、將認證檔新增到Jetson Nano

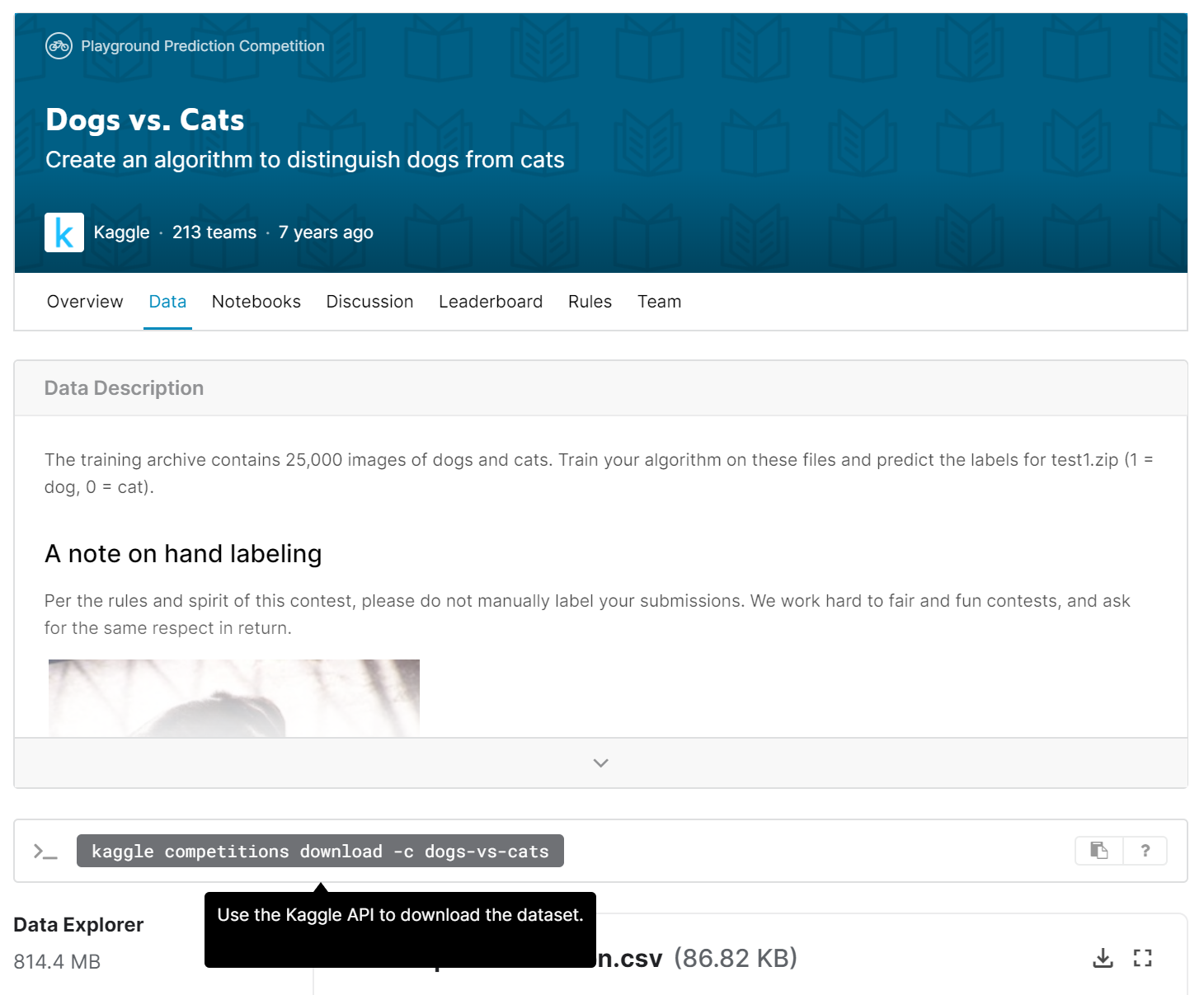
輸入以下程式碼，將 {usr}、{API key} 修改成自己的名稱、金鑰即可，執行之後Jetson Nano中就有kaggle中你的認證檔了，接下來直接進行下載。



