

Machine Learning 2017 Spring

Homework 3 Report

學號：B03902048

系級：資工三

姓名：林義聖

1. (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？

答：Figure 1 是我的 CNN model 的基本架構圖。這個模型的總參數數量為 6,746,215。部分細節無法完全呈現在上圖中，則一一列舉在下方：

- Validation：我將總數 28709 筆的訓練資料切成兩份，其中前 3000 筆為 validation 用，其餘的用來訓練模型。
- Image Pre-processing：我先將訓練模型用的資料，利用水平翻轉變成兩倍資料量 (51418 筆)，接著利用 Keras 的 ImageDataGenerator 來增加訓練資料。我使用了旋轉、水平位移、垂直位移、縮放和錯切 (推移)。
- Normalization：我將所有資料的亮度範圍，從 0 到 255，縮放至 0 到 1 之間。
- ReLU layer：在所有 Convolution Layer 之後，我都接上 Leaky ReLU 來作 elementwise activation，並設定 α 為 0.05。在這之後，Convolution Layer 的輸出 x ，會變成

$$f(x) = \max(x, \alpha x)$$

- Batch Normalization：在所有 Activation Layer 之後 (除了 output layer)，我再接上 Batch Normalization Layer。如此一來，可以加快訓練過程的收斂速度，也可以避免 Overfitting。
- Optimizer：我使用 Adam 作為模型訓練的 Optimizer。
- Batch：我將 batch size 設定為 128。

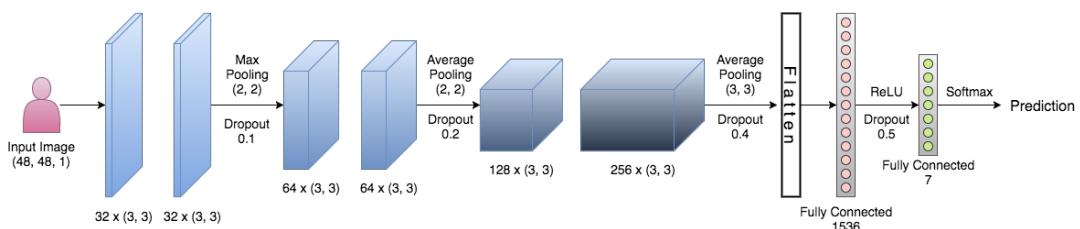


Figure 1: CNN model structure

訓練過程中，我一路將在 validation set 上有最高準確率的模型儲存下來，最終在 258 個 epochs 後，得到 0.7057 的準確率，而上傳之後，在 public set 上有 0.7127 的準確率。

Figure 2a 和 Figure 2b 分別是訓練過程的 loss 和 predictiton accuracy 變化圖，根據這個分析表，我們可以看出，valid loss 在 10 到 20 個 epochs 間就開始上升，但是 valid accuracy 仍然可以在後續的訓練過程中，一點一點持續上升，我推測，這是做了 Image Pre-processing 帶來的效果。雖然隨著 valid loss 的上升，模型好像已經 overfit 在 training set 上，然而，透過 ImageDataGenerator，我們的訓練過程並沒有一筆完全固定的 training data，也因此，模型不至於 overfitting，所以可以在 validation set 上預測出好結果。

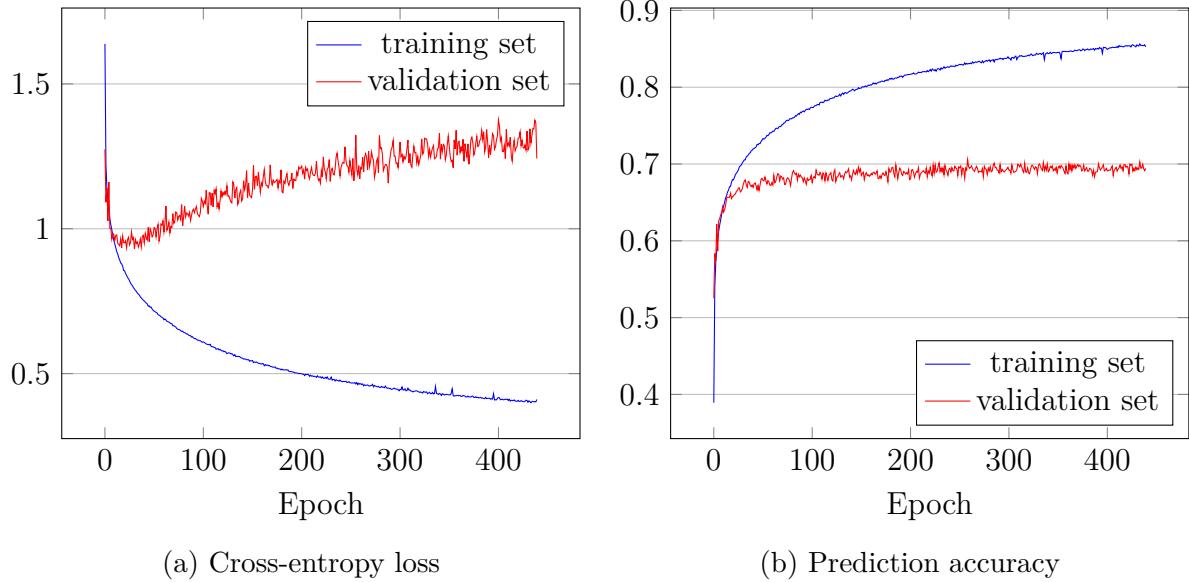


Figure 2: CNN model training

2. (1%) 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？

答：Figure 3 是我的 DNN model 的基本架構。這個模型的總參數數量為 6,581,255，與前一題的 CNN model 的 6,746,215 相去不遠。而這個 DNN model 的實作的細節，除了下方提及的 Activation Layer 和 Dropout 以外，其餘皆與前述的 CNN model 相同，好能比較他們之間的結果。

