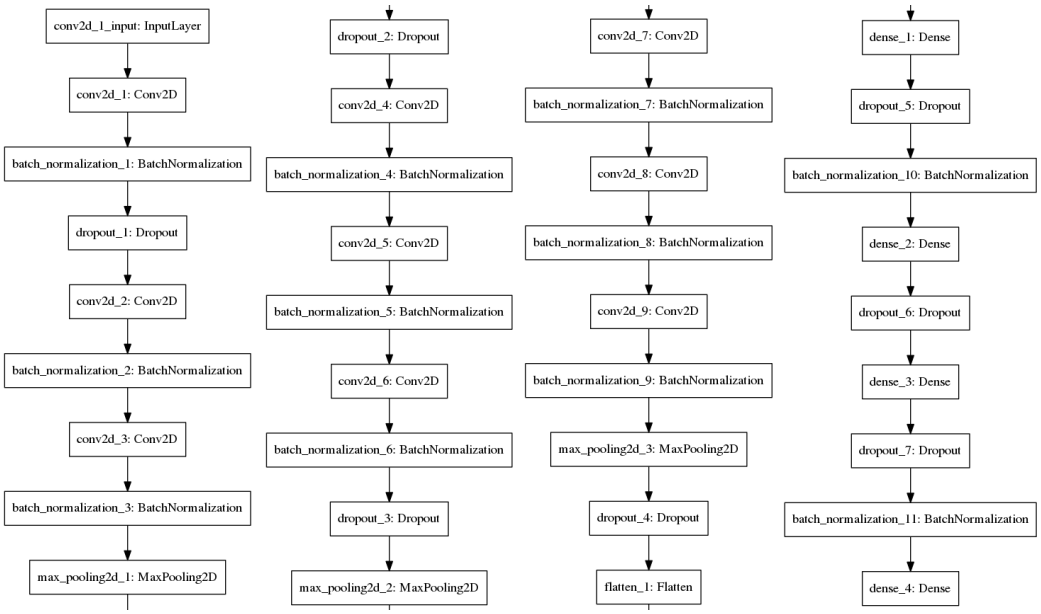
HW3: Image Sentiment Classification

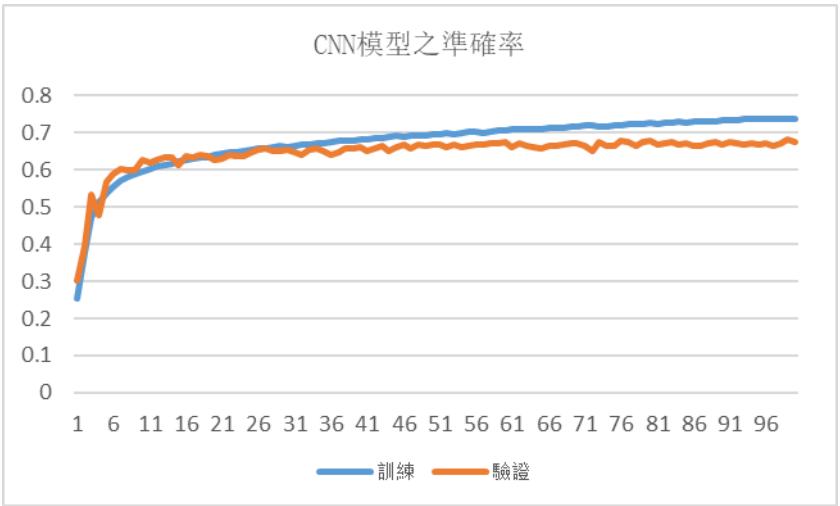
學號：R06922030 系級： 資工碩一 姓名：傅敏桓

1. 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？

(Collaborators: 自己)

