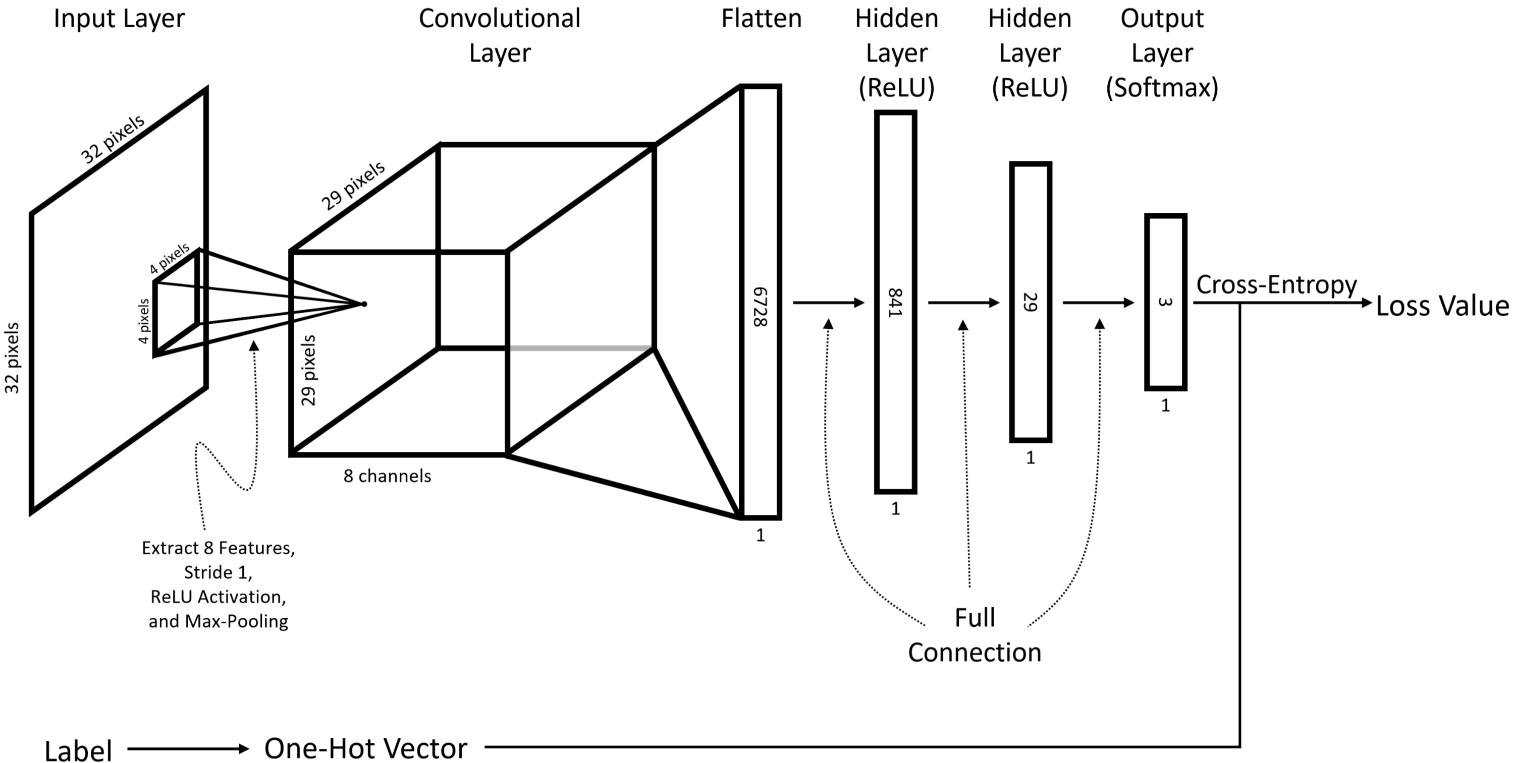
Homework 3

10910COM 526000 Deep Learning　王傳鈞　109062631

第一部分：架構簡介

關於我的convolutional neural network（CNN）架構，可以參考下圖：



圖一：CNN架構圖

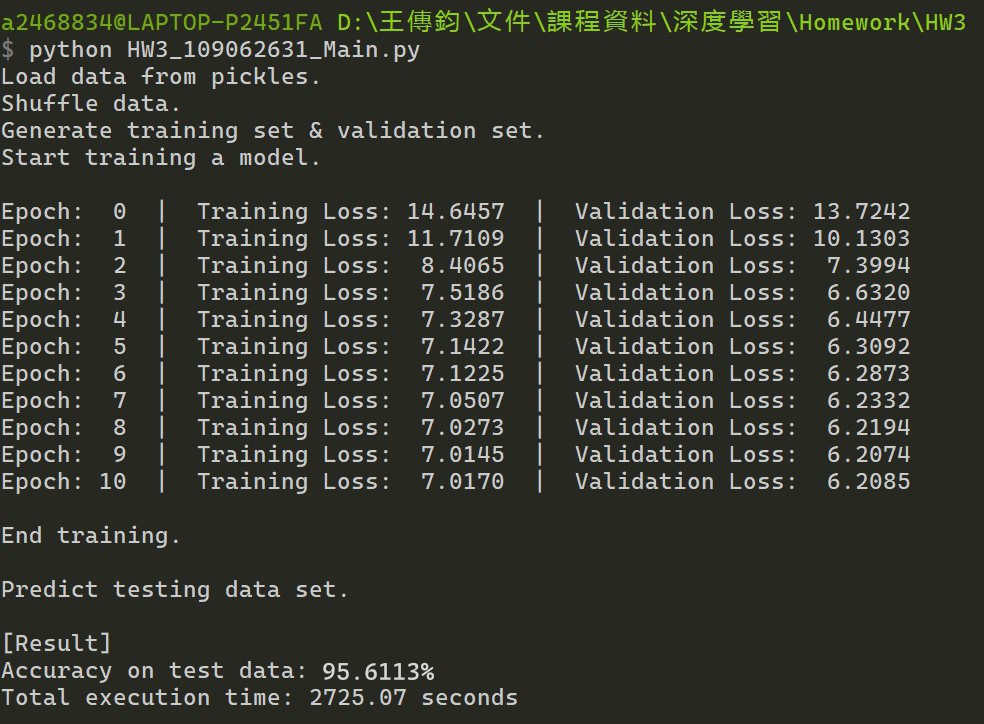
對於每張32x32畫素的照片，我先對其採用8種不同權重的4x4 kernel做convolution (stride = 1)，接著把每一個新獲得的畫素通過ReLU函數的運算，就可得到總共8 channels的29x29畫素卷積層結果。

因為接下來需要當作fully-connected layers的輸入，所以必須先進行展平（flatten）的動作。展平後的照片轉變為一個6728維的向量，因此後續fully-connected layers的「輸入層」具有6728個neurons。

經過一些嘗試之後，我發現若將learning rate調整至 左右，即可使用三層的fully-connected layers達到不錯的accuracy。因此，我採用此數量級的learning rate，搭配841個、29個與3個neurons各自所成的fully-connected layers，組成CNN後半段的結構。

最後，我的CNN對於每張輸入的照片，會輸出一個3維向量，向量當中的第一、第二、第三個維度，分別代表我的CNN判斷該圖片屬於「Carambula」、「Lychee」或「Pear」的機率大小。配合一開始已知該圖片屬於的類型，我們就可以計算出該次預測結果對應之cross-entropy loss。