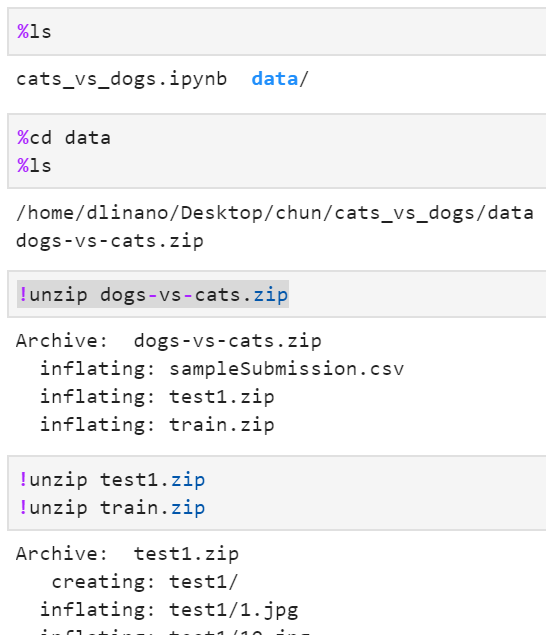
### 第四步、下載dataset到特定資料夾

這段的程式碼是從Kaggle數據集的網頁中Data的資訊中可以找到，其中 -p的引數是下載到指定目錄：



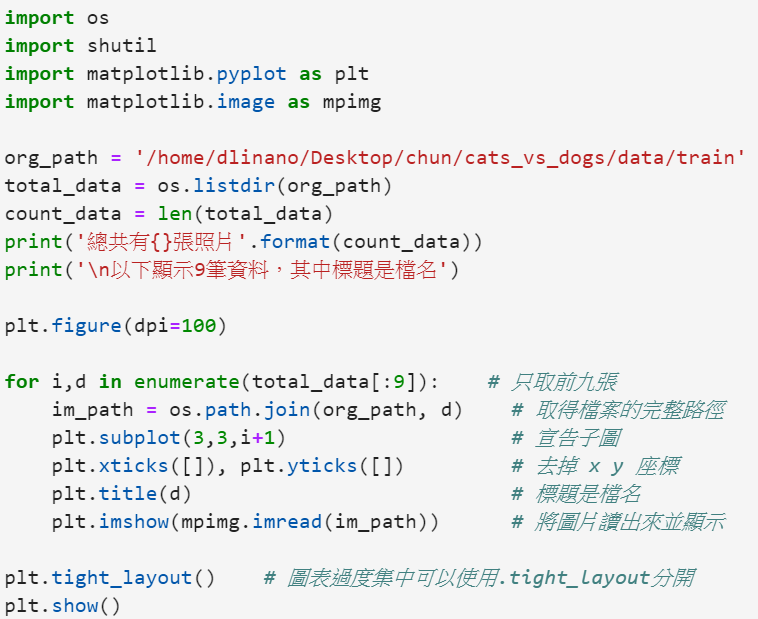


### 第五步、確認資料並解壓縮



# 資料處理

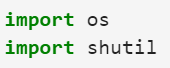
提供的資料是train跟test1兩個資料夾，在train這個資料夾中總共有25000筆資料，我們將前九張照片顯示出來看看

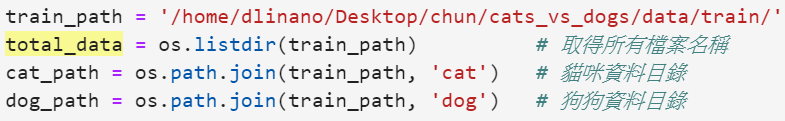




可以注意到資料的名稱是 {label}.{id}.jpg，我們的目標是要將貓跟狗兩種不同的資料分到不同的資料夾當中，所以們我會先新建一個屬於貓跟狗的資料夾，在透過檔名來做分類。

