**來用CNN辨識圖片**

卷積神經網路CNN是含有一個或多個卷積層的神經網路，目前常用於機器視覺應用領域。同樣我們使用Google Colab環境支援此次練習。

**機器視覺趨向使用CNN的原因**

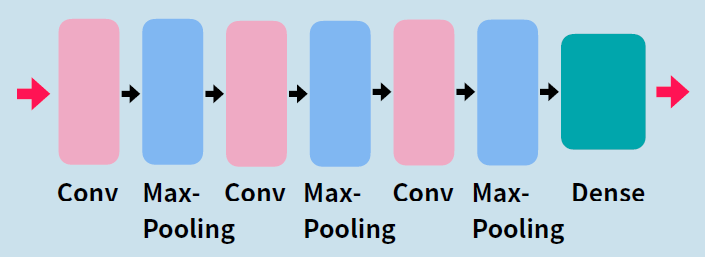
在CNN之前，機器視覺使用的是ANN密集神經網路來辨識資料，但ANN有兩點造成便是精度上有瓶頸。

第一點是輸入資料需要先將2D軸形式影像結構展成一軸，以利輸入1D的密集神經網路計算，但此舉會破壞視覺特徵之間的關聯。

第二點是影像通常含有不少的像素，若以密集神經網路來處理，所需要的計算能力會相當巨觀，且網路結構太複雜，而訓練資料不夠多時，會發生過度適配(overfitting)的問題。

結論是我們需要神經網路層數可以降低，且學習能力最好要更佳，於是CNN卷積神經網路大神也就應運而生了。

CNN的理論介紹可參考:深度學習CNN懶人包。



本次實作採用深度學習CNN加上MNIST手寫數字圖片資料集訓練神經網路。

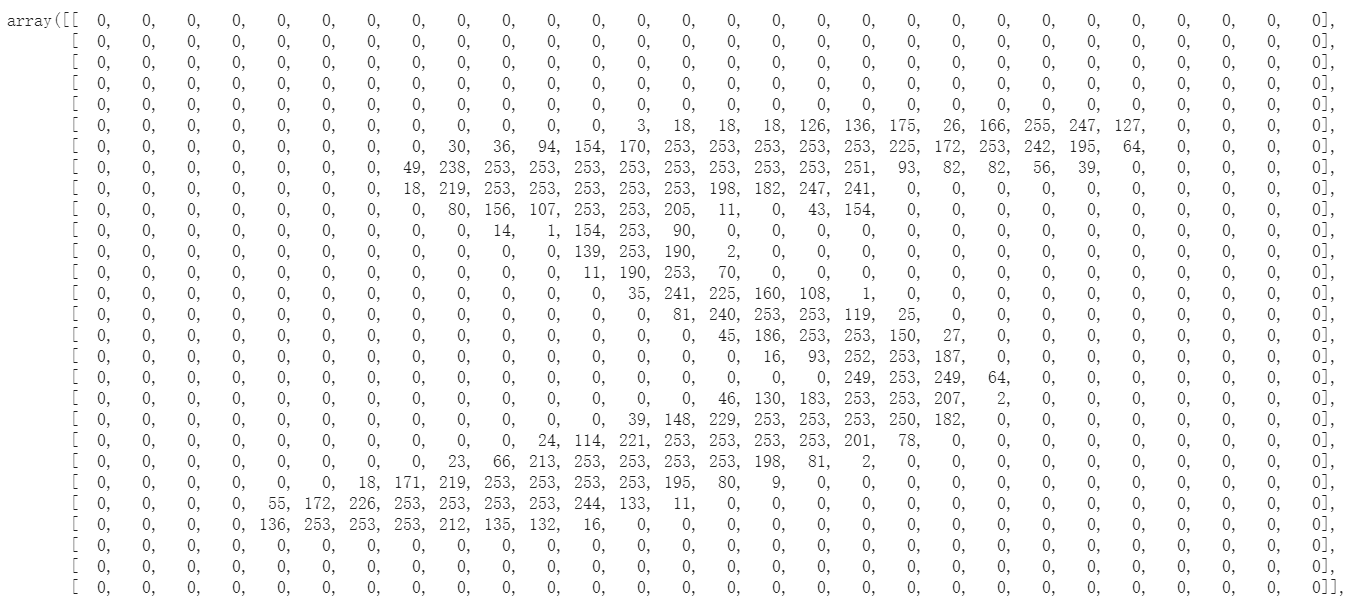
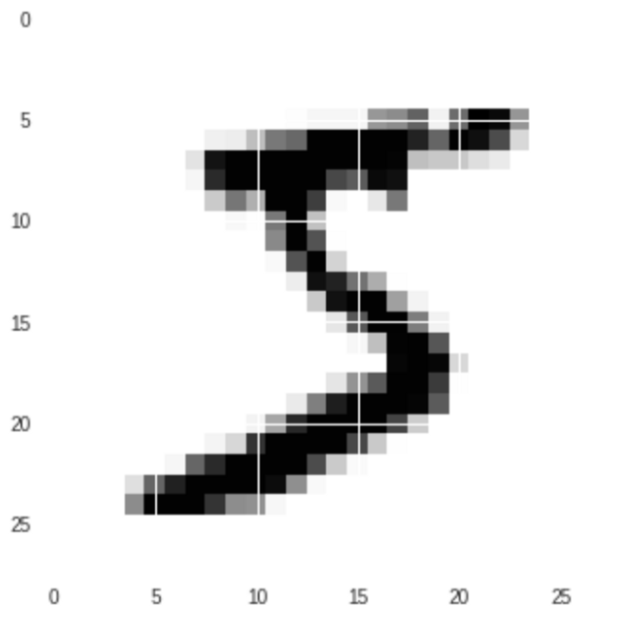
**CNN辨識手寫數字圖片**

