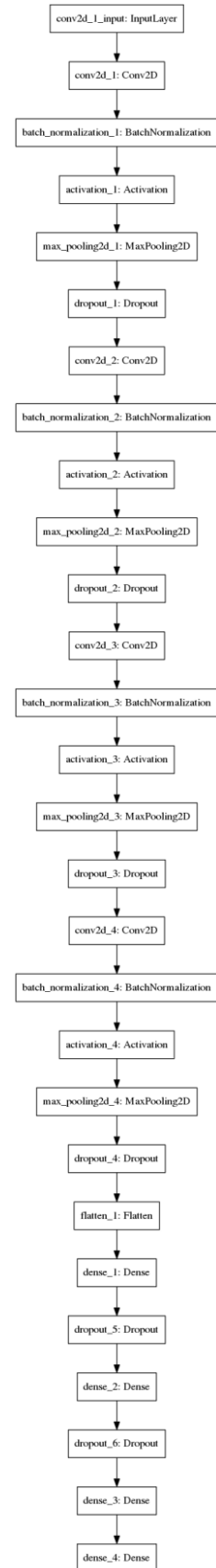
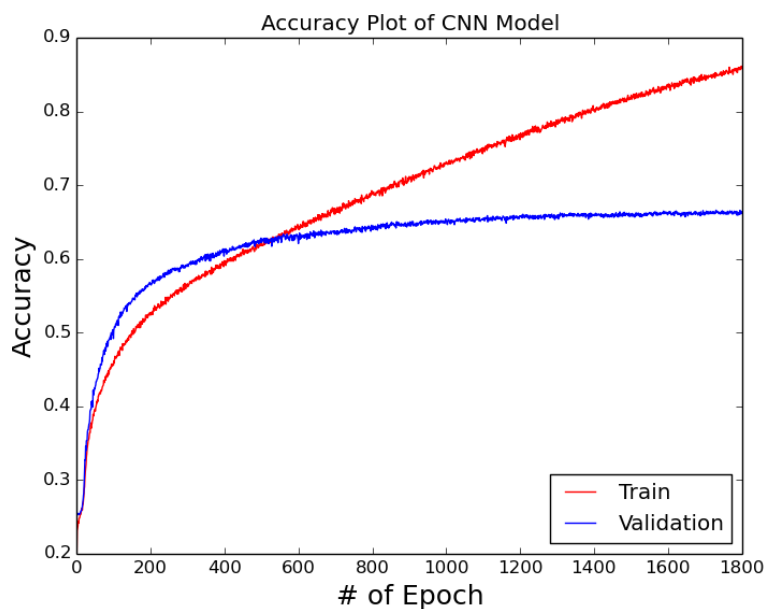


學號: R05921086 系級: 電機所碩一 姓名: 邱名彥

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: We use the Stochastic Gradient Descent method (SGD) found in the Keras library. Using a small learning rate of 0.00001 and iterating with 1800 epochs, we can achieve a Kaggle score of 0.65896 or 65.8% accuracy rate on the public data. We can see that at epoch 500, the validation begins to converge to its optimal point due to reasons such as overfitting. Further adjustments to the CNN model (as shown in the architecture and model summary) can prove to increase accuracy by adjusting the number of convolutional layers, convolutional parameters.



