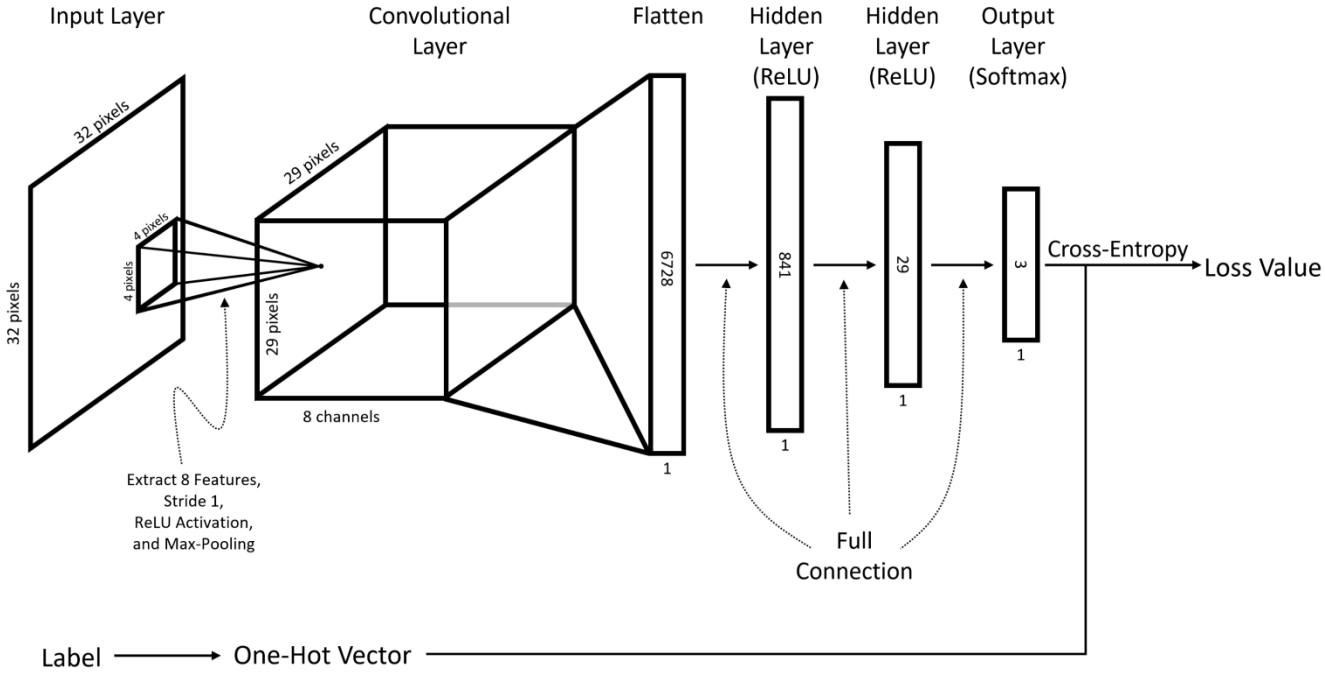
Homework 3

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 10^{-6} 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%。

```
a2468834@LAPTOP-P2451FA D:\王傳鈞\文件\課程資料\深度學習\Homework\HW3
$ python HW3_109062631_Main.py
Load data from pickles.
Shuffle data.
Generate training set & validation set.
Start training a model.

Epoch:  0  |  Training Loss: 14.6457  |  Validation Loss: 13.7242
Epoch:  1  |  Training Loss: 11.7109  |  Validation Loss: 10.1303
Epoch:  2  |  Training Loss:  8.4065  |  Validation Loss:  7.3994
Epoch:  3  |  Training Loss:  7.5186  |  Validation Loss:  6.6320
Epoch:  4  |  Training Loss:  7.3287  |  Validation Loss:  6.4477
Epoch:  5  |  Training Loss:  7.1422  |  Validation Loss:  6.3092
Epoch:  6  |  Training Loss:  7.1225  |  Validation Loss:  6.2873
Epoch:  7  |  Training Loss:  7.0507  |  Validation Loss:  6.2332
Epoch:  8  |  Training Loss:  7.0273  |  Validation Loss:  6.2194
Epoch:  9  |  Training Loss:  7.0145  |  Validation Loss:  6.2074
Epoch: 10  |  Training Loss:  7.0170  |  Validation Loss:  6.2085

End training.

Predict testing data set.

[Result]
Accuracy on test data: 95.6113%
Total execution time: 2725.07 seconds
```

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：$(C_{in}, H, W) = (1, 32, 32)$，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：$(C_{out}, H_{out}, W_{out}) = (8, H_{out}, W_{out})$，則 input 照片與 output 照片具有以下關係：

$$\text{output}(h_{out}, w_{out}) = \sum_{m=0}^{31}\sum_{n=0}^{31} input(m,n)K^T(m - h_{out}, n - w_{out})$$