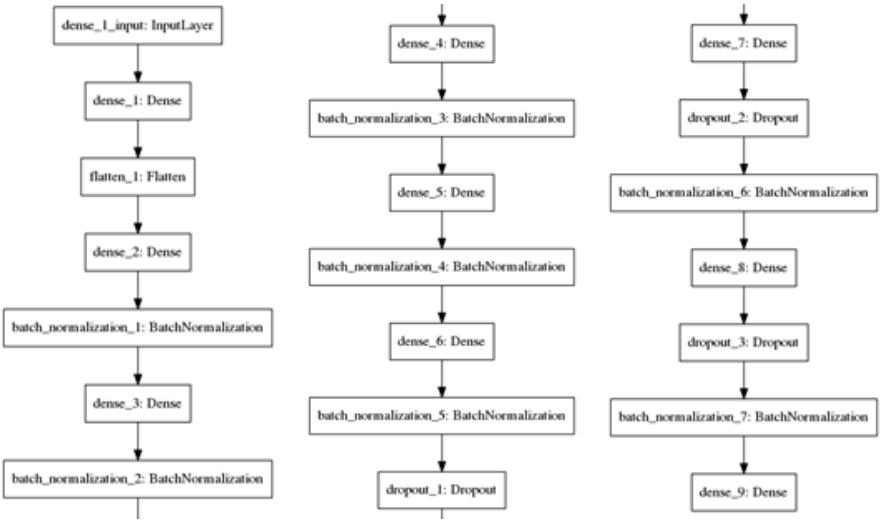


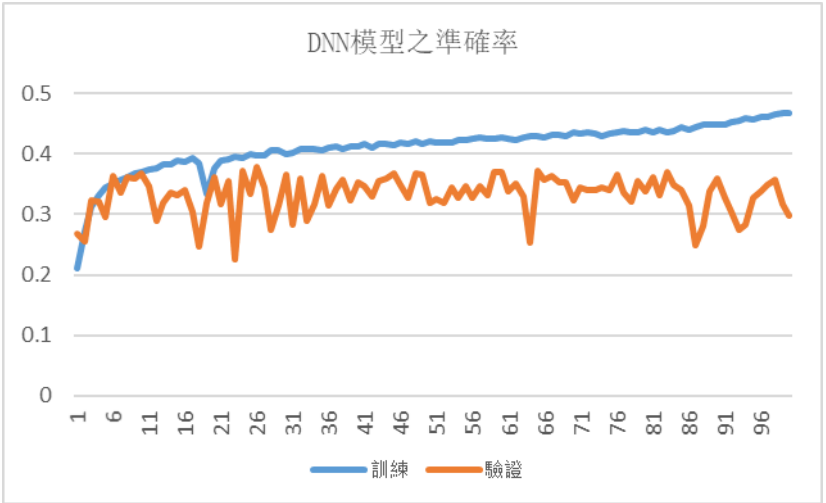
本次使用的 CNN 模型架構如上圖所示。每三層卷積層搭配一層池化層，共有九層卷積層；將卷積層的輸出壓平後通過數層全連接層，再透過 softmax 輸出得到最後的分類結果(另外，插入適量的批標準化層與 Dropout 層調適模型)，model.summary()顯示參數總量 787,143。訓練時將測試集隨機排序後切最後百分之 20 作為驗證集，使用 Keras 內建的 ImageDataGenerator 實做數據增強 (Data augmentation)，對原有數據進行平移、旋轉、翻轉、縮放等操作。訓練過程如下圖所示，以批大小 32 訓練 100 個迭代，以交叉熵為損失函數，透過 Adam 進行參數更新，取驗證損失最小之參數模型。準確率以 Kaggle private 分數為據，在測試資料上為 68.793%。



2. **承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？**
(Collaborators: 自己)

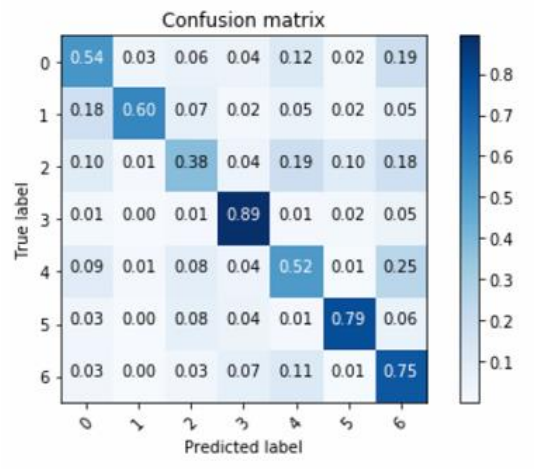
由於全連接模型需要的參數量較 CNN 模型多，使用參數量接近之全連接模型時，在盡可能維持相同架構的情況下，不得不將神經元的數量減少，可能因此模型的表現不佳，甚至根本難以完成訓練過程，在驗證集得到的準確率最高僅 37.77%。上課中有提到 CNN 實際上是比較節省參數的模型架構，從本題的觀察可以發現，在參數相近的狀況下，針對影像分類這個任務，使用 CNN 可以有較好的表現。本題使用的 DNN 參數總量為 799,525，模型訓練設定同 1.所述，架構示意圖如下。





3. **觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]**
(Collaborators: 自己，參考 scikit-learn 文檔)

