1. 目標

1.透過訓練及來讓模型學習這些數字，並使用測試及看模型能不能成功辨識。

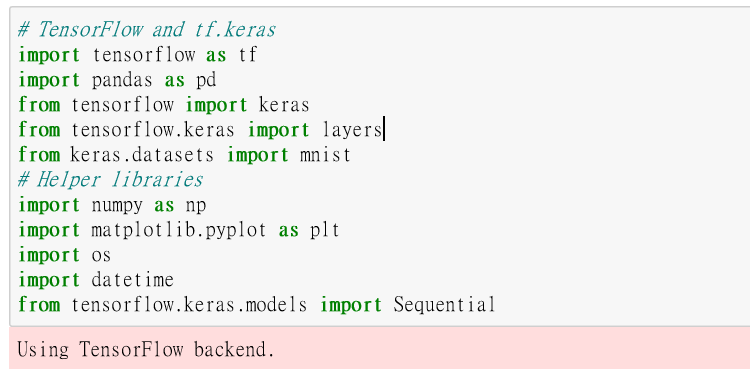
2.Epochs固定為20，其餘參數皆可調整(EX:層數、神經元數量)

3.嘗試解決overfitting，任選一種方法(所以最少有兩種model)

4.使用tensorboard觀察訓練變化

1. 方法

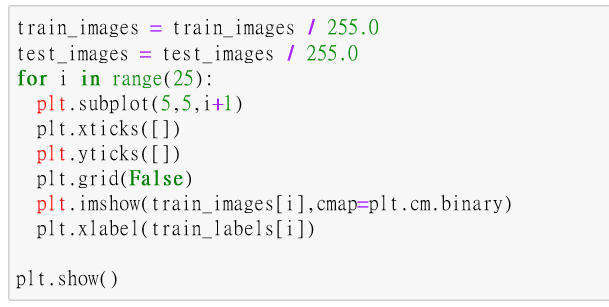
1.利用Dopout 的方式訓練:



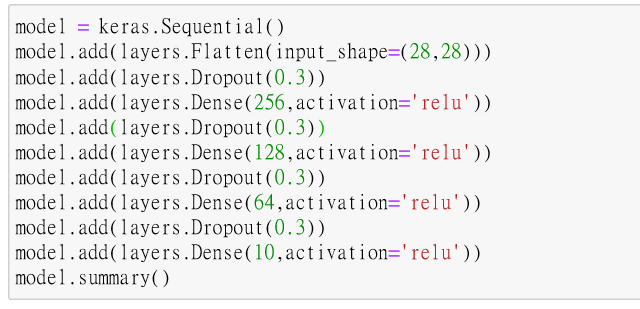
引進各個函式庫，並且將keras.datasets套件中的mnist模組也引用近來。

