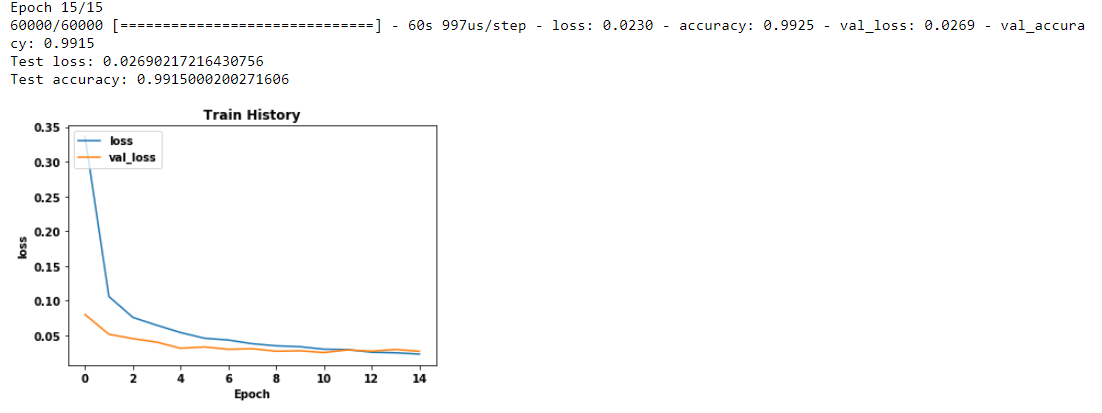
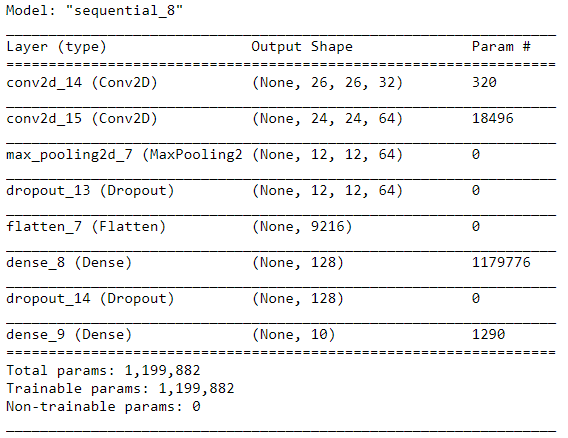
**機器學習系統設計實務與應用**

HW2 B063012054 林祐安



(未改變參數前)



(模型架構)

使用CPU Ryzen5 2600訓練每個epoch需耗時一分鐘左右，模型架構僅兩層卷積以及全連接層，理論上耗時不需太久，但在此次實作使用的樣本有60000份，數量不少，因此耗時長。為提高效率最簡單的方法是加大”batch\_size”，原先為256，由於資料有60000筆，可以試著把batch\_size提高至1024(每個epoch 約耗時47秒)、4096(每個epoch約耗時45秒)，觀察發現batch\_size=1024與batch\_size=4096的耗時差不多，這表示再增加下去意義不大，且訓練效果可能會非常差勁。較為進階點可以修改optimizer、甚至於模型架構。

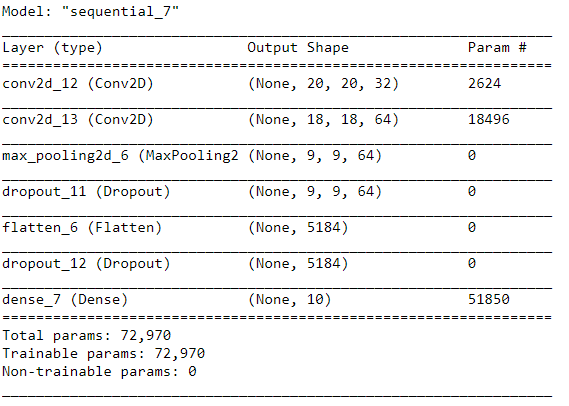
--------------------------------------------------------以下為修改後模型--------------------------------------------------------



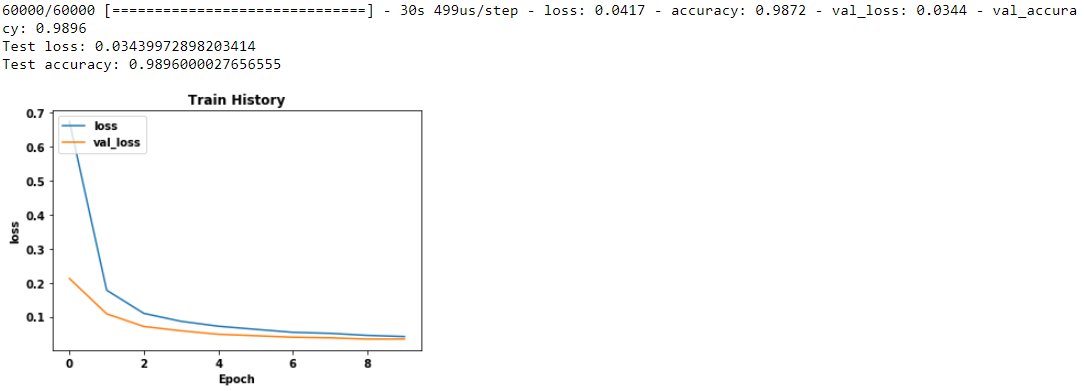
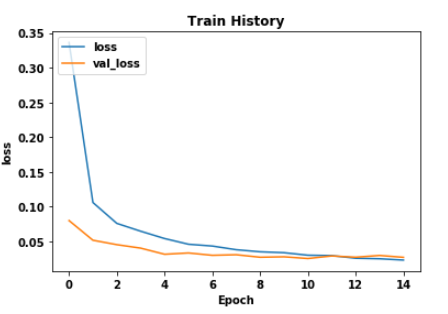
(修改後code)