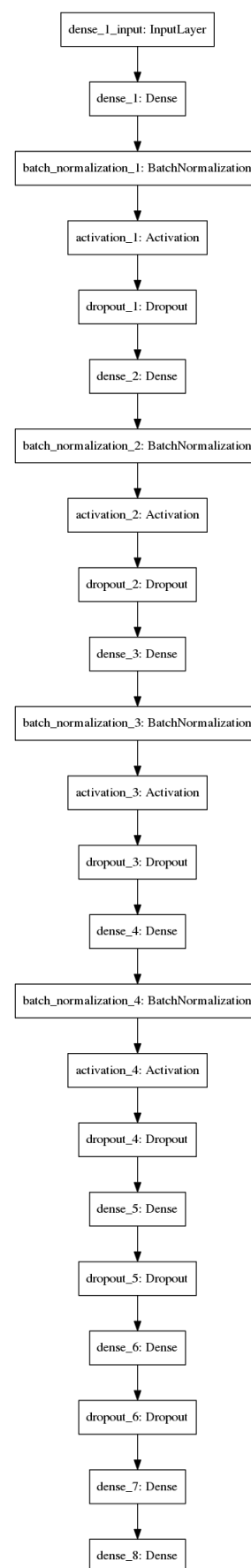
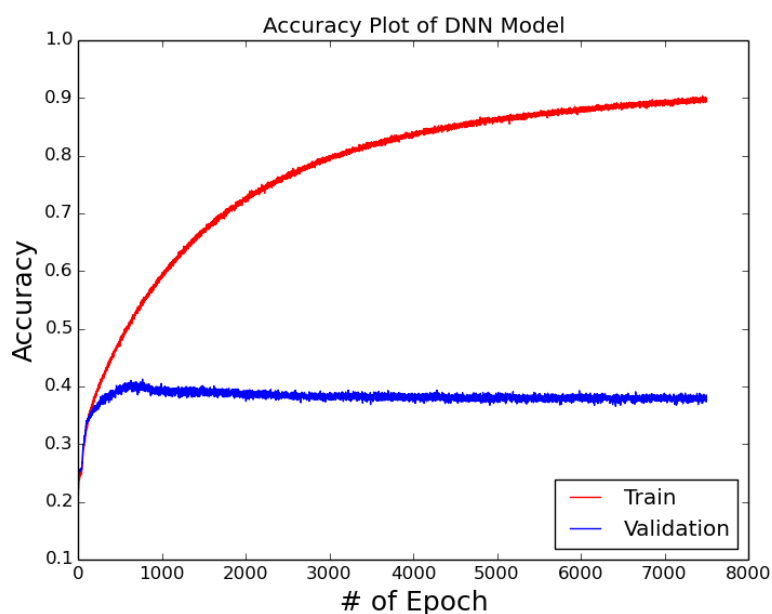
2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

答：To convert our CNN model to a general DNN model, we replace all the Conv2D layers with Dense layers and remove the MaxPooling2D layers. We scale up layer parameters accordingly to try and preserve the same model architecture while trying to maintain the same number of parameters. With our DNN model, we observe that accuracy is significantly reduced and begins to converge around epoch 500 like our CNN model. Our DNN model has a Kaggle score of 0.38702 or 38.7% accuracy which is significantly lower than our CNN model (65.8%). This shows how effective CNN models are at image classifications when compared to the conventional DNN architectures. This is due to images being processed with convolution which optimizes the neural network inputs by extracting features good for image classification, or in our case, sentiment classification.

	CNN Model	DNN Model
Trainable Parameters	1,894,873	1,891,879
Non-Trainable Parameters	1,884	4,288
Total Parameters	1,896,757	1,896,167

```
Total params: 1,896,757
Trainable params: 1,894,873
Non-trainable params: 1,884
```

```
Total params: 1,896,167
Trainable params: 1,891,879
Non-trainable params: 4,288
```