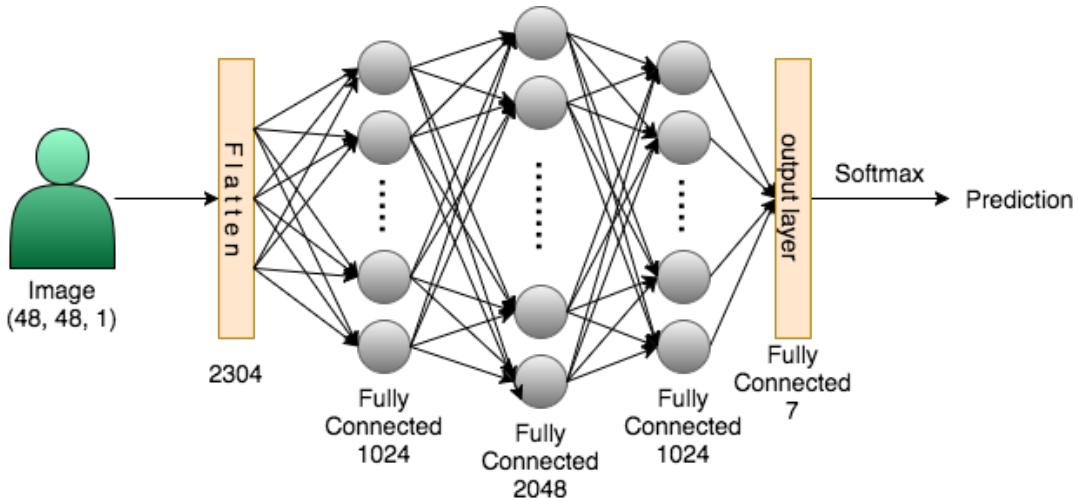


Figure 3: DNN model structure

- ReLU layer：在所有 hidden layer 之後，我都接上 ReLU 來作為 activation function。因此，前一層 hidden layer 的輸出 x 會變成

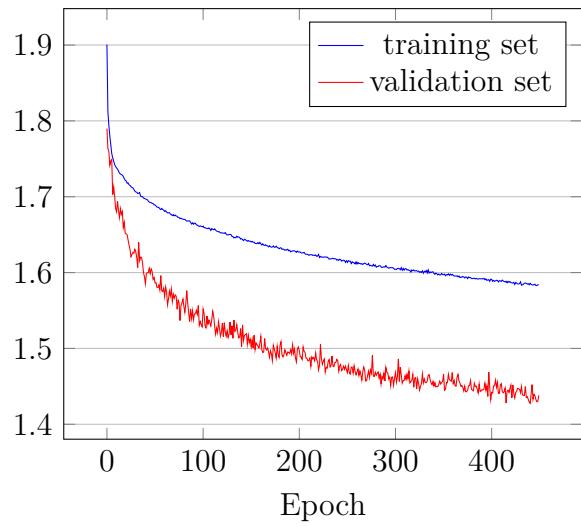
$$f(x) = \max(0, x)$$

- Dropout：在 Figure 3 圖中的三層 hidden layer 之後，我分別加上 0.3, 0.4, 0.5 的 dropout。

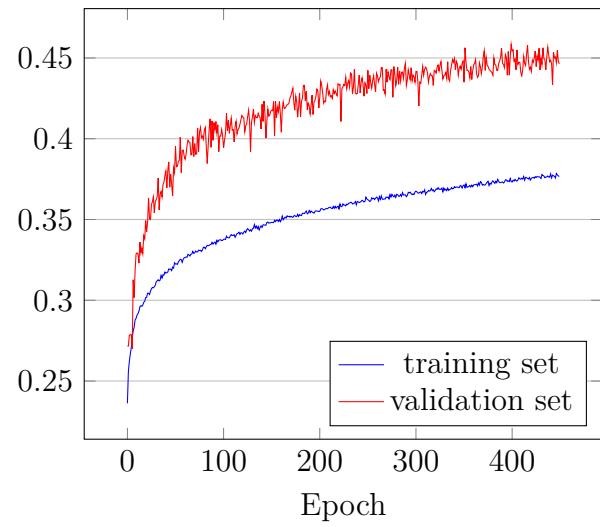
訓練過程中，我記錄了所有 training set 和 validation set 上的 loss 和 accuracy，最終在 399 個 epochs 的時候，在 validation set 上得到 0.4583 的準確率。Figure 4a 和 Figure 4b 分別是訓練過程的 loss 和 prediction accuracy 變化圖。