紅框以及紅箭頭部分是修改的地方，為防止第三次作業沒東西可以寫，這次作業先討論紅框部份，紅箭頭的部份是刪除一層Dense。

1. 將第一層的卷積加大成9X9的Mask，因為圖片僅有黑白兩種顏色且只有1個channel，因此不需要小的Mask去抓取特徵，提高效率；第二層則維持不變，提高分辨率。
2. 修改第二次防止過擬合的Dropout至0.25，前面已Dropout一次，模型非常簡單且只有黑白兩色，理論上Dropout可以拿掉，但此資料量不少，因此斟酌改成0.25。
3. optimizer由Adadelta修改至Adam，Adam對於local min. 有極佳的效果，收斂速度雖然不是最快，但是最穩定，也是現在最常使用的optimizer。
4. 由前面的經驗可以知道，在epoch 12以後已開始收斂，且batch\_size=1024是瓶頸，預期更改後的模型效率較高，因參數量大幅減少，所以調整epoch至10。

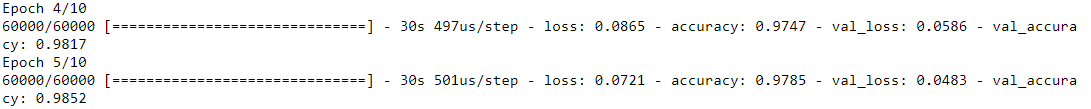


(修改後模型架構，刪除一層全連接層)

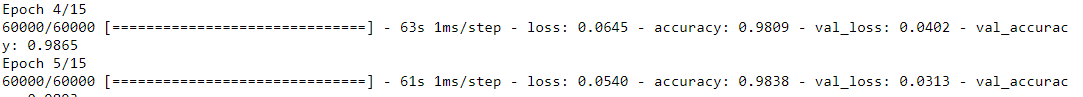


(修改後模型訓練10th epoch與右方修改前的模型Train History)

先觀察修改後的模型，可以看到每個epoch的時間由一分鐘減少到半分鐘，快了一倍的時間，效率大幅提高，事實上在4th epoch時loss已下降到0.0865，與原本的模型相比，雖然一開始的Loss較高，但由於動量關係，下降速度頗快，在epoch 4時兩者loss其實已差不多。



(修改後模型 4th、5th epoch)



(修改前模型 4th、5th epoch)

兩者的正確率、loss差不多，因此這是一個非常成功的改良版模型，大幅提高了訓練效率。

以上是由CPU訓練的成果，接下來是使用GTX 1066做訓練。



(未修改前模型)



(修改後的模型)

原先使用CPU需分別耗時1分鐘與0.5分鐘，使用GPU僅需5秒與3秒，使用GPU果然最有效率。明顯看到就算是使用GPU，修改後的模型訓練時間也是未修改前的一半。

比較大的問題是前述紅字部份，既然拿掉了第一層Dense，第二個Dropout的用處在哪？以及如果第一層Dense存在，那拿掉兩個Dropout是不是沒有太大影響？由於後者牽扯到模型架構的變動，這部份會留到第三次作業實作，先討論前者。因為原本還有一層Dense在中間，把1/4神經元關掉後壓成128個輸出再Dropout一次，這兩次Dropout因為中間加了一層Dense所以作用不同，但修改後的模型拿掉第一層Dense後其架構為Dropout(0.25)->Flatten()->Dropout(0.25)，其實和Flatten()->Dropout(7/16)道理相同，寫二層Dropout實屬多餘，這部份是在後續檢查時發現的問題，當初修改model時沒有注意到，因此特別拿出來討論；若使用相同的數據庫，而僅僅是增加卷積層(保留原本的第一層Dense)，理論上Dropout影響會比較明顯，因為抓取的特徵更細微，丟掉這些特徵會影響前幾個epoch判讀，但此次圖片單純應不會有甚麼差別，對於RGB圖片較敏感；Dense的去留則於下次作業的問題與討論提出。

本次實作CNN的Hello World，雖然是很大眾的資料庫，但也是很重要的學習經驗，因為資料單純，更容易著手思考該怎麼處理，網路如何建立，參數的挑選都是需要一步步的累積。此手寫資料庫中其實有許多連人類都難以辨識的圖片，假設人類對此圖片認知為9，但機器認為是4，在人類不確定的狀況下如何判斷機器是錯的？當圖片辨識的境界到了一個高度，就會產生如此的問題，因此在討論正確率99%附近時，無法斷定這個模型是否已完美，又或是需要再改良，整個資料庫有六萬張圖片，裡面不乏可以辨識但標錯的，當然，我們也不可能再去一張一張檢查。

Code link：

[Training by CPU](https://github.com/autotntfan/Design-practice-and-applications-of-machine-learning/blob/master/L2/DPPML_HW2_B063012054_cpu.ipynb)

[Training by GPU](https://github.com/autotntfan/Design-practice-and-applications-of-machine-learning/blob/master/L2/DPPML_HW2_B063012054_gpu.ipynb)