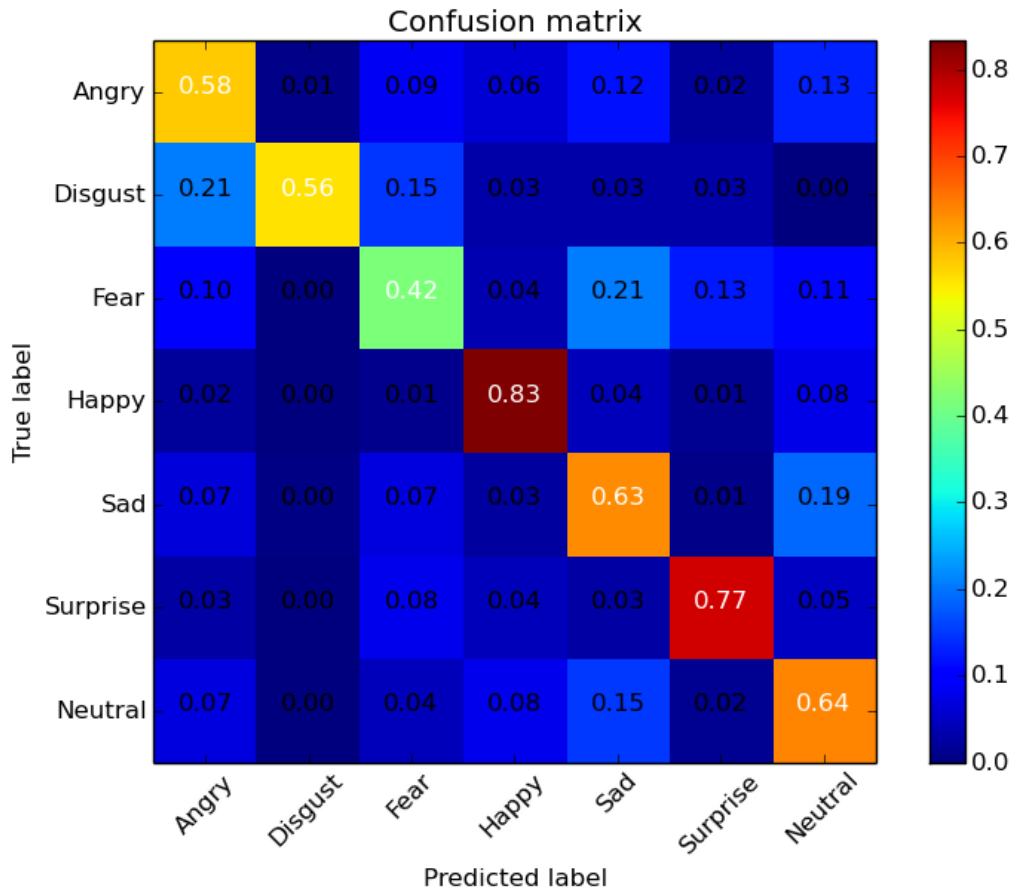
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 128)	295040
batch_normalization_1 (Batch Normalization)	(None, 128)	512
activation_1 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 256)	33024
batch_normalization_2 (Batch Normalization)	(None, 256)	1024
activation_2 (Activation)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 512)	131584
batch_normalization_3 (Batch Normalization)	(None, 512)	2048
activation_3 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 1248)	640224
batch_normalization_4 (Batch Normalization)	(None, 1248)	4992
activation_4 (Activation)	(None, 1248)	0
dropout_4 (Dropout)	(None, 1248)	0
dense_5 (Dense)	(None, 512)	639488
dropout_5 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 256)	131328
dropout_6 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 64)	16448
dense_8 (Dense)	(None, 7)	455

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 64)	640
batch_normalization_1 (Batch Normalization)	(None, 46, 46, 64)	256
activation_1 (Activation)	(None, 46, 46, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 64)	0
dropout_1 (Dropout)	(None, 23, 23, 64)	0
conv2d_2 (Conv2D)	(None, 21, 21, 128)	73856
batch_normalization_2 (Batch Normalization)	(None, 21, 21, 128)	512
activation_2 (Activation)	(None, 21, 21, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 10, 10, 128)	0
dropout_2 (Dropout)	(None, 10, 10, 128)	0
conv2d_3 (Conv2D)	(None, 8, 8, 250)	288250
batch_normalization_3 (Batch Normalization)	(None, 8, 8, 250)	1000
activation_3 (Activation)	(None, 8, 8, 250)	0
max_pooling2d_3 (MaxPooling2D)	(None, 4, 4, 250)	0
dropout_3 (Dropout)	(None, 4, 4, 250)	0
conv2d_4 (Conv2D)	(None, 2, 2, 500)	1125500
batch_normalization_4 (Batch Normalization)	(None, 2, 2, 500)	2000
activation_4 (Activation)	(None, 2, 2, 500)	0
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 500)	0
dropout_4 (Dropout)	(None, 1, 1, 500)	0
Flatten_1 (Flatten)	(None, 500)	0
dense_1 (Dense)	(None, 512)	256512
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
dropout_6 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 64)	16448
dropout_7 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 7)	455

We provide the model summary printout as references

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：We use 20% of the total data as validation set using the CNN model with 65.8% accuracy. We create the confusion matrix shown below displaying the probability of our CNN model accuracies.



From our results, we can observe that our model can predict 'happy' and 'surprise' sentiment images. Our model is unable to accurately predict negative sentiments such as 'fear', 'angry', and 'disgust' sentiments. We observe classifications with >0.1 misclassification rates as significant

'Fear' sentiments are often misclassified as 'Angry', 'Sad', 'Surprise', or 'Neutral' sentiments.