透過觀察這些變化圖，我們可以歸結兩個 DNN model 與 CNN model 不同的點：

- 在經過相同回合數後，我們可以明顯看出，DNN model 還沒有收斂，藉由 dropout，validation loss 仍顯著地低於 training loss，因此可以推測 model 還有相當大的進步空間。此外，雖然經過了相同數目的 epochs，DNN model 預測的準確率完全無法與 CNN model 相比，即使兩個模型的參數數量相近。經過這樣的比較，我們可以斷定，在表情辨識這個任務上，CNN model 明顯地優於 DNN model，不僅能收斂地更快，也能達到更高的準確度。值得一提的是，DNN model 的訓練相對的快，可以花大約一半的時間，便達到同樣數目的 epochs。
- 繼續訓練這個模型，達到 650 epochs 後，這時的訓練時間已經與 CNN model 相去不遠，然而，DNN model 的訓練還沒收斂，看起來仍有不小的進步空間。透過觀察 Figure 5，我們可以發現，它相較於 Figure 4 中的曲線，幾乎沒有明顯改變，因此可以推斷，DNN model 要達到收斂，可能要再花上相當長的時間。所以，即使 DNN model 仍有進步的空間，要訓練好它所花的時間早已遠超 CNN model，且未必能達到更高的準確度。

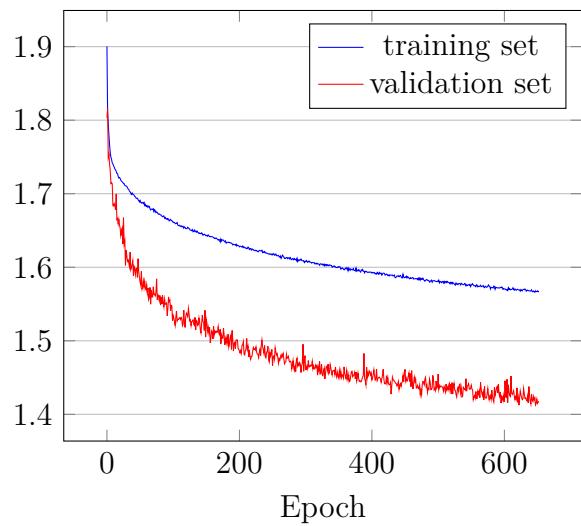


(a) Cross-entropy loss

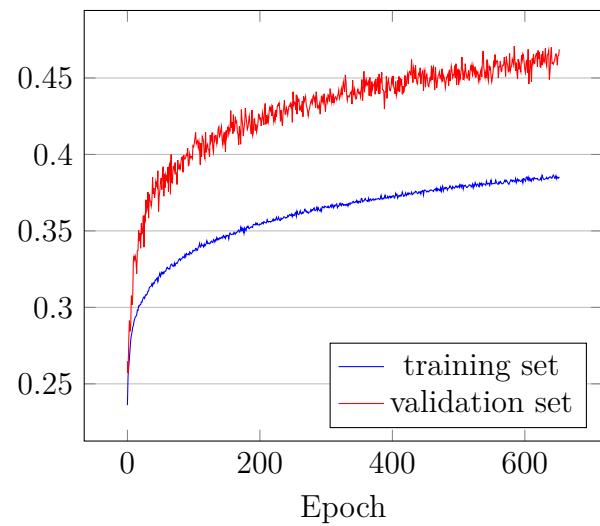


(b) Prediction accuracy

Figure 4: DNN model training



(a) Cross-entropy loss



(b) Prediction accuracy

Figure 5: DNN model training to 650 epochs

3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？(繪出 confusion matrix 分析)

