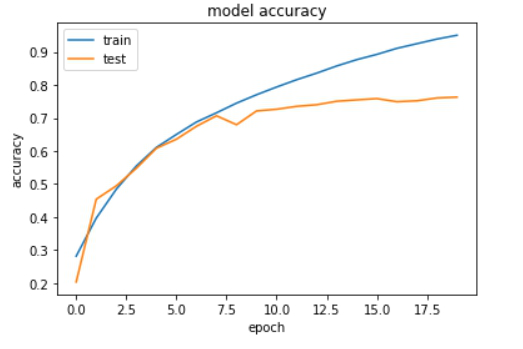
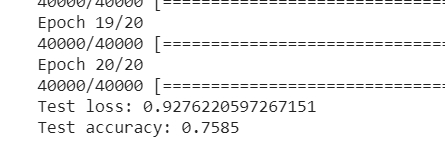
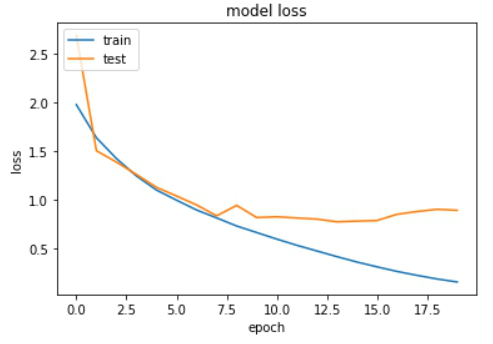
E14056114 徐顯舜 機械109

1.結果呈現

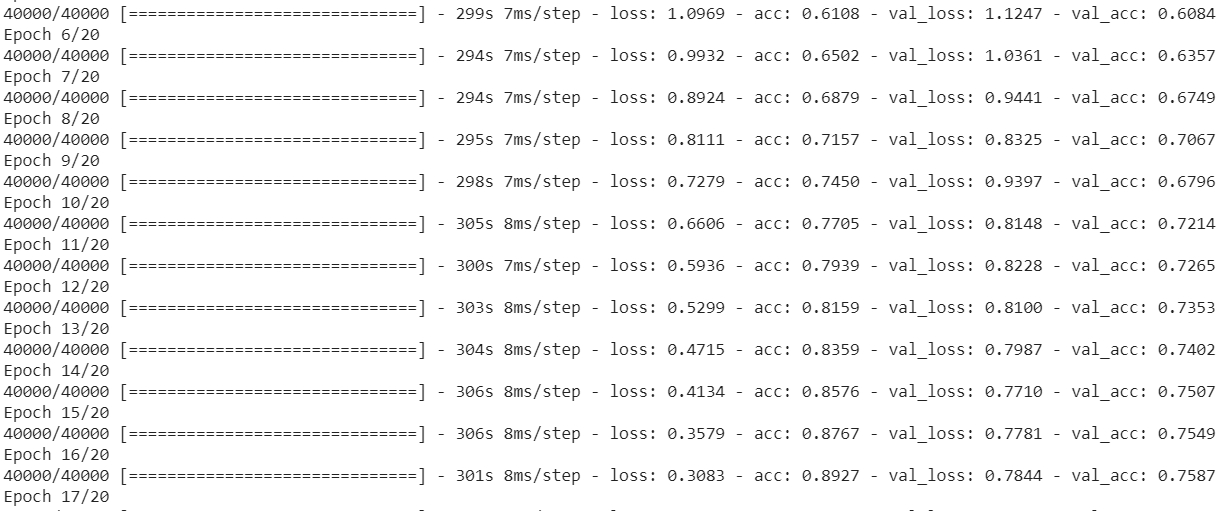


雖然一開始的正確率不高，但經過幾次的訓練過程後越來越趨近於**76%**





可以發現loss越低，準確率越高，而最後的loss還是些微的上升，但跟原始cnn程式相比好了一些。



(訓練過程擷取)