先導入函式庫以及宣告基本的目錄位址：

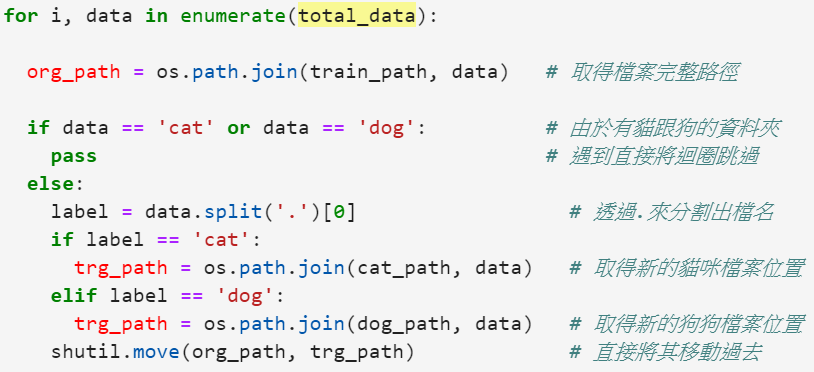




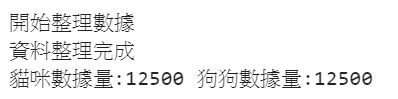
確認目錄是否存在，如果沒有就創建一個：



接下來就是主要整理的程式碼了：



整理完可以發現貓跟狗都已經到各自的資料夾了，各有12500張照片：



# 透過torch製作數據集

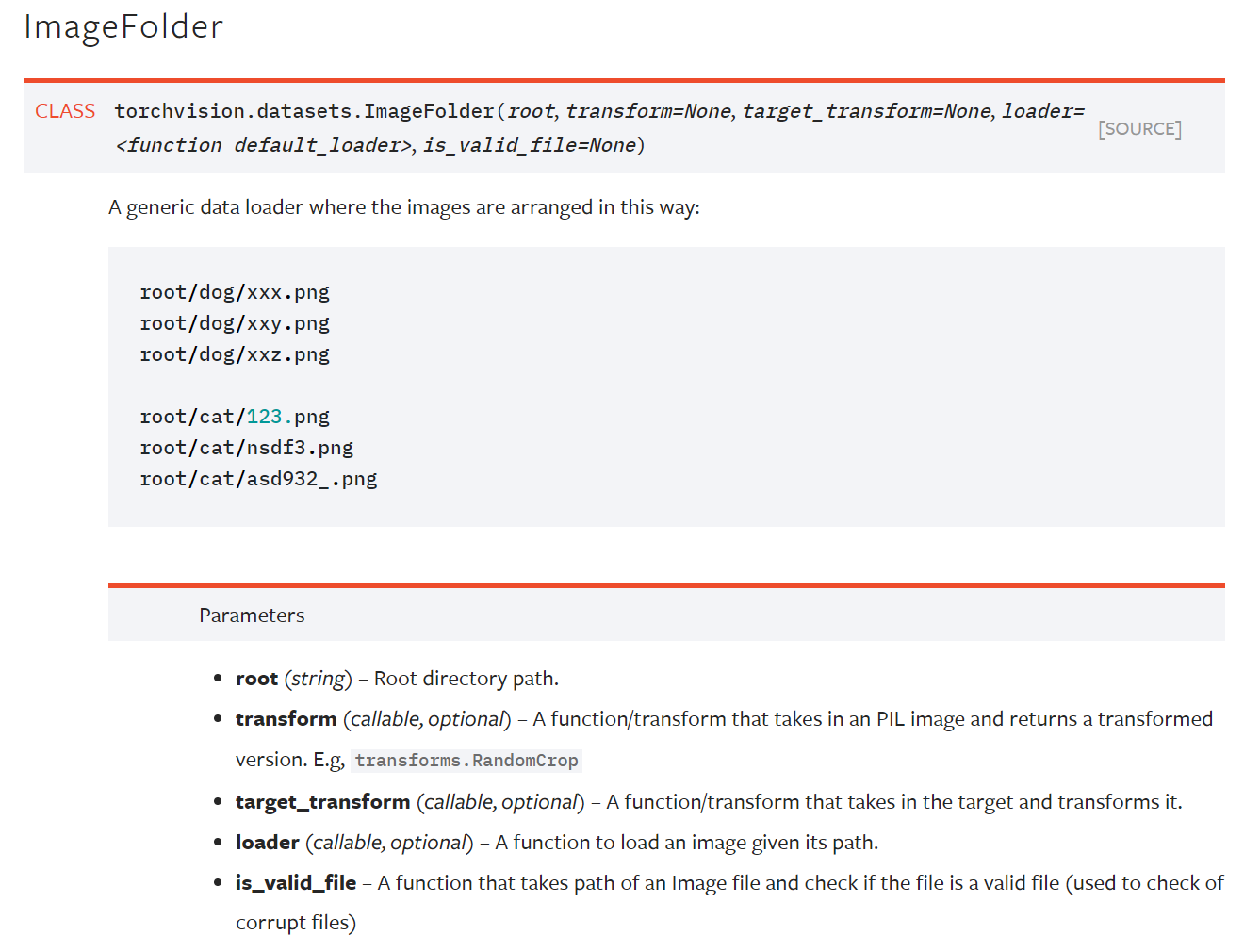
在PyTorch當中客製化數據集是必要的，因為有時候你的數據是「一張照片配一個標籤」有時候是「一張照片配多個標籤」甚至還有個多元的標籤形式，所以這邊會提到怎麼去製作自己的數據集。

## Dataset、DataLoader之間的關係

PyTorch將所有數據打包在torch.utils.data.Dataset 當中，這邊可以選擇你要怎麼提取你的數據集 ( 單筆 )，像是前面所說的一張照片配一個標籤或者搭配更複雜標籤，也可以在建置數據集的時候使用數據增強，透過變形、裁切來增加數據量、神經網路模型的強健度；宣告完之後再將 Dataset 打包進 torch.utils.data.DataLoader 當中去推送，這邊你可以選擇一次要丟幾張照片出來進行平行運算。目前我比較常用的有兩種定義數據集的方式，如果是已經用資料夾分類好的就可以使用ImageFolder 來製作數據集，如果要取得更多訊息就會客製化一個數據集。