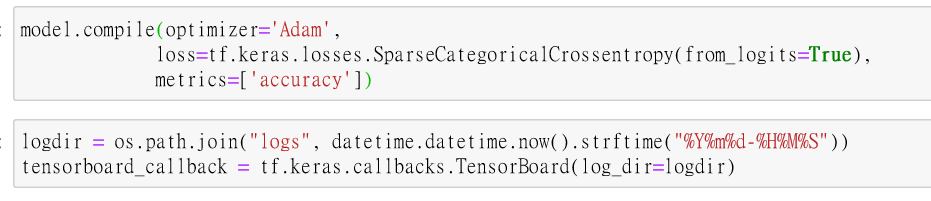


引進train\_images 和test\_images的大小以及相對應的標籤集的數量，並印出各個數值以及train\_images的第一個圖片。



將train\_images 和test\_images的數量 除以255，放入一個5\*5的資料集，並顯示出來。

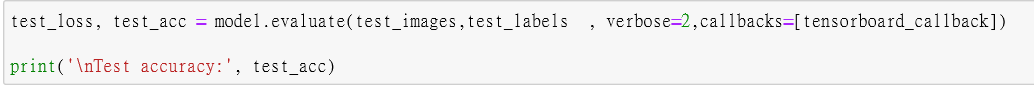


設計神經網路每一層的Dropout=0.3、並設計每一層的神經元數以及設定輸入為一個28\*28的圖片

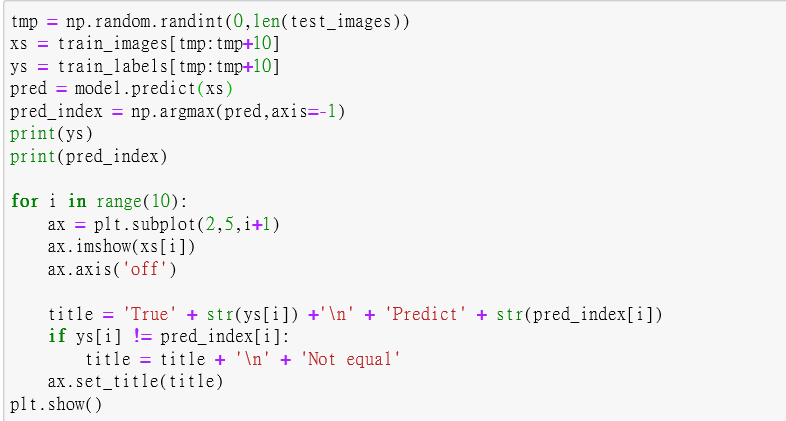
建立logs檔以便利用tensorboard 來觀察model的準確度以及誤差



訓練model 20次，並回傳跟測試集的差距(loss)以及準確度(accuracy)