經過大約2725.07秒之後，就可以得到根據上圖架構訓練出的CNN模型，具有預測測試資料集（testing set）accuracy約為95.6113%。



第二部分：訓練模型

* Forward propagation

基本上就是拿資料，放入CNN model當中，然後取得預測值；因此，流程就如同圖一的架構圖所示：圖片先經過卷積層抽取重要特徵，展平所得知張量，放入CNN後半段之三層的fully-connected layers，經過每一層的neurons進行矩陣乘法與ReLU函數轉換，進入最後一層的output layer做矩陣乘法、softmax函數轉換，最後再與label所對應之one-hot vector，一起計算cross-entropy loss。

由於convolution的計算較為複雜，因此在這邊特別提出來說明：

假設輸入的每張照片具有維度：，而我們希望透過8種不同權重的4x4 kernel做特徵抽取，也就是希望輸出的張量具有維度：，則input照片與output照片具有以下關係：

，其中 、、 是8種4x4 kernel當中的某一種。

* Backward propagation (Fully-Connected Layers)

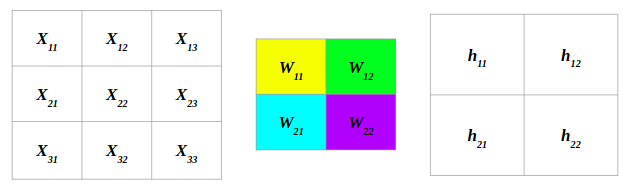
因為output layer與hidden layers的activation function不同，所以進行backward propagation時所使用的偏微分函數也不同。（以下的代表位於某層的第i個neuron，把前一層節點的輸出值經過矩陣運算之後的值）

* + Output layer
  + Hidden layer
* Backward propagation (Convolution Layer)