答：我使用 sklearn 套件當中的 `confusion_matrix` 函式來算出 confusion matrix，並繪製出 Figure 6。

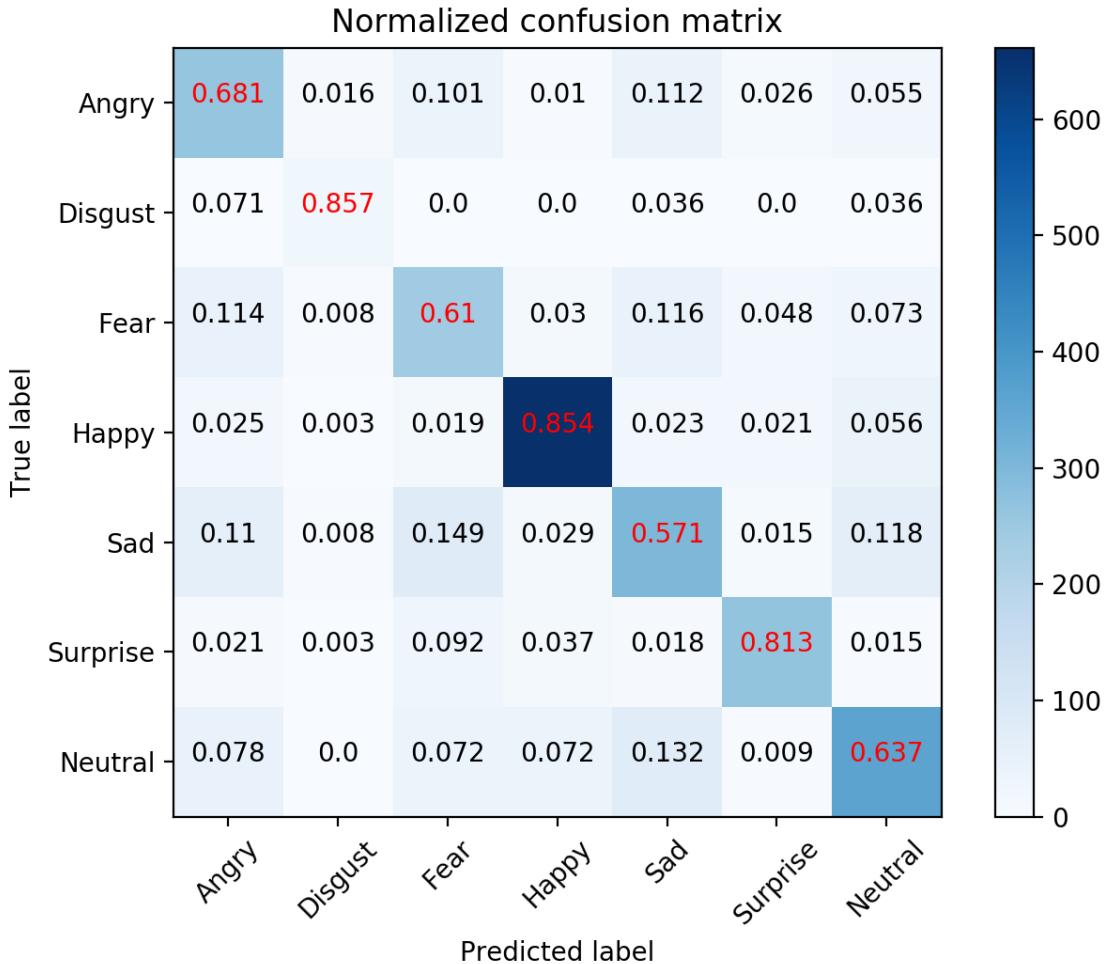


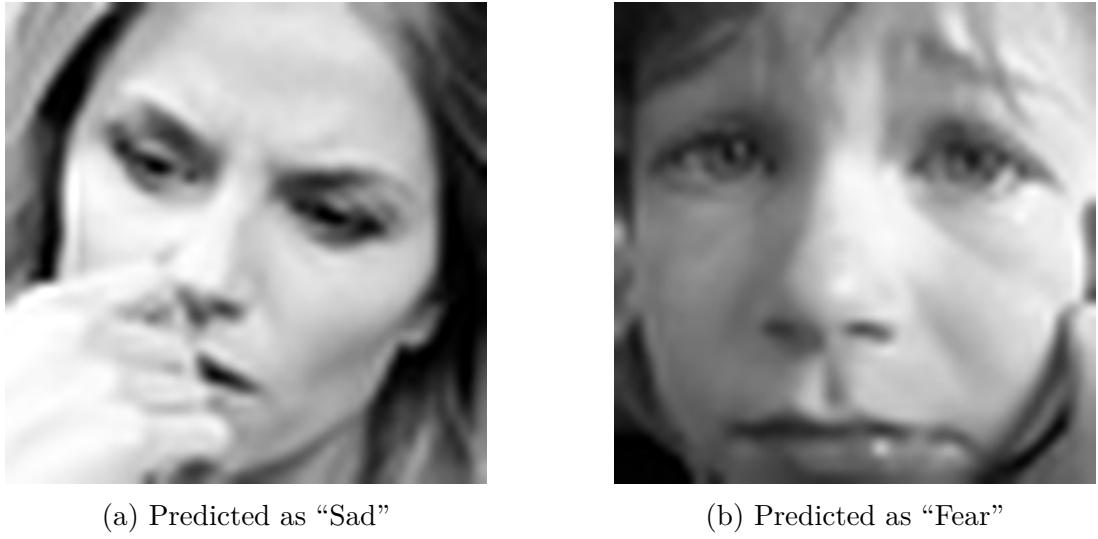
Figure 6: Normalized Confusion Matrix

觀察 confusion matrix 時，可以發現表情 Sad 和 Fear 之間，以及 Sad 和 Neutral 之間都很容易預測錯誤。先以 Sad 和 Fear 為例，以下有兩張來自 training set 的圖片，其中 Figure 7a 被預測成 Sad，然而它的標記其實是 Fear；而 Figure 7b 被預測成 Fear，然而它的標記其實是 Sad。

我認為 Figure 7a 是很容易誤判的，連我認為它是屬於 “Sad”。至於 Figure 7b，我覺得也不容易判斷，難以斷定是屬於 Sad 還是 Fear。而 Table 1 是 model 判斷出的那兩張圖片表情的機率分佈，從這個分佈，也可清楚看出，人容易搞混的圖片，model 也難以精確地做判斷。

Table 1: Probability distribution

| # | Angry | Disgust | Fear | Happy | Sad | Surprise | Neutral |
|-----------|-------|---------|------|-------|------|----------|---------|
| Figure 7a | 0.06 | 0 | 0.07 | 0 | 0.87 | 0 | 0 |
| Figure 7b | 0 | 0 | 0.48 | 0 | 0.44 | 0 | 0.08 |



(a) Predicted as “Sad”

(b) Predicted as “Fear”

Figure 7: Misprediction between class “Sad” and “Fear”

4. (1%) 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？

答：我從 `train.csv` 裡面的前 3000 筆資料當中，選了第 11, 15 和第 30 張圖片來繪製 saliency maps，分別是 Figure 8, Figure 9 和 Figure 10。

首先，Figure 8 是一張標記為“生氣”的表情。所以，我認為判斷的重點在雙眼附近的皺紋，而 Figure 8c 中的 heatmap 上高亮的區域正是那裡。而 Figure 9 是一張標記為“快樂”的圖片。我認為判斷的重點是雙眼，以及嘴角的弧度，顯然地，從 Figure 9b 來看，model 抓到的重點與我的判斷非常相像。最後，Figure 10 是一張標記為“驚訝”的圖片。明顯地，瞪大的雙眼和大開的嘴巴是我們判斷的依據，而 Figure 10b 中也清楚地顯示了模型是 focus 在這個部分。