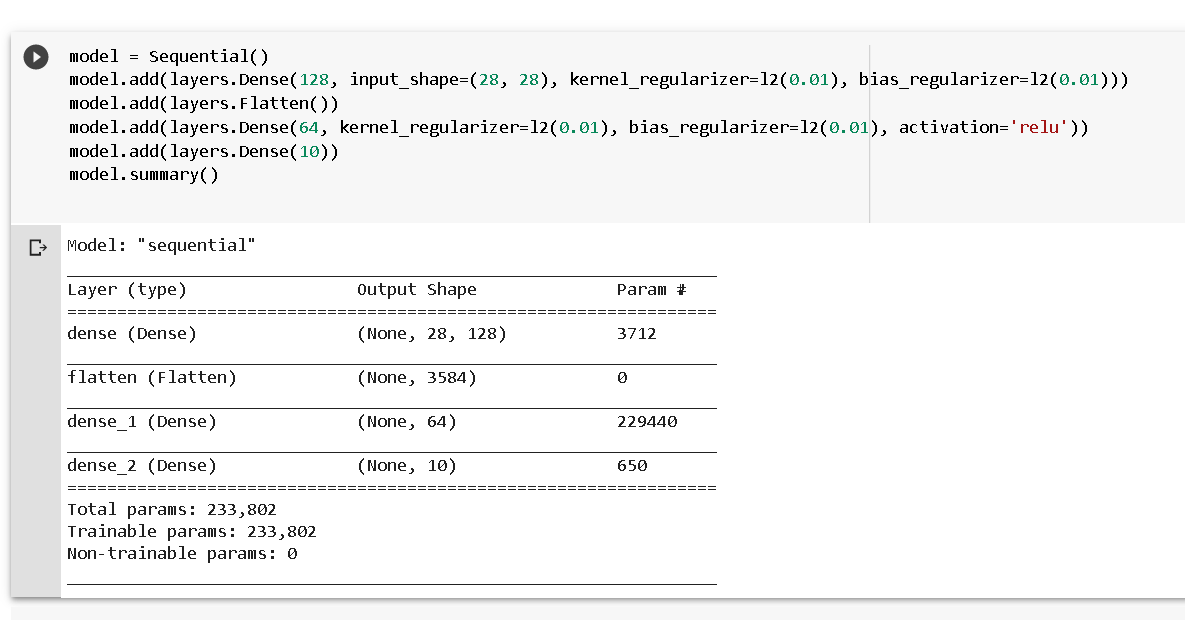


回傳最後訓練完的模型的誤差以及準確度



隨機測試5個數字並送入模型檢測，並觀察隨機亂數得值跟辨識的值是不是一樣。

2.正規化



此模型運用4層，從開頭到尾依序為

1. 密集層128神經元並加入regularizer權重正規化(權重與偏差)
2. flatten層
3. 密集層64神經元並加入regularizer權重正規化(權重與偏差)
4. 密集層10神經元