因為卷積層backward propagation的過程較為複雜，所以這邊捨棄本次作業的例子（即32x32畫素的照片），改用以下更小的範例來解釋此過程。

假設：我們只考慮一張3x3畫素的照片、一個2x2 kernel來進行卷積。

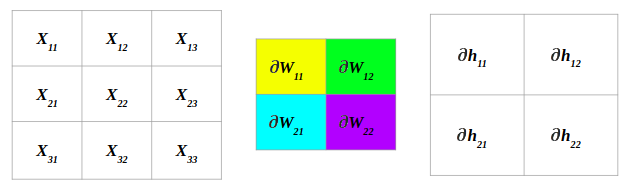
Forward prop.：



\*

=

，其中，其餘三個以此類推。

Backward prop.：（、、、是待求目標）

，其中，其餘可以類推。

第三部分：訓練結果與Loss

因為本次作業使用的資料集不是太複雜，所以CNN模型經過約10次的epoch之後，對於training loss和validation loss的變動，都趨向於收斂到定值，所以以下的圖表就以epoch總數為10的情況來繪製。

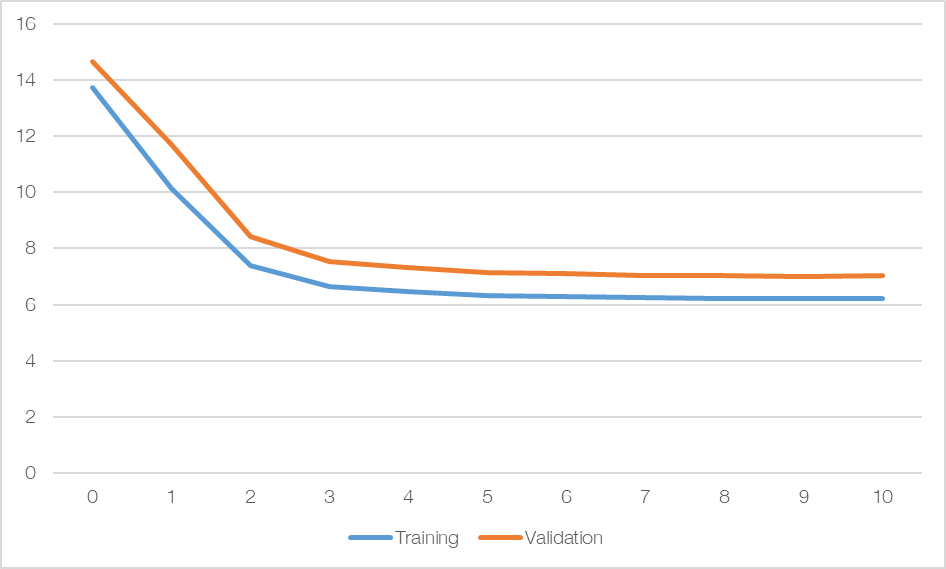


Figure: Total loss on training set & validation set

(X-axis: the number of epoches, Y-axis: C-E loss value)

第四部份：遇到之問題與解決過程

這次撰寫作業的過程中，遇到三個主要的難題：（一）如何決定有效的learning rate（二）如何決定卷積層的kernel size與kernel個數（三）如何決定CNN後半段fully-connected layers層數與每層的neurons個數。

* 第一個問題的解決過程

先前於別門課程，曾經有過一個參加深度學習競賽的慘痛經驗：由於一開始選定的learning rate過於巨大，導致模型無法收斂，然當下並無意識到問題源自於此，導致浪費許多天打轉在其他非癥結點處。

汲取以上之經驗，我體認到對於越複雜的訓練資料集，或是使用越多的neurons在fully-connected layers時，越應該注意到learning rate的初始嘗試範圍不能過大。因此，此次的作業我針對learning rate自的數量級開始嘗試，我再權衡所需的epoch不需太大以及訓練時間不應過長的情形，逐步放大learning rate至左右到達一個不錯的平衡。

* 第二個問題

由於我是第一次撰寫關於實作卷積層的程式片段，所以許多方面並無法達到計算量最小化與演算法最佳化。經過多次實驗證實：在8個process平行運算之下，進行一輪所有照片的卷積運算，在使用8種kernel時平均需 秒，使用16種kernel時平均需 秒；因此，我決定控制CNN模型有最少的卷積運算，以避免訓練時間過長和loss收斂的不隱定性。在經過多方嘗試之後，我發現只需要一層的卷積層，搭配至少8種kernel進行卷積，就可抽取出許多有意義的特徵，令CNN模型後半段fully-connected layers的負擔大幅下降，不需疊加多層fully-connected layers也能有好的accuracy。

* 第三個問題的解決過程

一般來說，適度地增加fully-connected layers的層數與每一層neuron的個數，可以讓CNN學習更複雜的圖片特徵。然而，因為我的CNN只有一層卷積層，所以無限制的增加fully-connected layers的層數只會讓整個CNN模型朝向專注於太過細節的特徵，進而產生overfitting。

在這次的作業當中，我採用每層之間縮小29倍數量的neurons來設計fully-connected layers，靈感是來自於卷積層輸出張量為8x29x29。由於最後的accuracy可以有95%上下，因此我認為這是一個不錯的選擇。