## 使用ImageFolder建置數據集

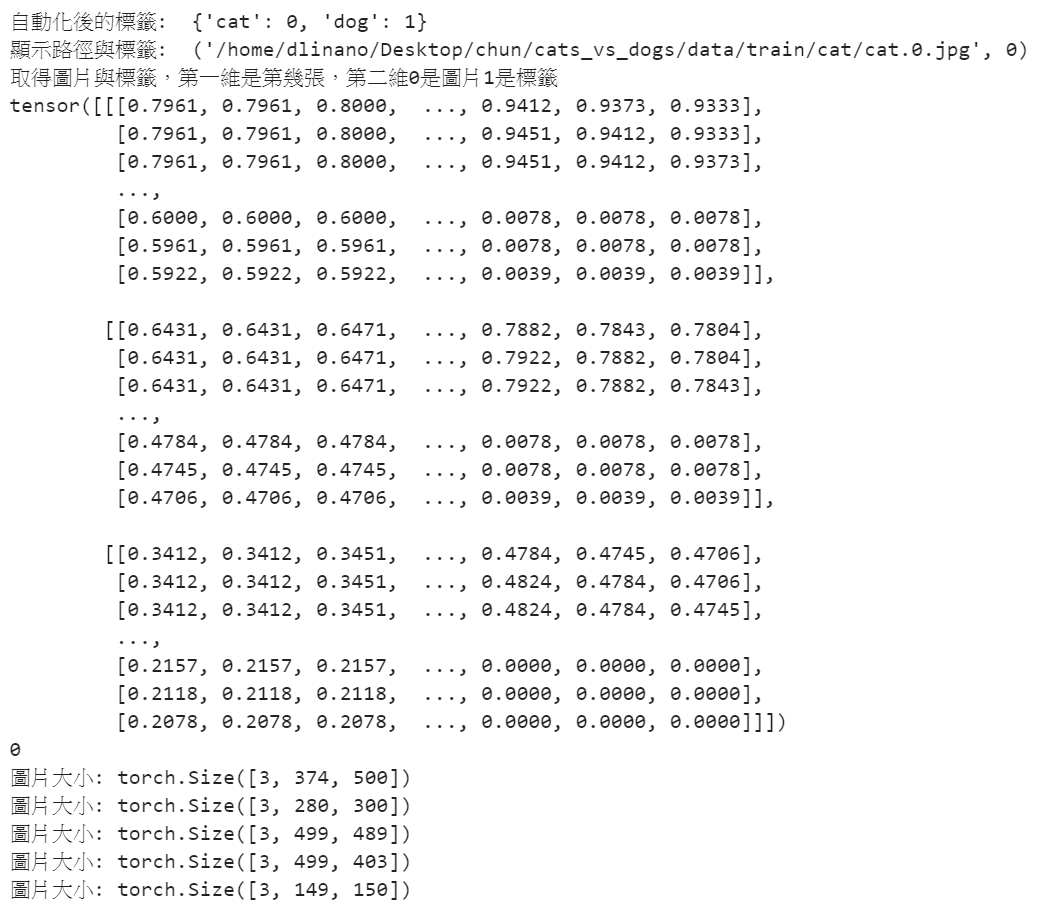
這是torch提供最一般的數據集整理方式，我們先前已經將貓跟狗都丟進各自的資料夾了，所以等等直接使用這種簡單粗暴的建制方式。