括號裡面0.01是正規化參數的值，

kernel\_*regularizer*是對於權重的正規化

Bias\_\_*regularizer*是對於偏差量的正規化

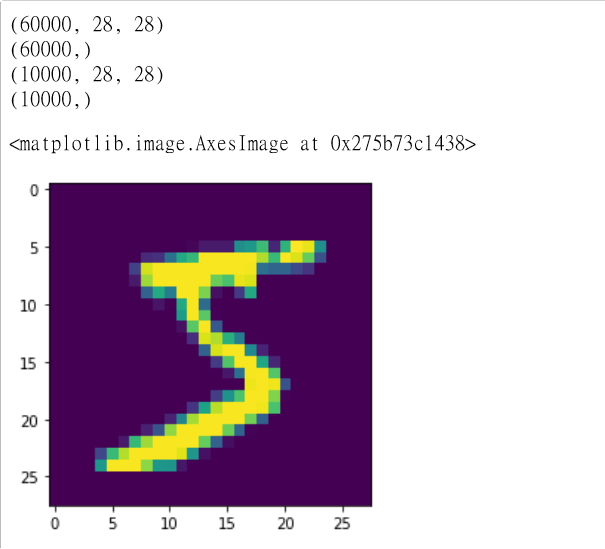
L2是使其正規化的時候，讓權重大的縮小，而權重原本比較小的維持小的權重

在此不使用L1是因為會導致被選中的權重縮為0，過於極端解決overfitting的效果會不如使

用L2。

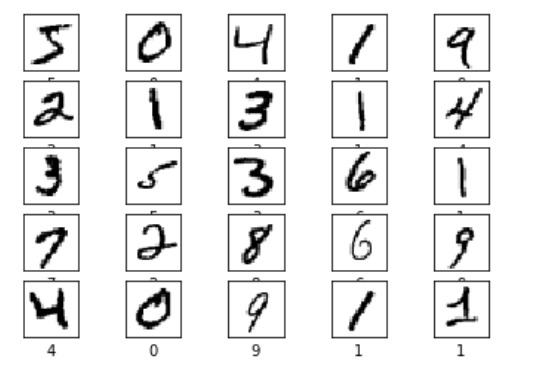
1. 結果

1.Dropout=0.3

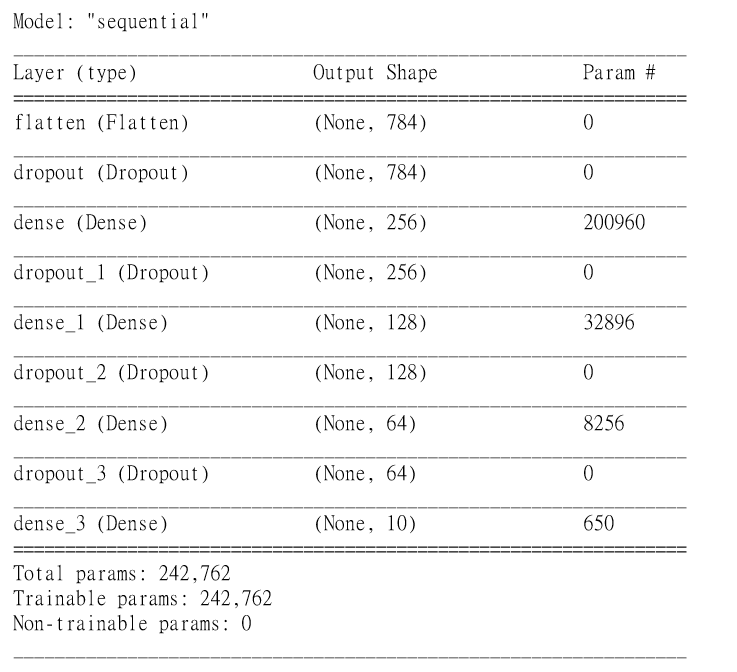


train\_images的60000張照片，大小為28\*28，相對應的標籤也是60000張

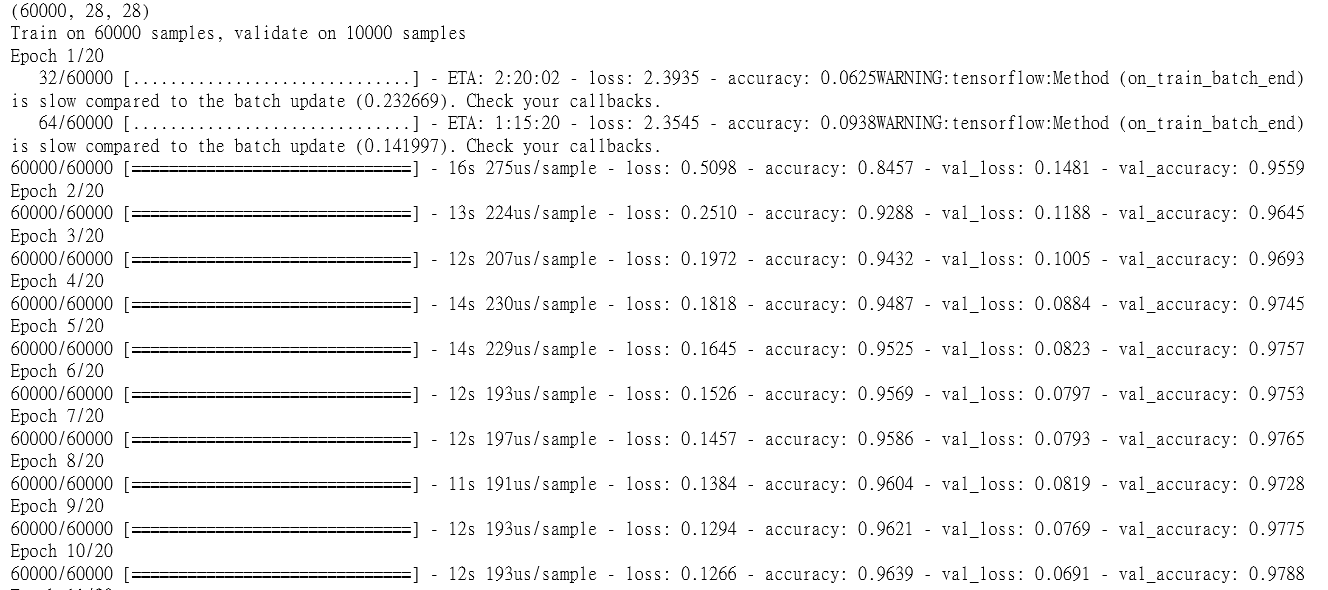
test\_images的10000張照片，大小為28\*28，相對應的標籤也是10000張