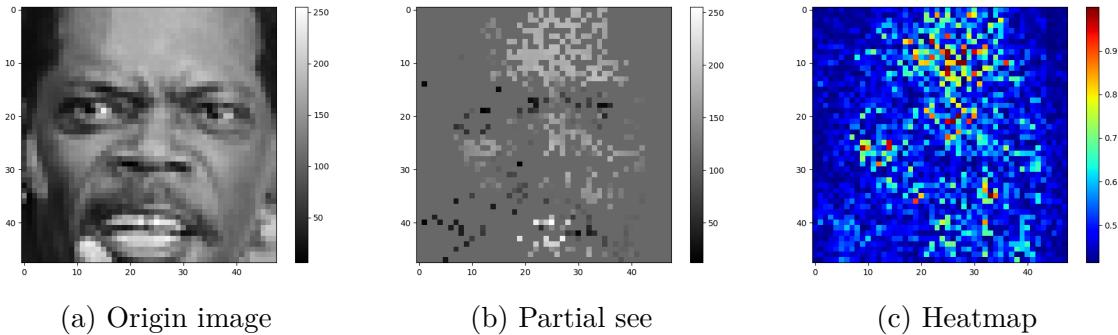


Figure 8: An “Angry” image in `train.csv`

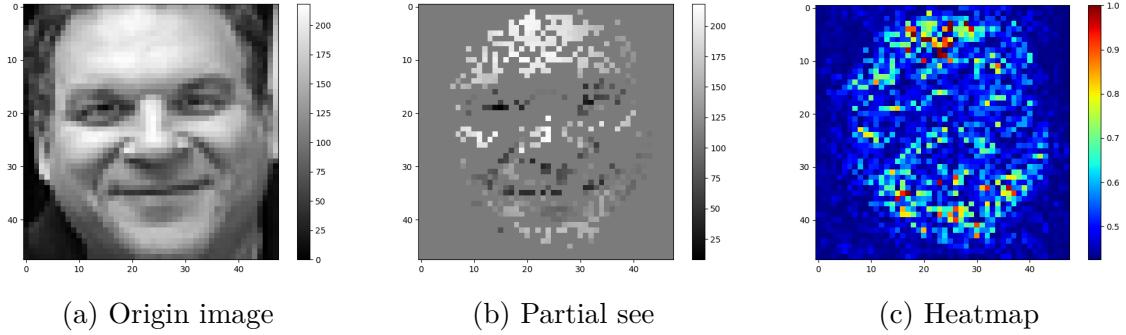


Figure 9: A “Happy” image in train.csv

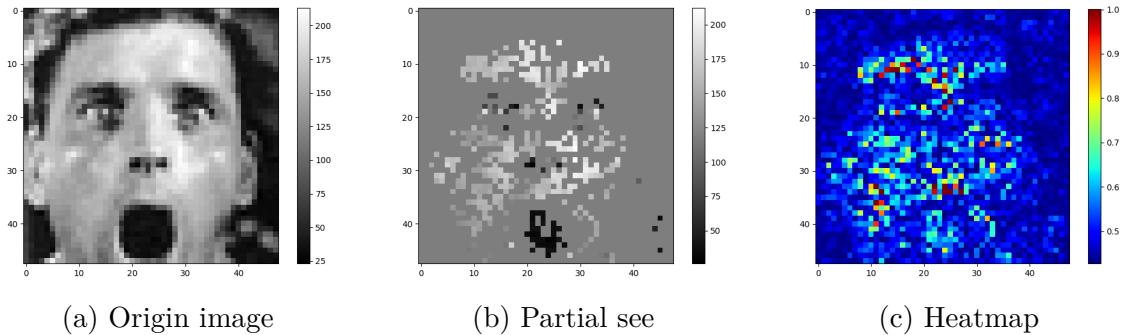
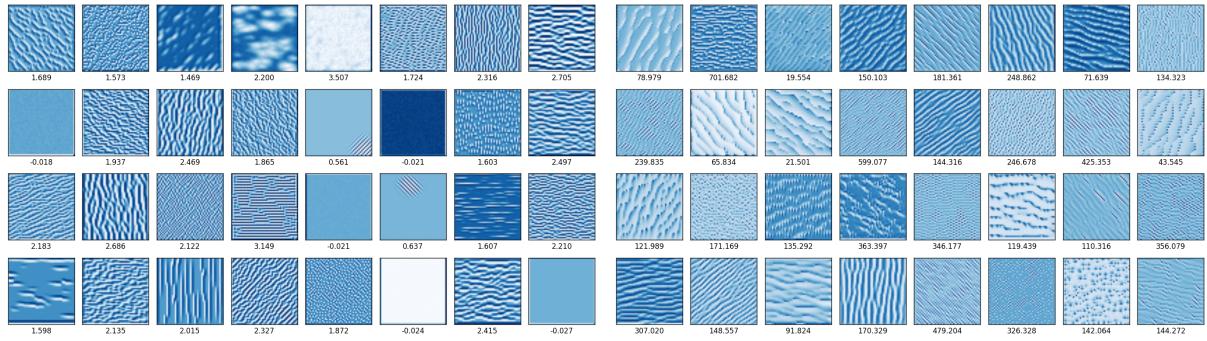