'Disgust' sentiments are often misclassified as 'Angry' or 'Fear' due to similarities in features.

'Angry' sentiments are sometimes misclassified as 'Sad' or 'Neutral'

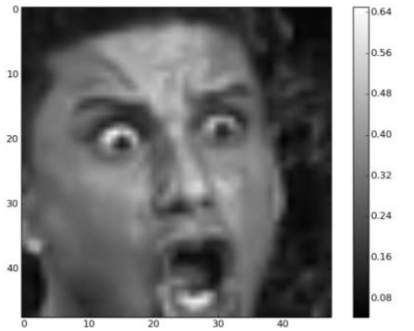
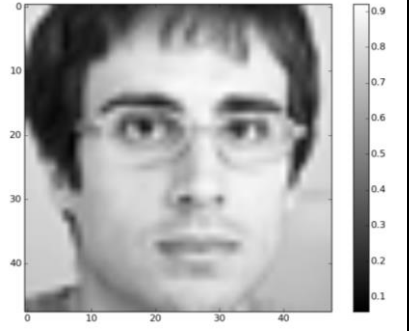
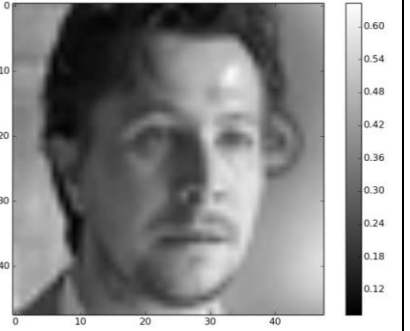
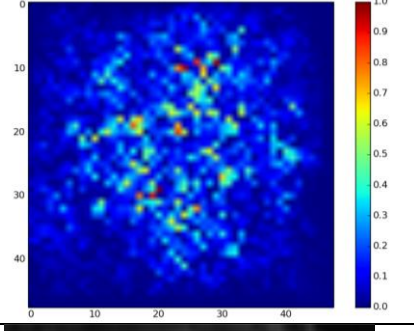
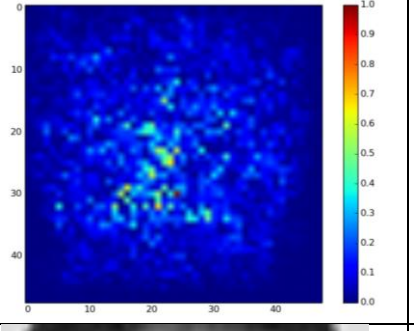
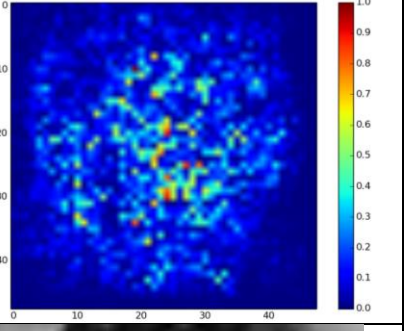
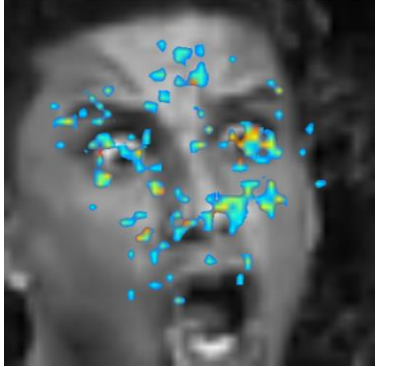
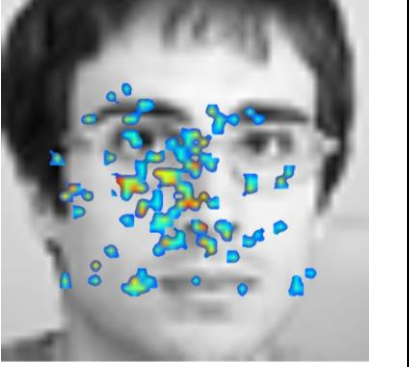
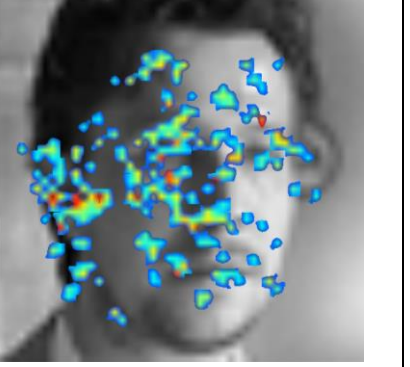
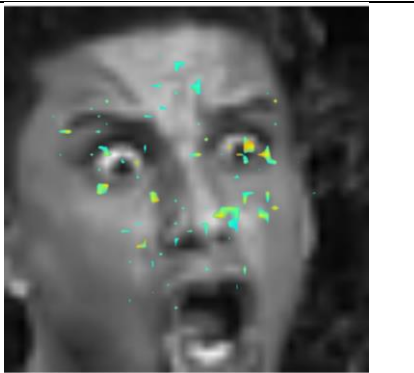