1.卷積層的濾鏡數在第一與二層，我們設定為32與64個。

2.只用一層池化層。

3.使用ReLU激活函數。

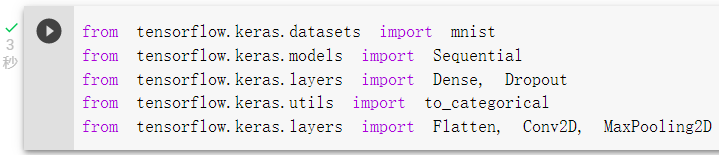
MNIST手寫數字圖片資料集中每張圖片都是28\*28像素的黑白數字圖案，每一個像素值為0(黑)到255(白)之間的整數，筆跡的像素會依深淺給予不同知數值。

****

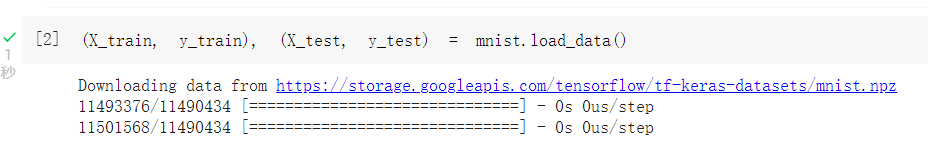
實作流程如下:

使用Google Colab環境執行此次練習，可參考文末說明。

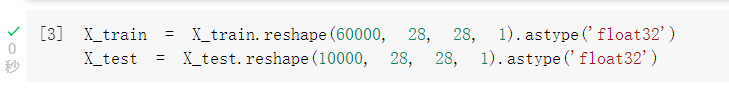
#### 步驟一， 匯入模組

Conv2d實作卷積層，MaxPooling2D實作池化層，Flatten層負責將卷積層或池化層處理完的多軸特徵圖轉成單一軸資料，並匯入密集層神經網路。

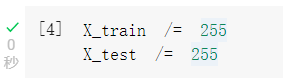
步驟二，匯入MNIST資料集

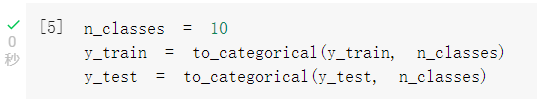


資料傳遞給Conv2d()時必須先轉換成4軸陣列的形式，分別是(樣本數，寬，高，顏色通道)其中顏色通道若是單色則為1，若是全彩則設為3。



將數值正規化讓值介於0到1之間





步驟三，規劃 CNN 模型的架構



1.第一及二層卷積層的濾鏡數為32與64

2. 濾鏡視窗為3\*3

3.activation=ReLU激活函數

4.MaxPooling2D()使用於減少特徵圖尺寸，pool\_size設為2\*2，輸出的特徵圖會是輸入的1/4。

5.使用Dropout()降低overfitting問題。

6.Flatten()把MaxPooling2D()輸出的特徵圖展平成1D陣列。最後送入隱藏層Dense。

7.將輸出層轉換成機率。

步驟四，訓練CNN神經網路模型

我們每次從訓練資料集X\_train，Y\_tain取128個樣本執行10個週期訓練(比ANN訓練時少了190次)，並以測試資料集X\_test，Y\_test做驗證。