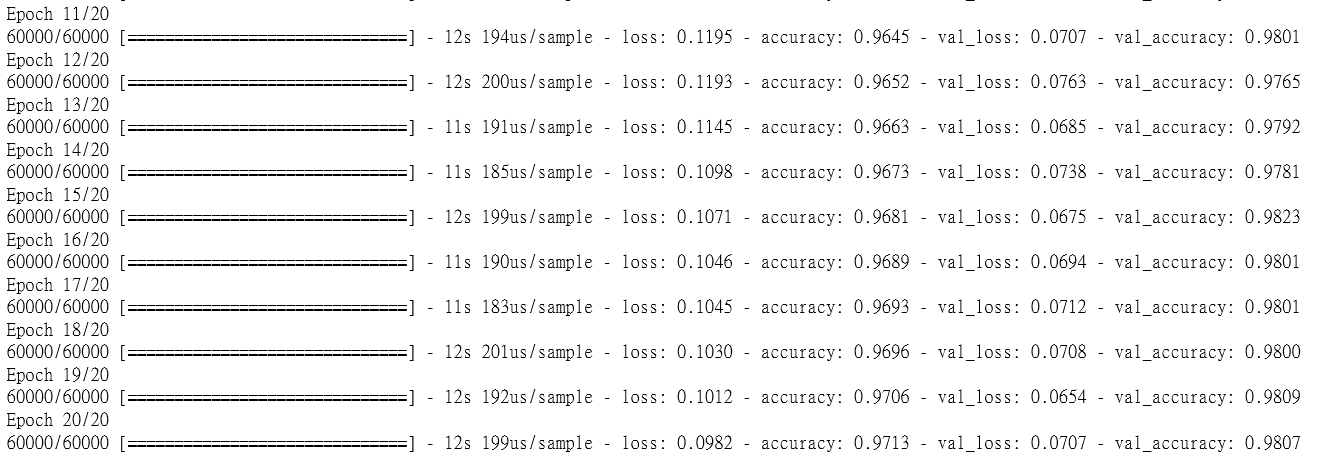
顯示25個篩選出來的資料



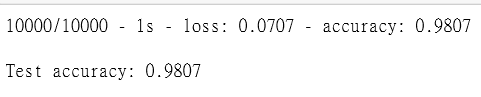
有四層神經網路，神經元數分別為256、128、64、10，最後為將各自要運算的次數加總，   
 並顯示出來。