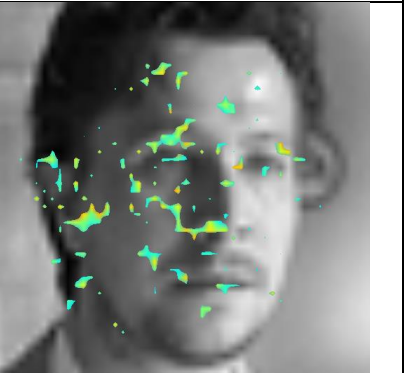
'Sad' sentiments have a chance of being misclassified as 'Neutral'

'Neutral' sentiments also have a chance of being misclassified as 'Sad'

This can be because these sentiments are hard to distinguish even when asking real people to classify emotional sentiments. Another reason for misclassification of sentiments can be due to bad data present in the training set which were not filtered out which negatively influenced our model while training. A correlation between 'Sad' and 'Neutral' sentiments indicate that the data may not be a good reflection or biased in some manner.

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：We give 3 examples of saliency/heatmaps which overlays the colormap on the original image according to a threshold. We arbitrarily choose a threshold of 0.3 and 0.7 for overlaying the color map on the

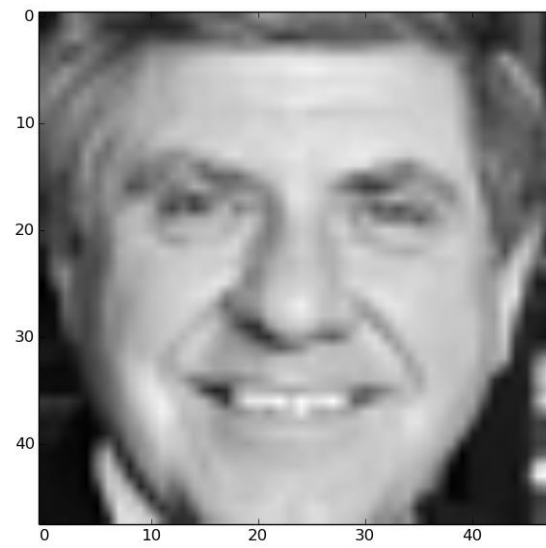
	Image 1	Image 2	Image 3
Original Image			
Color Map			
Overlaid Image Threshold at 0.3			
Overlaid Image Threshold at 0.7			