2.討論

環境: Google colab

前置設定:在一開始執行以下程式

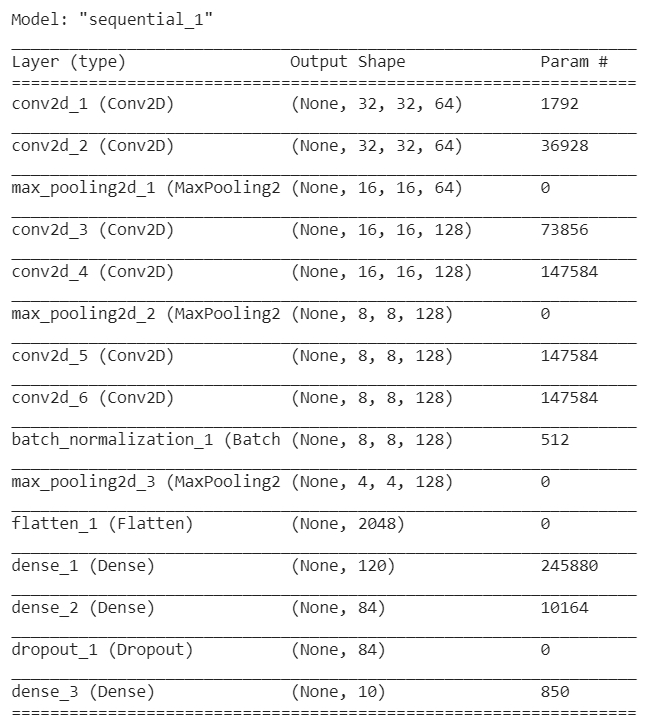
!pip install tensorflow==1.14.0

import tensorflow as tf

print(tf.\_\_version\_\_)

因為要使用的一些function貌似Tensorflow 2.20不支援，所以要降版至1.14.0

整體模型流程(但看完其他網路文章發現有些層放錯位置):



神經網路層，其中有兩層隱藏層

輸出層

第三組卷積+池化層以及BN加快收斂速度

第二組卷積+池化層

第一組卷積+池化層

這次的話其實設計的過程沒有太大的想法(主要做很多次的convolution和看看其他的優化器和function)，主要是去熟悉這些function的參數和意義。

model.add(Conv2D(filters=64, kernel\_size=3, input\_shape=(32, 32, 3), activation='relu', padding='same'))

此行意思為有64個filters（即我們稱的filter或kernel），kernel大小為3×3，卷積後輸入以及輸出的圖形大小不變（padding = ‘same’，即圖像周圍會補值），輸入的圖形維度為32x32x3，使用relu啟動函數。由於定義了64個filters，因此本層會輸出64個16×16的影像。

model.add(MaxPooling2D((2, 2), strides=(2, 2)))

接下來是池化層，本層參數比較少，我們只要定義pool size為(2,2)，即尺寸縮小為一半。本層接收了上一層卷積層的64個32×32影像後，會輸出16個14×14的影像。

 model.add(BatchNormalization(axis=-1))

使得深層神經網絡訓練更加穩定，加快了收斂的速度，甚至同時起到了正則化的作用。可以 training 加速，可以防止 gradient vanishing 的問題，可以幫助 sigmoid 或是 tanh 這種的 activation function可以讓參數的 initialization的影響較小

model.add(Flatten())

接著我們需要一個平坦層，將特徵值轉為一維資料以供後續的全連結層使用。本層接收第三個池化層傳來的4x4x128（=2048）資料之後，轉為1維有2048的值的矩陣。

model.add(Dense(120, activation='relu'))

建立全連結層中的隱藏層(即傳統神經網路中的全連結層)，指定其神經元數目為120個（此數目可調整），啟用函數使用Relu。

model.add(Dropout(rate=0.25))

在各兩層的卷積與池化後，我們加入Dropout層，它最大的功用是減少過度擬合。在深度學習的訓練過程中，Dropout會讓每次batch run都依據機率丟棄一定比例的神經元不予計算，使得每一次都好像在訓練不同的神經網路一樣。上方的程式中，我們定義該Dropout層每次訓練時要丟棄25%的神經元。

model.add(Dense(10, activation='softmax'))

本模型的最後一層是輸出層，也就是要輸出十種0~9的分類值，一般我們都會使用softmax作為分類模型輸出層的啟動函數。

 model.compile(loss='categorical\_crossentropy',

                  optimizer=ada(lr=0.01),

                  metrics=['accuracy'])

模型建立好之後，接下來要準備訓練此模型，在訓練之前，要使用compile來定義訓練的參數。損失函數使用深度學習分類模型中最常用的交叉熵cross entropy，梯度下降法使用Adagrad(learning rate = 0.01，但看到網路上很多人建議用adam，但還沒特別研究這些optimizer差別)，模型的評估方式則是以accuracy為優先。

瑕疵與改進地方:

查看其他網路文章發現Dropout和Batch normalization層放錯地方， Dropout層一般加在全連結層後以防止過度擬合，提升模型泛化能力。而很少見到卷積層後接Dropout（原因主要是卷積參數少，不易過擬合），而且Batch normalization可替換Dropout，而且表現得更好。但兩者放置位置不同:只在有全連接的時候用Dropout，在卷積之間使用 Batch normalization。

所以程式碼應該為:

model.add(Conv2D(filters=128, kernel\_size=3, activation='relu', padding='same'))

model.add(BatchNormalization(axis=-1))

model.add(Activation('relu'))

或

model.add(Dense(120, activation='relu'))

model.add(Dropout(rate=0.25))