，其中 $h_{out} \in \{0, 1, \cdots, H_{out}\}$、$w_{out} \in \{0, 1, \cdots, W_{out}\}$、$K$ 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 z_i 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

    $$\frac{\partial Loss}{\partial z_i} = \text{softmax}(z_i) - \mathbb{I}(i == \text{label}) \text{ for } i \in \{0, \cdots, 3\}$$

    $$\mathbb{I}(i == \text{label}) = \begin{cases} 1 \text{ , if } i \text{ is equal to label} \\ 0 \text{ , otherwise} \end{cases}$$

  - Hidden layer

$$\frac{\partial Loss}{\partial z_i} = \mathbb{I}(z_i > 0)$$

$$\mathbb{I}(z_i > 0) = \begin{cases} 1 \text{ , if } z_i \text{ is greater than one} \\ 0 \text{ , otherwise} \end{cases}$$

- 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.：

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

$*$

W_{11}	W_{12}
W_{21}	W_{22}

$=$

h_{11}	h_{12}
h_{21}	h_{22}

，其中$h_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$，其餘三個以此類推。

Backward prop.：（∂w_{11}、∂w_{12}、∂w_{21}、∂w_{22}是待求目標）

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

∂W_{11}	∂W_{12}
∂W_{21}	∂W_{22}

∂h_{11}	∂h_{12}
∂h_{21}	∂h_{22}

，其中$\partial w_{12} = x_{12}\,\partial h_{11} + x_{13}\,\partial h_{12} + x_{22}\,\partial h_{21} + x_{23}\,\partial h_{22}$，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。

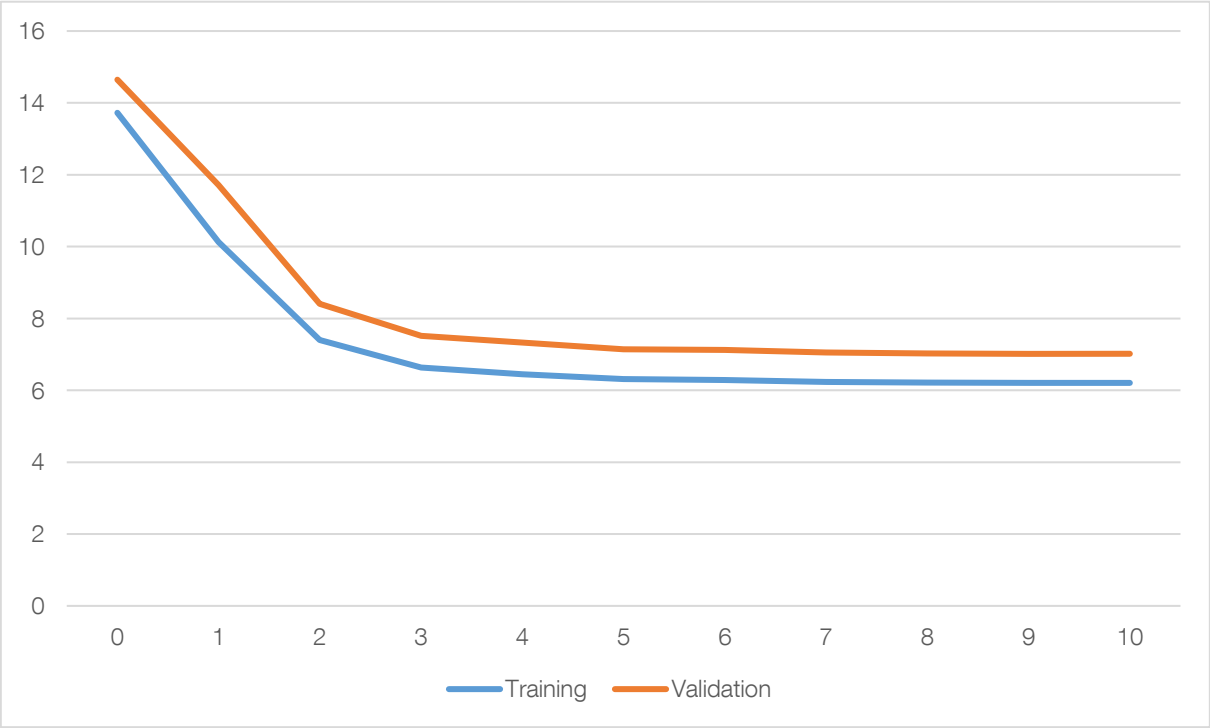


Figure: Total loss on training set & validation set
(X-axis: the number of epoches, Y-axis: C-E loss value)

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10^{-8}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10^{-5}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 56.73 秒，使用 16 種 kernel 時平均需 103.48 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。