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**。

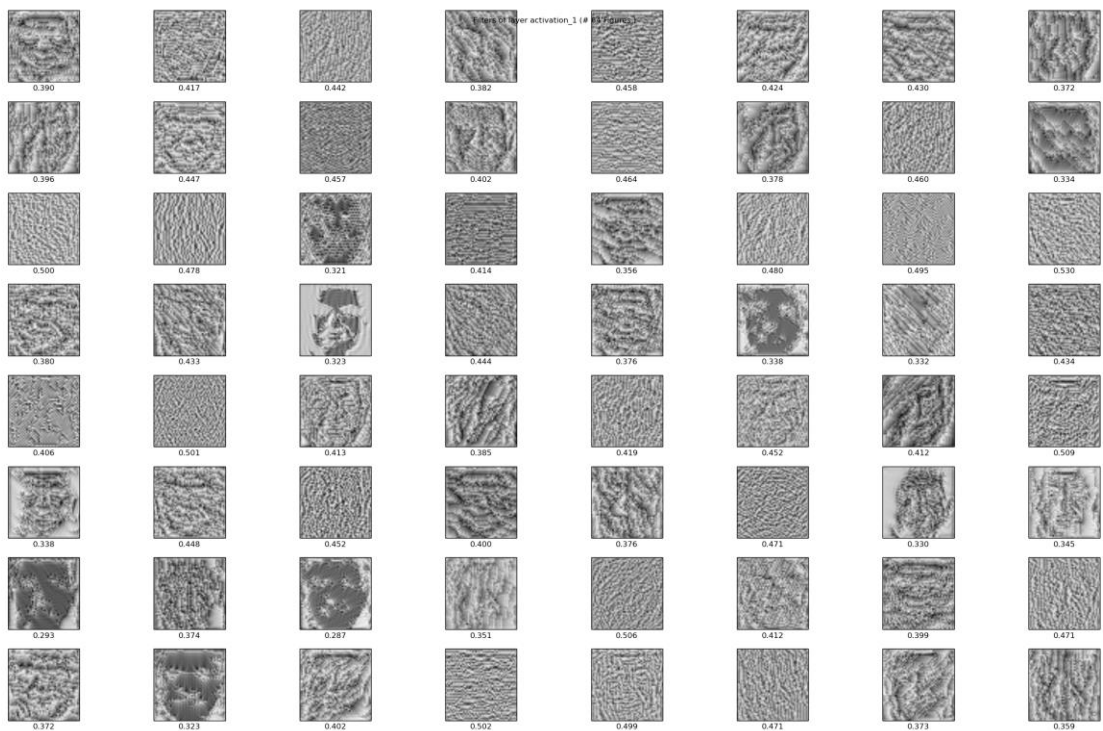
答：

By analyzing each layer, we can see that the convolution layers extract features with each filter. The features get more prominent which determines which feature will 'activate' the activation layer. We have shown two sample images. The first sample will show just the activation layer, while the second sample image will show all four activation layers shown in our CNN model.