Figure 10: A “Surprise” image in train.csv

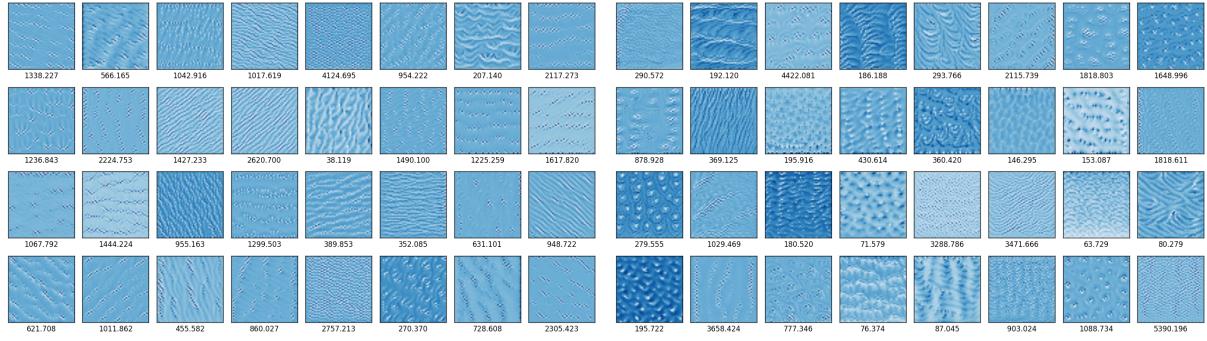
5. (1%) 承 (1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate。

答：首先是 Figure 11，這整組圖片都是透過 gradient ascent 得到最容易激活 Leaky ReLU Layer 的圖片。我們可以觀察到，從 Figure 11a 上，也就是第一層當中，大多數圖片紋理都比較粗糙。也就是說，底層的 filter，基本上是在抓取圖片當中臉部的輪廓。而到了較上層的 filter，也就是在 Figure 11d 中，已經可以看到一些很重要的臉部特徵，如：眼睛、耳朵和皺紋等等。



(a) Layer: leaky relu 1 (epoch 60)

(b) Layer: leaky relu 2 (epoch 60)

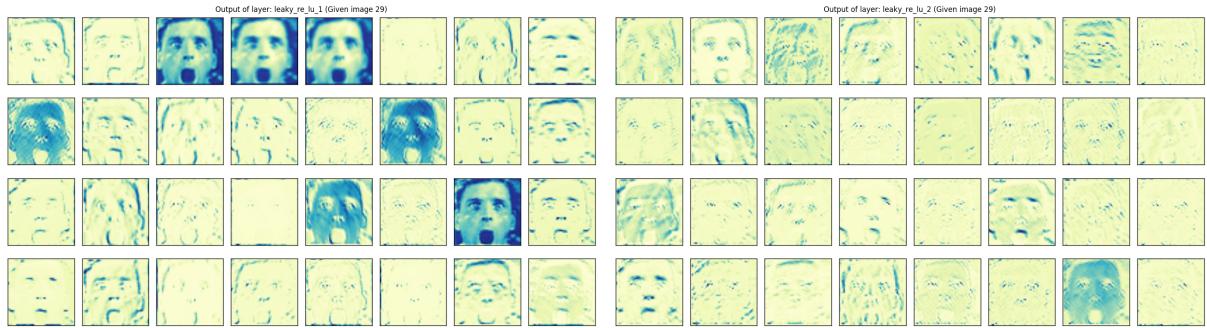


(c) Layer: leaky relu 3 (epoch 60)

(d) Layer: leaky relu 4 (epoch 60)

Figure 11: Gradient ascent from white noise

而 Figure 12，則是將 `train.csv` 當中的第 30 張圖片（屬於模型訓練時的 validation set）餵進去後，第一和第二層 Leaky ReLU Layer 輸出的結果。



(a) Layer: leaky relu 1

(b) Layer: leaky relu 2

Figure 12: Filters output (image id 29)

Bonus (1%) 從 training data 中移除部份 label，實做 semi-supervised learning。

答：為了實作 semi-supervised learning，我從 training set 當中挪去了 12000 筆資料的 label，而 validation set 則是與前述相同的 3000 筆資料，然後用剩餘的資料開始訓練模型。模型的架構與第一題描述的完全相同，好能夠比較兩者間的差異。

訓練過程中，每訓練完一個 epoch，我就對沒有 label 的圖片進行預測，如果預測出來，屬於某個類別的機率大於 50%，我就將它加入 training set。Figure 13 呈現的是 semi-supervised learning 的訓練過程，如果比較 Figure 13b 和 Figure 2b，可以明顯看出，前者明顯弱了不少，即使在相同 epoch 數量下，準確率也與後者有相當差異。而 Figure 14 則是訓練過程中，剩餘的 non-label 資料數量。