訓練結果如下:

第一次，準確率為0.9800



第十次，準確率為0.9908



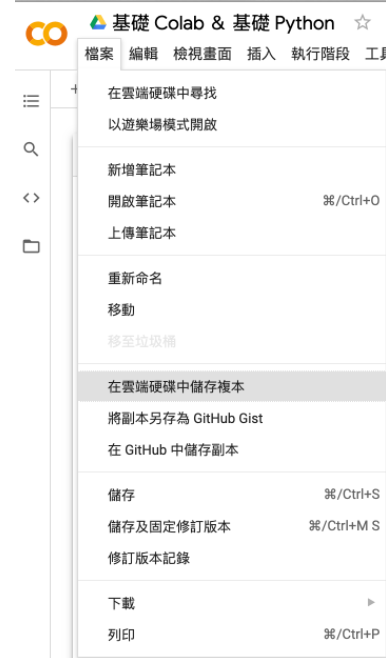
發現確實比單層隱藏層ANN實作時得到的準確率更高。

**附錄:前置作業Colab環境準備**

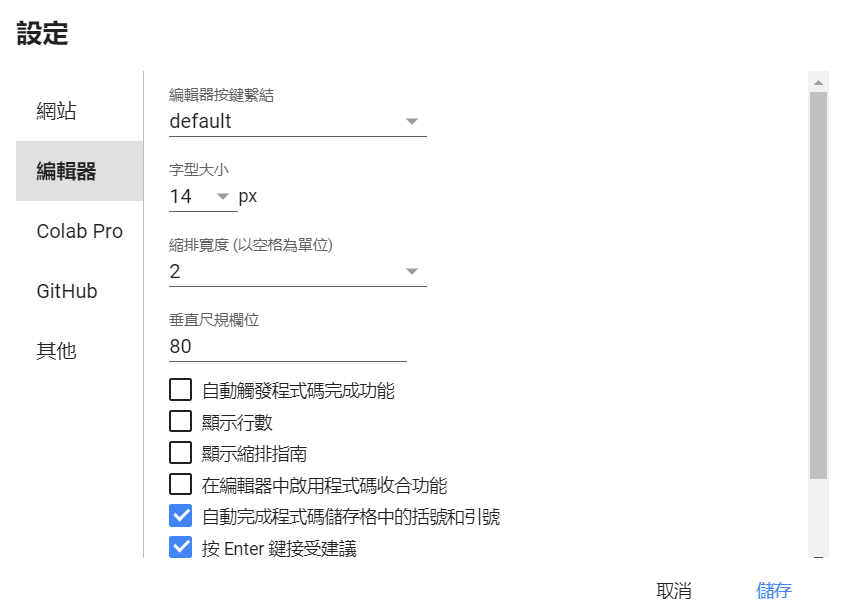
1.必須有自己的google帳號:使用自己的個人的 Google 帳號才可以執行自己 Colab 上的程式。

2.先下載 Chrome瀏覽器，並登入 Google 帳號。

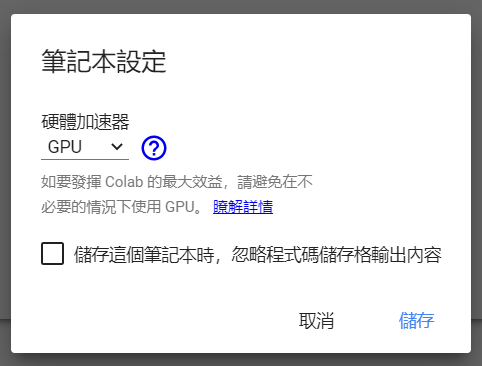
3.建立副本至個人目錄，否則不能存檔。



4.在⼯具 > 設定>編輯器中,反勾選「⾃動觸發程式碼完成功能」。



5.(必須做)設定執行階段類型。



6.療癒模式(非必須)，在⼯具 > 設定>其他, 勾選 「柯基⽝模式」和「貓咪 模式」，就有可愛的貓 貓狗狗來陪你寫程式了。