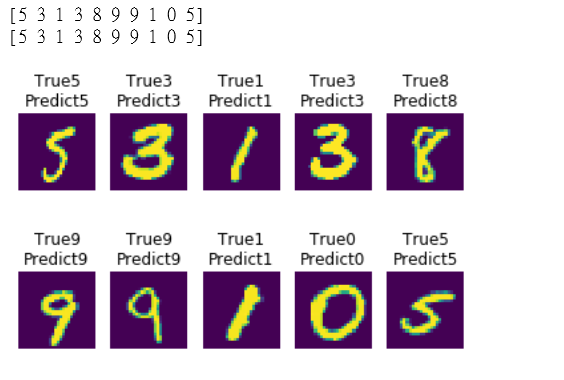
顯示1~10次的訓練情形



顯示11~20次的訓練結果

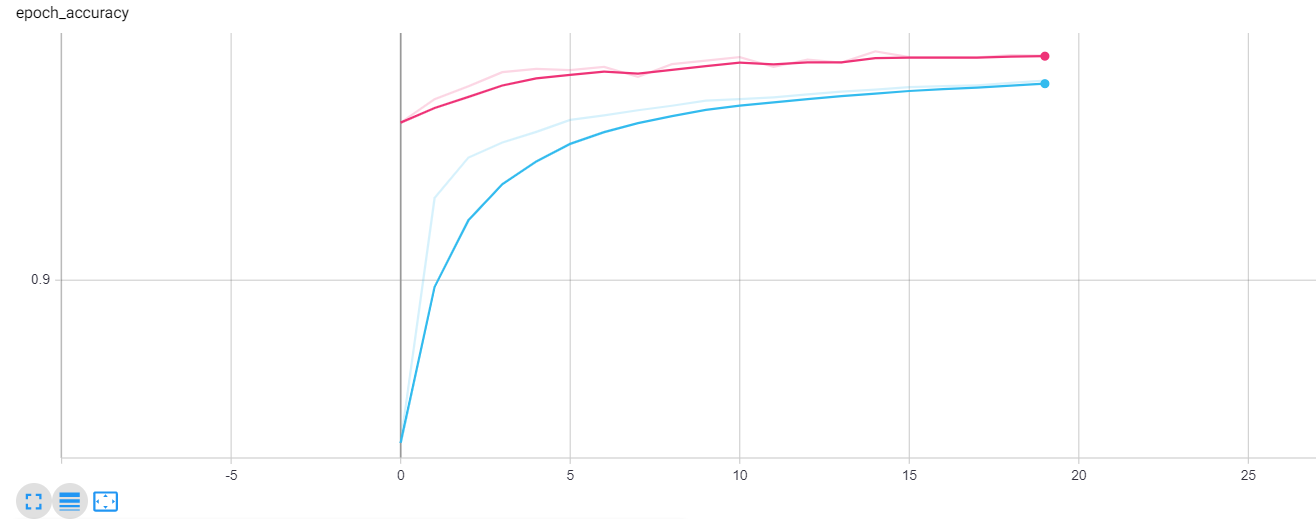


最後訓練完的模型為:損失:0.0707 準確率:0.9807

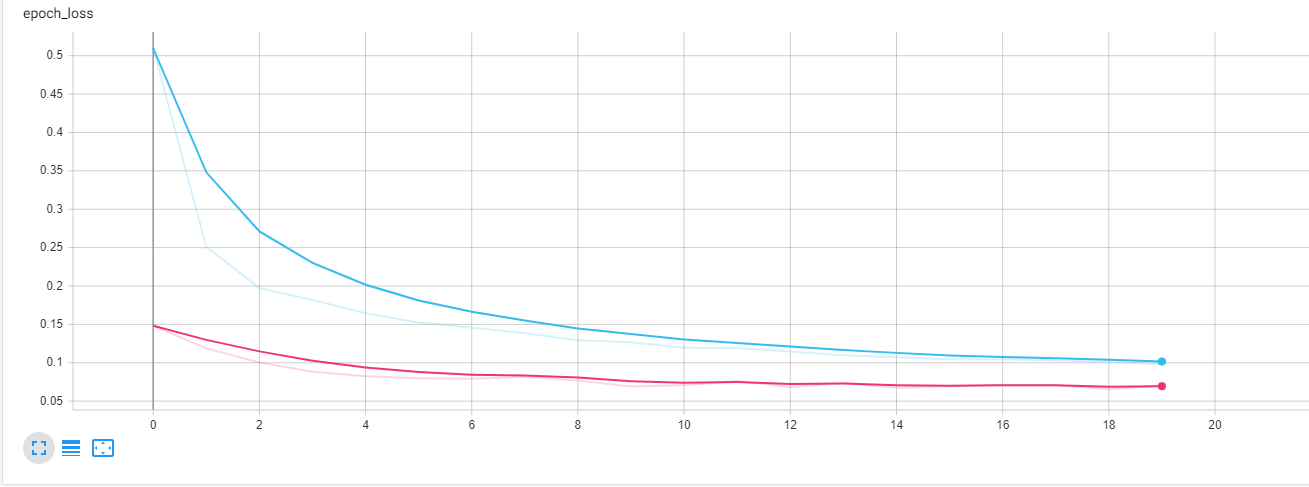


第一行為隨機亂數，丟進模型辨識後，第二行會顯示模型辨識的情形，在此成功辨識且輸出正確。