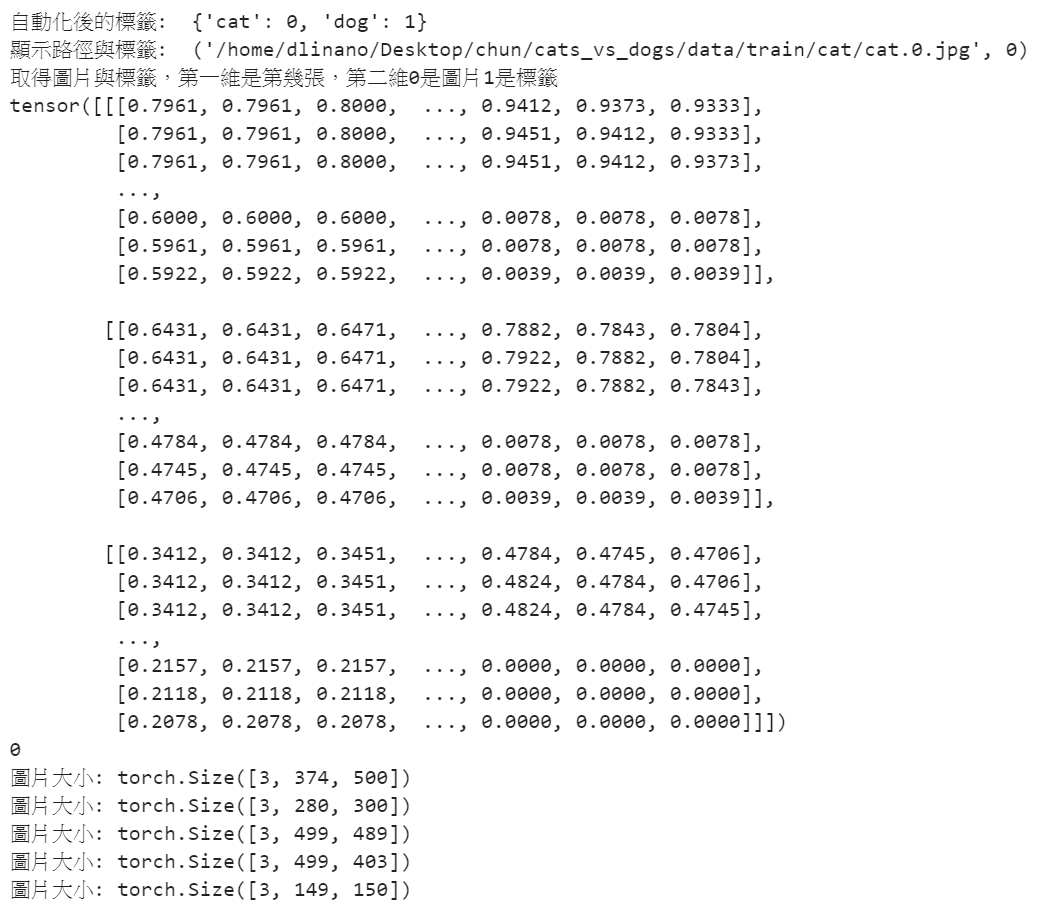
程式碼如下，在建置數據集的同時我們會先定義transform，這個目的通常用於數據處理、增強，可以做裁切、變形、轉檔等功能，也是很重要的環節：



我們將數據集拿出來看可以發現貓跟狗已經幫我們分類成0跟1了：



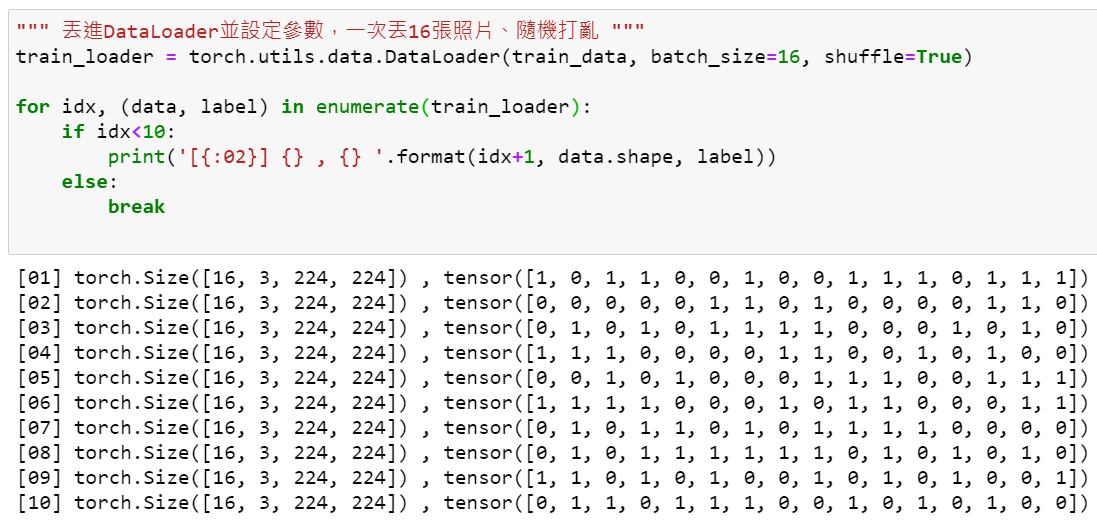
再來會注意到一個問題，我將前五張圖片拿出來查看大小，發現每一張圖片的大小都不一樣，這時候必須去處理維度的問題，不然捲積神經網路沒辦法跑

通常遇到這種問題最簡單粗暴的方式是直接在transform的地方加上resize，當然會有更好的解法但這邊暫時先不講述，我們將所有的圖片都先縮放到 224 \* 224並且轉換成Tensor後做一下正規化。