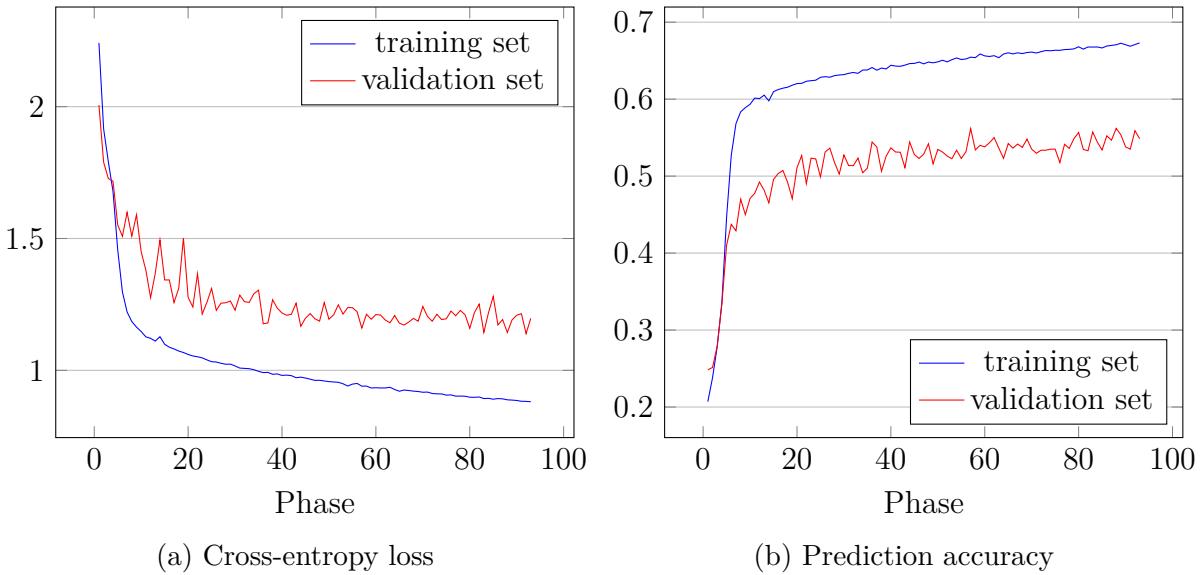


Figure 13: Semi-supervised training

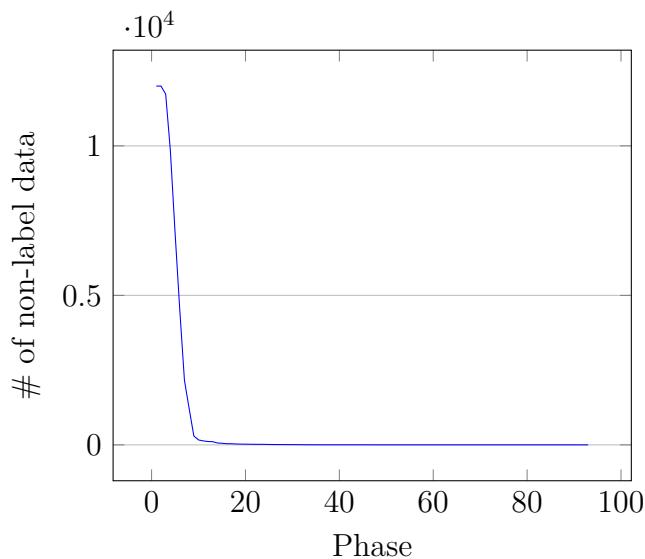


Figure 14: Semi-supervised learning (non-label data)

Bonus (1%) 在 Problem 5 中，提供了 3 個 hint，可以嘗試實作及觀察（但也可以不限於 hint 所提到的方向，也可以自己去研究更多關於 CNN 細節的資料），並說明你做了些什麼？(完成 1 個：+0.4%，完成 2 個：+0.7%，完成 3 個：+1%)

(a) 透過視覺化 Filters 分析一個較弱的模型。

答：相較於在 Problem 5 中使用的模型，我所比較的模型在 valid set 上只有 52% 的準確率，相較於前面提及的模型（在 valid set 上有 70% 準確率）著實弱了不少。Figure 15 是從 white noise 圖片透過 gradient ascent 後得到的。而 Figure 16 與 Problem 5 中 Figure 12 所使用的是同一張圖片。

經過一番觀察和比較，我認為從 Figure 11d 和 Figure 15d 兩張圖中，可以看出後者的 filters 有較多仍是大範圍分布的紋理，而較少有可被明顯識別的臉部特徵。至於 Figure 12 和 Figure 16 之間，可以看出後者多呈現出原先圖片上的人的臉部輪廓，沒有得到更高層次的臉部特徵，相較之下，從前者的 filter output 裡，我們較不容易看出原圖片的個人臉部輪廓。因此，可以推斷前者在表情特徵的抽取上做得更好。