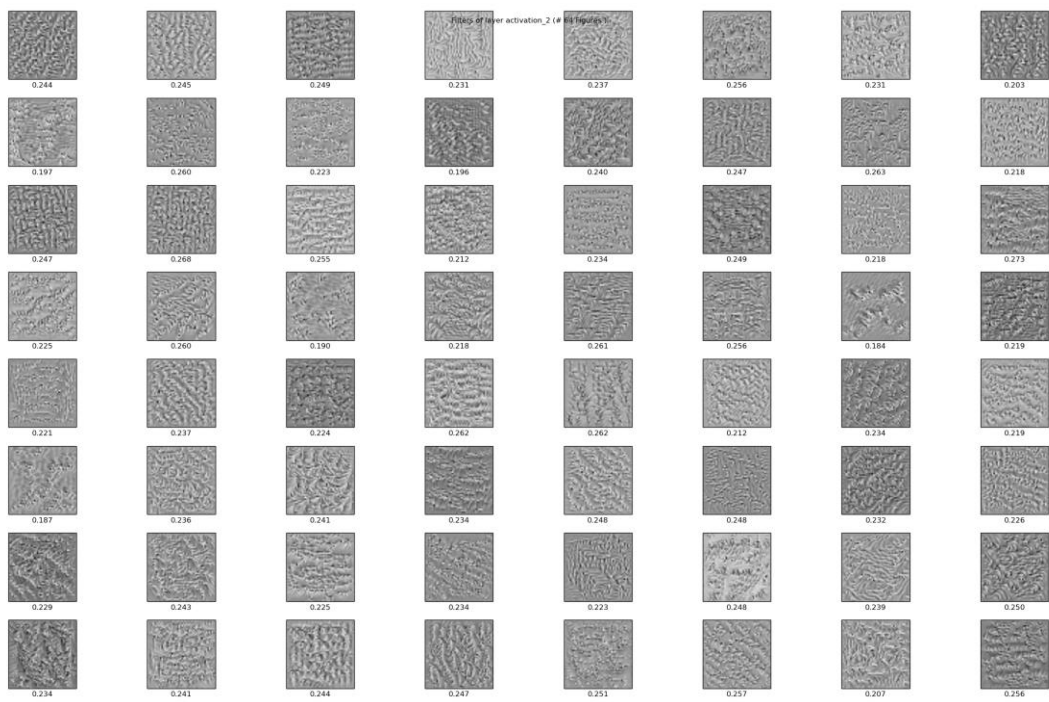




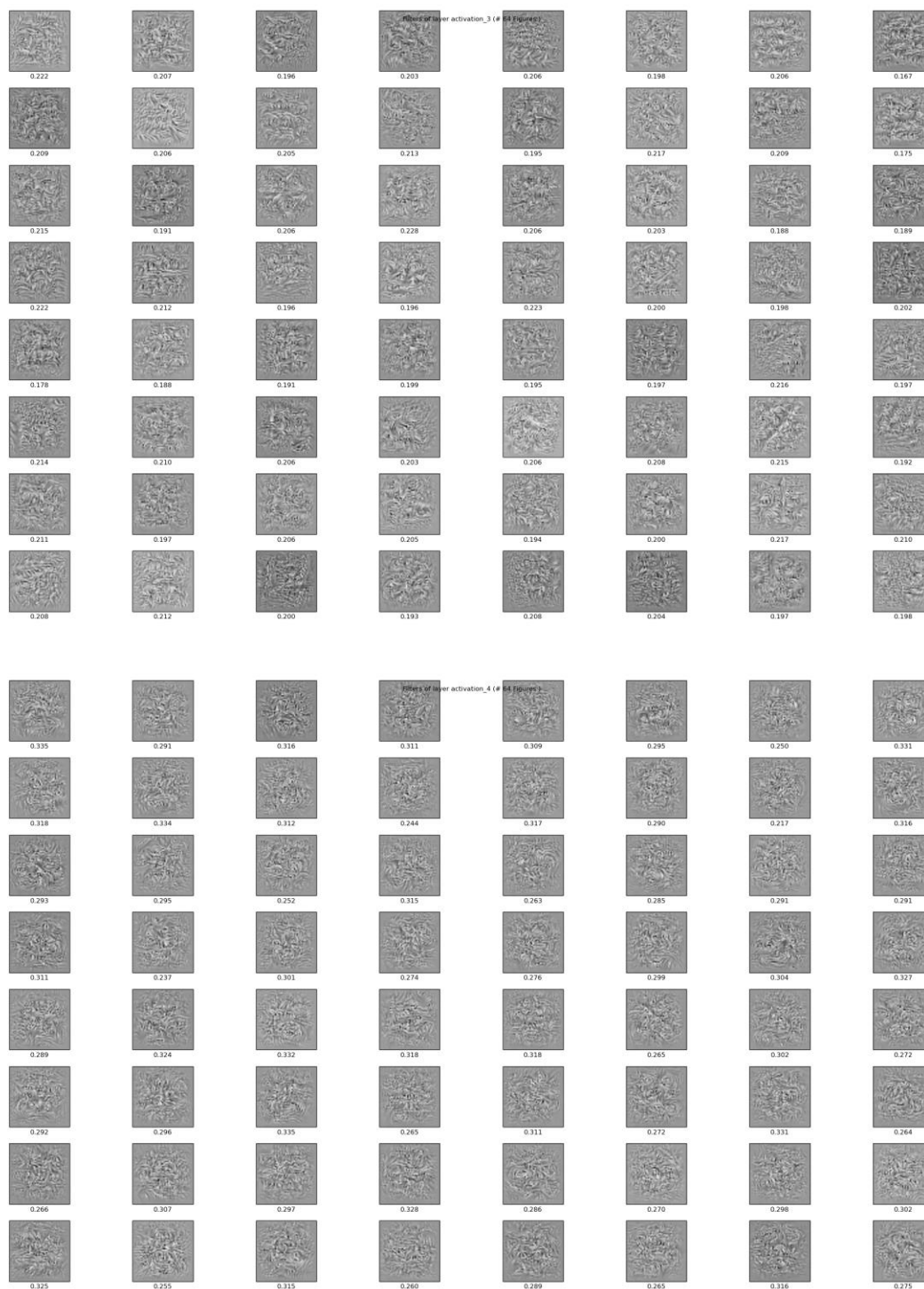
Original image



Filter_activation 1



Activation filter 3



Activation filter 4

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

[Bonus] (1%)