## 使用DataLoader批次丟出數據

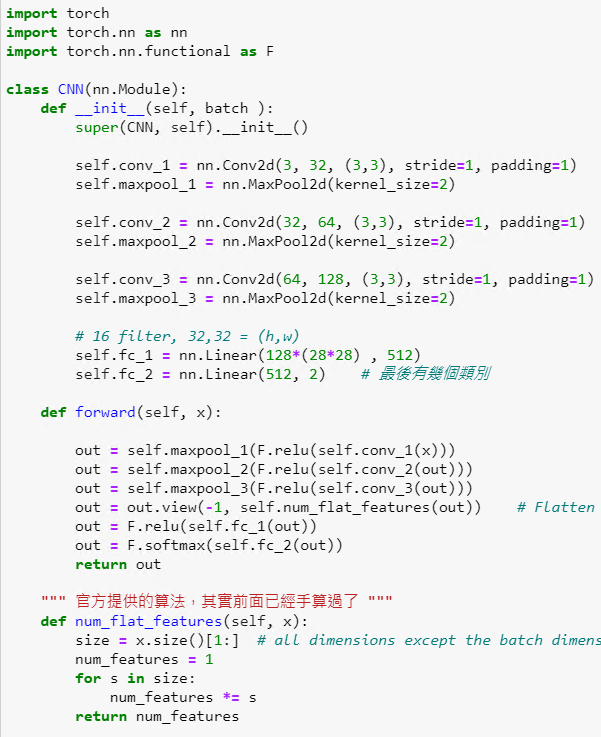
這邊我為了防止跑太久將迴圈限制在10次，可以注意到每一次都出去都是16張 ( 維度為 [16, 3, 224, 224] )，並且從label那邊可以看出來都是打亂的。



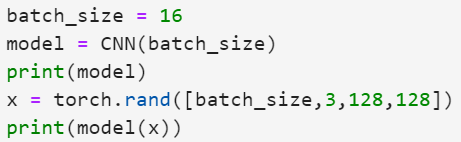
## 建立一個捲積神經網路

捲積神經網路的概念就不贅述，主要流程就是 捲積( Conv ) > 池化 ( MaxPool ) > 攤平 ( view ) > 全連接 ( linear ) > 輸出 (維度要注意)，這邊我在最後一層加了softmax讓兩者加起來為1比較好看，除此之外要注意的地方是必須要自己去計算每經過一層捲積池化，圖片會變成多大，因為最後攤平的時候必須要先宣告資料輸入的維度是多少，計算公式為：

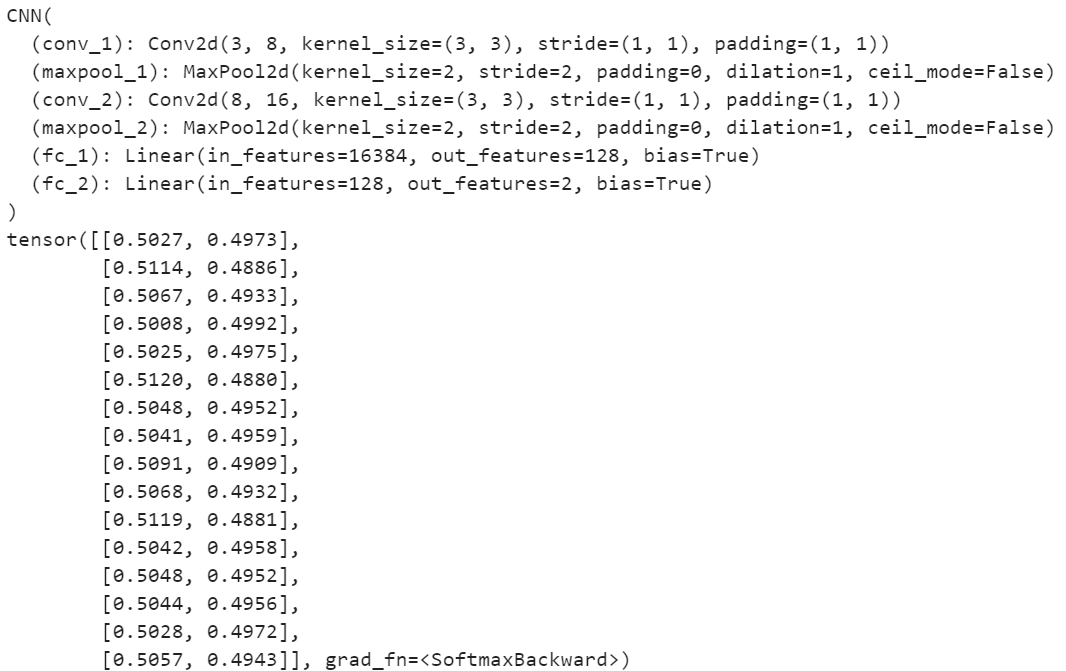
所以的會發現全連階層的第一層輸入是 128 \* 28 \* 28，其中128就是最後一層conv的kernel數量，28\*28則是自己算出來的。