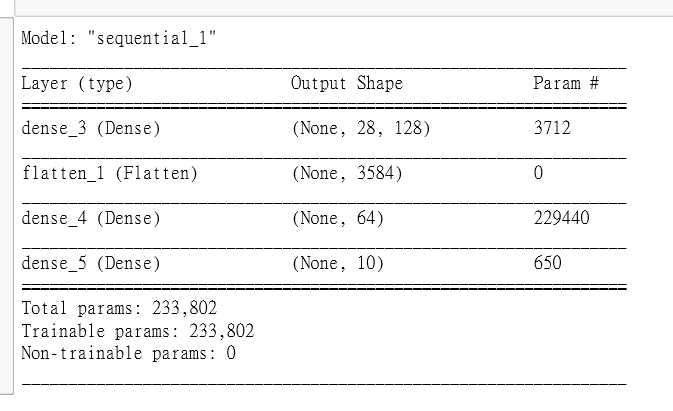
Tensorboard輔助觀察準確度跟訓練次數的關係



Tensorboard輔助觀察誤差跟訓練次數的關係

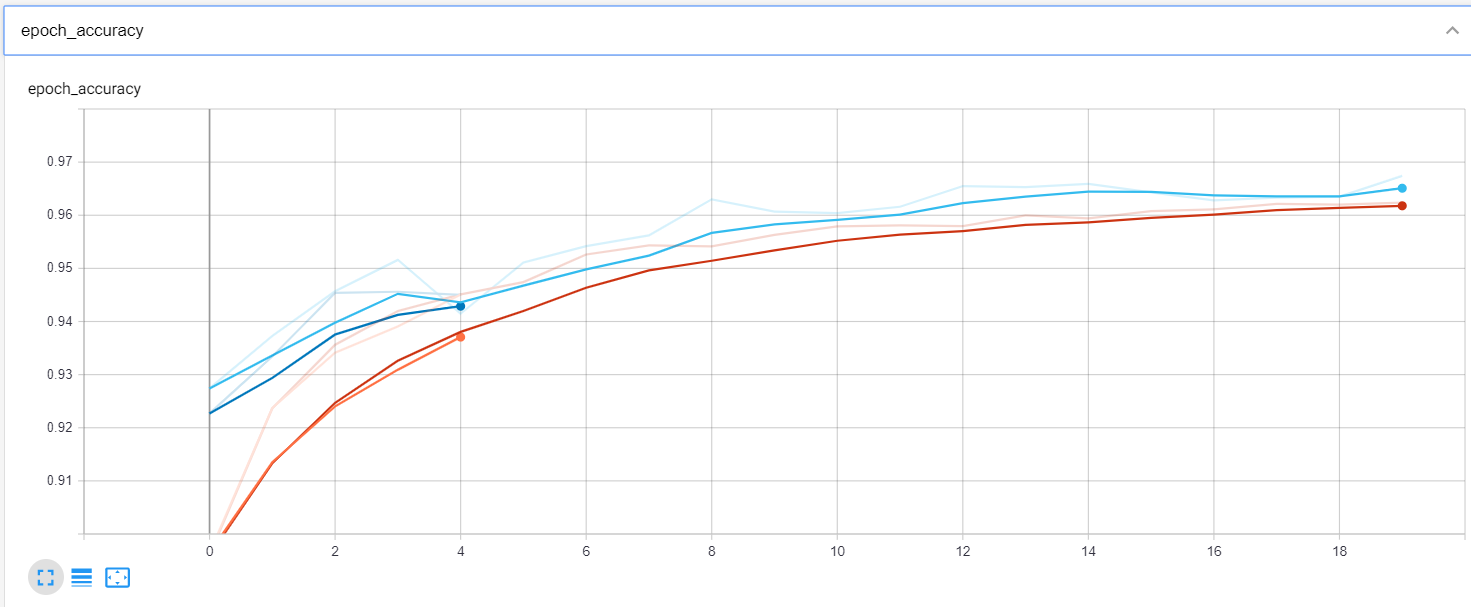


2.正規化



輸出結果顯示神經元數為233,802

利用tensorboard觀察訓練20次中的準確度。



利用tensorboard觀察訓練20次中的誤差。