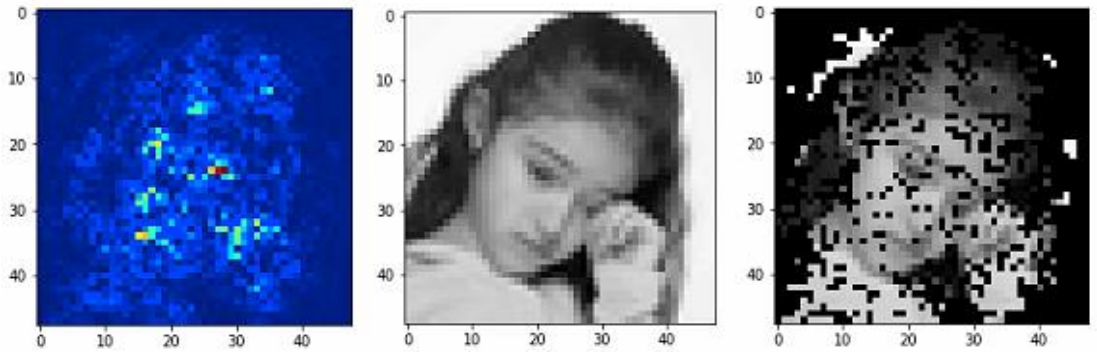
實驗設定：測試集隨機排序後切最後百分之 20 作為驗證集，以 1.所述之模型架構訓練 50 個迭代，取訓練過程中驗證集交叉熵最小之模型參數，再以同樣的驗證集經由訓練後之模型進行預測。以真實標記與模型預測標記繪製之混淆矩陣如下圖。



此模型在類別 3（高興）預測得最好，有 0.89 的正確率，而在類別 2（恐懼）預測的最差，僅有 0.38 之正確率，推測應該是訓練資料中類別 3 的影像特徵的同質性較高，而其他辨識度較低的類別則有可能是同類影像之間特徵差異性較大。另外，觀察對角線以外數值較大之部分，可以發現哪些類別之間容易被分類錯誤，其中類別 0、2、4 被誤分類至類別 6（中立）的比例都不低，類別 2 也同樣容易被誤分類至類別 4。

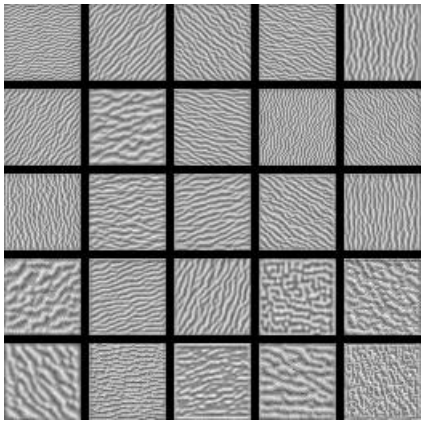
4. **從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？**
(Collaborators: 自己)

本題中使用的是自 3.所述之驗證集取出的某張類別 4 的影像。以下由左至右分別為該圖的 saliency map、原始影像及經過遮罩後的影像。可以發現梯度在人臉的部分、或與人臉有連動的部分相對較大，尤其是在各臉部器官的位置，而背景的白色部分梯度相對較小。由此推測模型在進行分類問題時，主要聚焦於影像中人臉的部分。

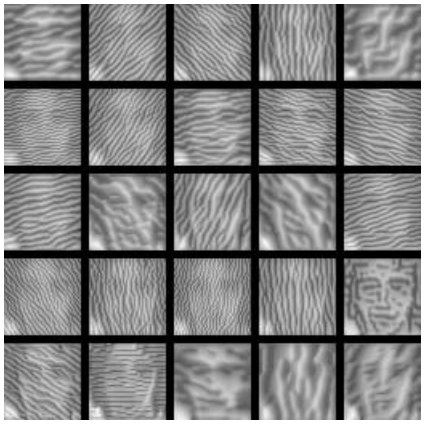


5. **承(1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate。**

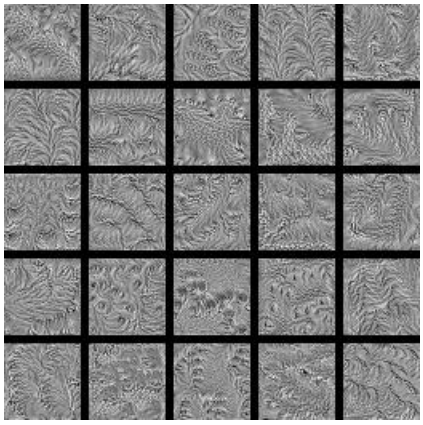
(Collaborators: 自己，參考 Keras 的教學)



第二層卷積層，從噪音開始



第二層卷積層，從特定影像開始



第六層卷積層，從噪音開始

本題試圖透過梯度上升的方法找出最容易激發特定層濾波器之影像，梯度上升 100 個迭代後按照損失大小排序的結果如圖所示。我們可以觀察到靠近輸入端的卷積核抽取的是一些比較簡單的特徵，大多為條紋狀或點狀，而靠近輸出端的卷積核則略顯複雜。

參考資料

http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html