可以將神經網路打印出來或者導入一筆Data看看



最後輸出是二維的因為有貓跟狗兩個類別，如果位置0的數值比較大代表神經網路判斷為貓，反之位置1的數值比較大則是判斷為狗：

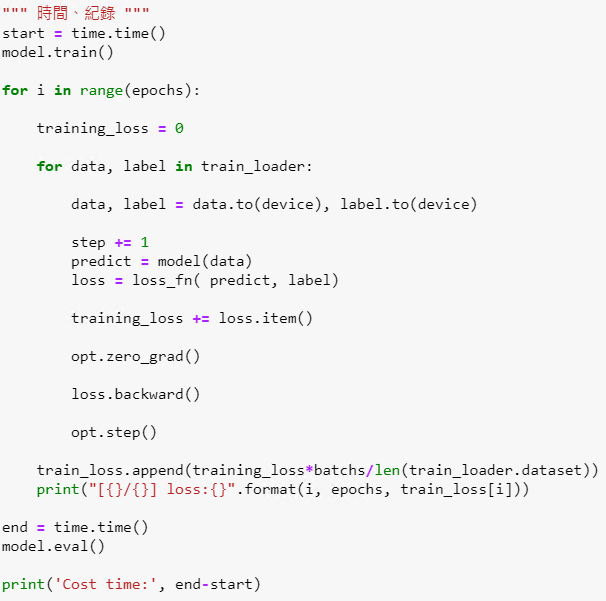


# 開始訓練

先將基本的設定好，這邊要注意的是我將模型丟到GPU中去訓練，因為是分類問題所以使用CrossEntropy當作損失函數，最後使用Adam來當優化器。



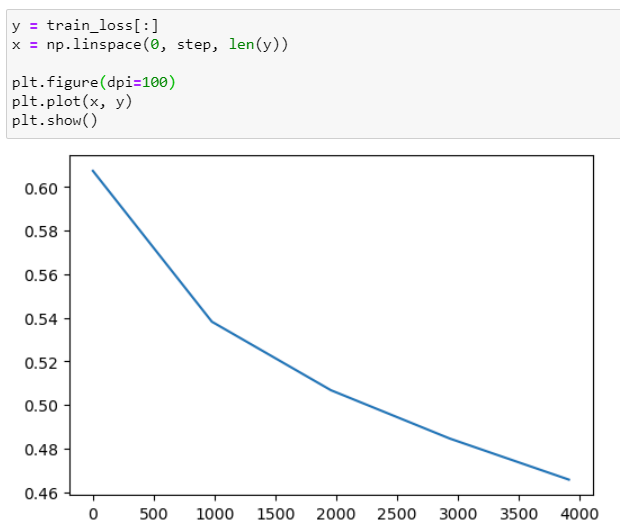
訓練之前記得先將張量也丟到GPU中，其餘的訓練流程皆與之前教學雷同，我特別將每次迭代的Loss存下用來視覺化用，這邊我看到網路上很多人會使用model.train()、model.eval()，這是torch自動會將 BatchNorm、Dropout在驗證的時候關掉，由於我們現在自己建的神經網路沒有這兩層所以其實是可以不用寫的，不過礙於之後會陸續增加技術，BatchNorm是一定會寫入的，所以現在大家可以先習慣寫這兩行，訓練開始前將模型設成訓練模式，結束的時候設成驗證模式。



訓練完的結果，在一般電腦使用RTX 1080訓練耗費了約545秒，而JetsonNano上使用gpu訓練每一次epoch約1300~1350秒，5個回合訓練完大概是6500秒左右也就是幾乎快兩個小時了，看似效能差距很大，但是其實考量到它的價位、體積以及運算能力，其實是相當的不錯了！