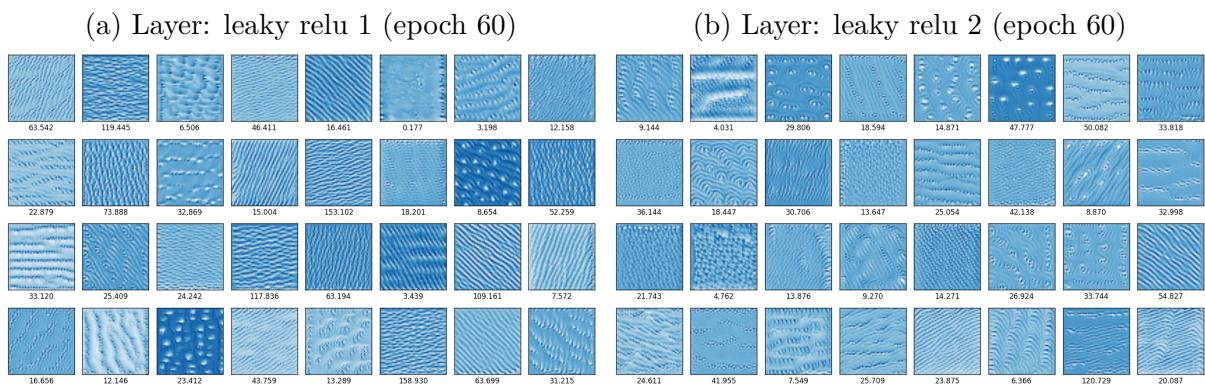
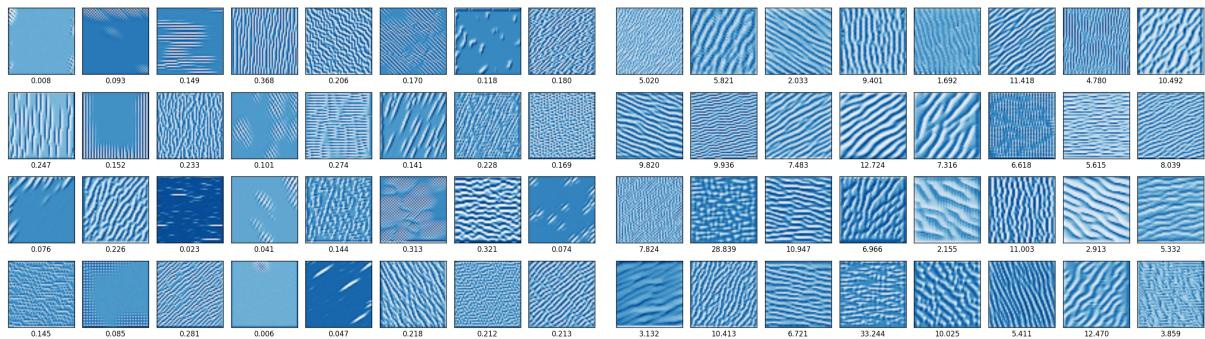
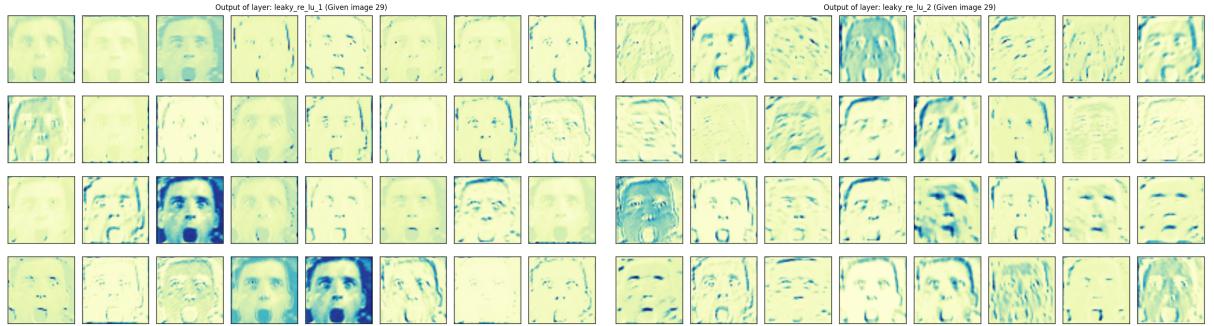


Figure 15: Gradient ascent from white noise (weak model)



(a) Layer: leaky relu 1

(b) Layer: leaky relu 2

Figure 16: Filters output (weak model, image id 29)

(b) 找出最能激活特定類別的圖片。

答：參考助教給的網站，我從 white noise 圖片，透過 gradient ascent 得到一張最能激活“Happy”類別的圖片（模型預測的機率分佈為 Table 2），即 Figure 17。為了使圖片能夠有些看得出的紋路，我每隔幾個回合就加上 gaussian filter 和 median filter，不然原先出來的圖片看起來只是一堆雜點。然而，從這張圖片上我們還是難以看出任何足以辨識出的表情。

Table 2: Probability distribution

| # | Angry | Disgust | Fear | Happy | Sad | Surprise | Neutral |
|-----------|-------|---------|------|-------|-----|----------|---------|
| Figure 17 | 0.0 | 0.0 | 0.0 | 0.99 | 0.0 | 0.0 | 0.0 |

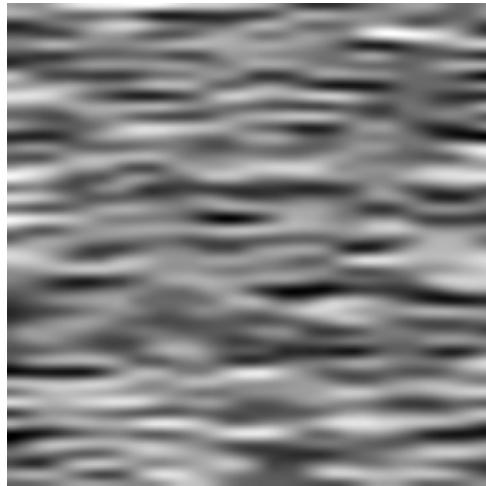


Figure 17: A happy image (ascent from white noise)

(c) 從一張正常照片，透過 gradient ascent 產生 adversarial image。

答：跟上一題的做法類似，只是這次是將一張 validation set 裡的圖片作為 input，透過 gradient ascent 得到一張新的圖片，而被模型判斷為另一個類別。我選擇 validation set 裡的第十五張圖片，即 Figure 18a，而模型有 99% 信心判定它屬於“Happy”，經過 2500 個 epoch 的 gradient ascent 後，得到 Figure 18b，而它被模型判定為“Sad”，且同樣有 99% 的信心。



(a) A “Happy” image



(b) A “Sad” image

Figure 18: A image from “Happy” to “Sad” (image id 29)