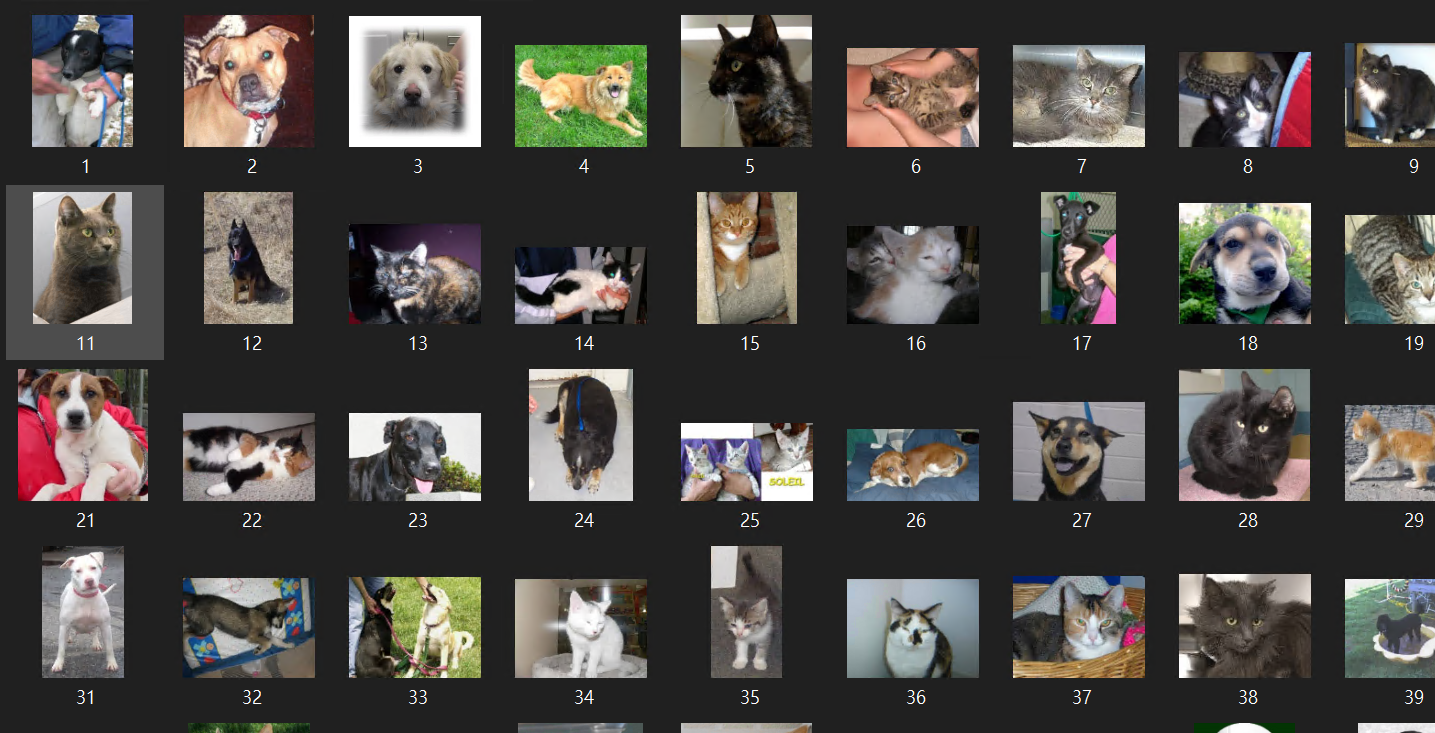
|  |  |
| --- | --- |
|  |  |

我有先將loss儲存起來，所以這邊可以直接調用並視覺化：

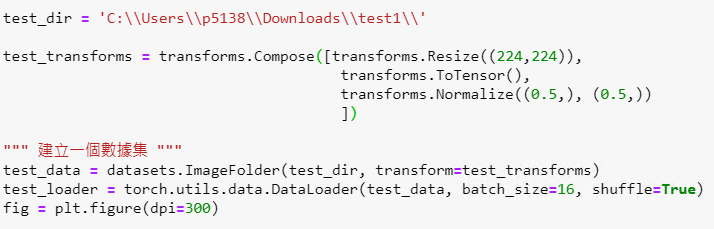


# 測試數據

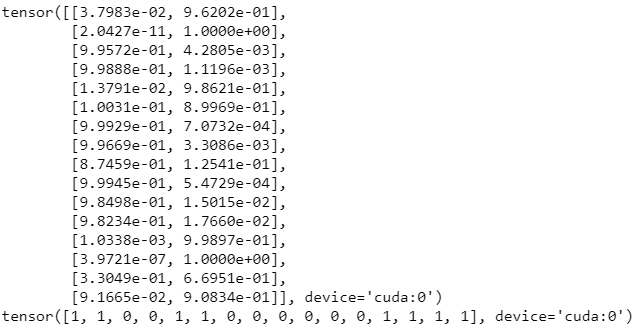
驗證集的資料型態比較不同，我為了使用ImageFolder在test1資料夾中又新增了一個test資料夾並且將圖片都丟入其中，最後用ImageFolder打包。



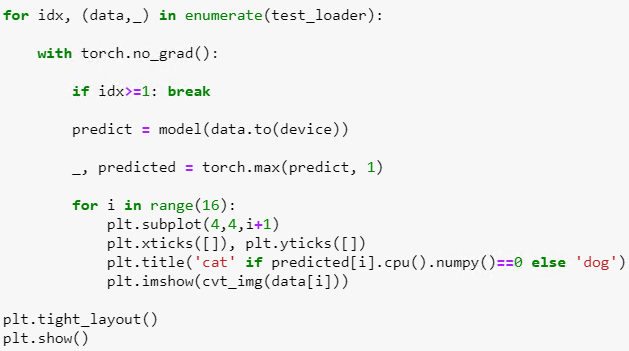
當初有做什麼樣的圖片變化，測試的時候一樣要做，否則預測就不準確了：



接著就可以進行預測了，這邊我們只取第一個batch的圖片也就是16張圖片，將其丟進神經網路模型後會獲得一組數據 [ 16, 1 ]，代表16張圖片的預測結果，我們可以透過程式碼取得較大數值的維度，用來判斷貓( 0 ) 還是狗 ( 1 )。



預測的程式碼如下：



最後結果可以看到，這次預測１６張圖片只有錯２張圖片～

