MSBD 6000B– Project 2

Name: Tsz Him Ng (Ryan Ng)

Student ID: 20383422

Summary

I have used program R and I have chosen the convolutional neural network model for doing this project. There are 4 layers in the model. Two of them are convolutional layers and two of them are hidden layers. 2569 rows (follow from “train.txt”) in the data will be the training set in the model. 550 rows (follow from “validation.txt”) will be the validation set and 551 rows (follow from “test.txt”) will be the testing set. The best accuracy in our model is around 71% (Screenshot stored in Appendix 1).

Pre-Processing

Before starting to run the model, I resize the image and create an array for storing the data for training, testing and validating. The size for each image is not consistent so I resize the image to 64 * 64. For the dataset that included label (i.e. training and validating dataset), the array was created with 4-dimensional and the columns are the data labels (i.e. 5 columns in the array). For example, if we are using 2569 rows to train the model, which means the array will be 2569 rows and 5 columns. The value of each cell would be the transform of image to array. For the testing set, the label are not included, so I built the array with 551 rows and 5 columns and each cell should have value 0. After that, each cell in the array will divide by 255 to get the value range between 0 and 1.

Model

I used the convolutional neural network model for testing the performance for analyzing the images. The model is convolutional neural network model and the model contains 4 layers – 2 convolutional layers and 2 hidden layers. We have separated two parts of layers for doing classification. To achieve the best accuracy in our model, the following guideline is our configuration in the model.

The first convolutional layer would have filter size 16; the second layer would have filter size 32. The dropout rate would be 0.25. Following the convolutional layers, there are 2 hidden layers. The first hidden layer will have 128 units with dropout rate 0.4 and 64 in the second with dropout rate 0.3.

I used cross entropy as cost function. The cost function as the average of computing the neg-log-probability of the correct class in the predicted value. Error function will be calculated the error as 1 minus accuracy (which choose by the maximum predicted probability as predicted class).

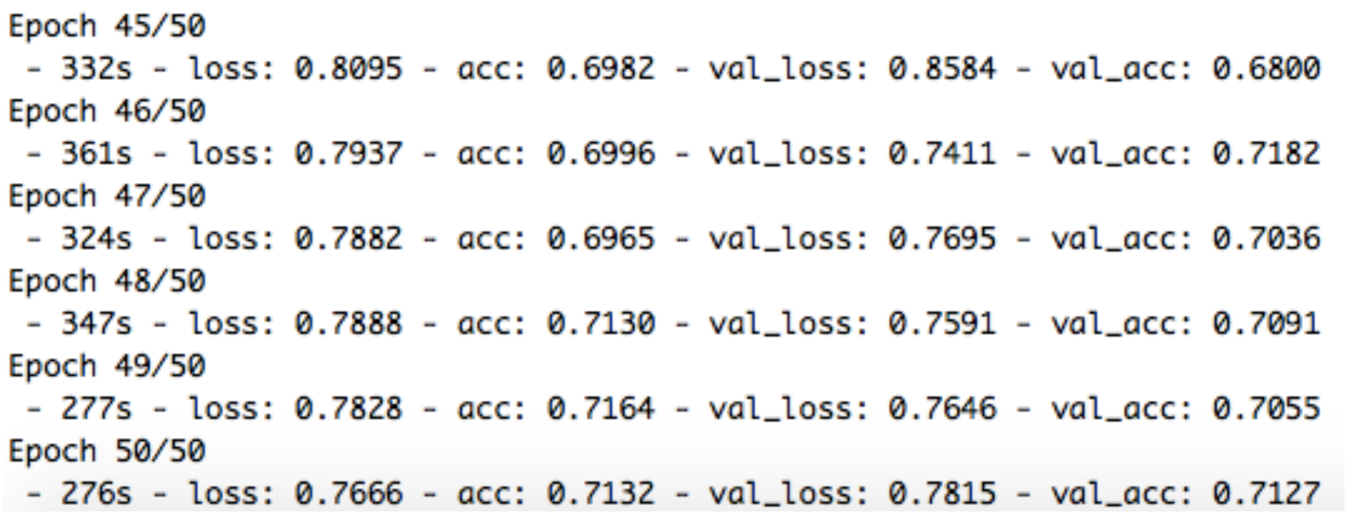
In addition, I set the batch size to 128. 50 epochs will be run in the model.

With above setting, the model would only have 57% accuracy (See the graph in Appendix 2). Thus, I would add an image model generator for enhancing the model. The generators allowed 90 degrees rotation of image, image's width shift and image's height shift. After the modification, there is significant performance improvement after adding the generator (i.e. 71% accuracy).

Result and Conclusion

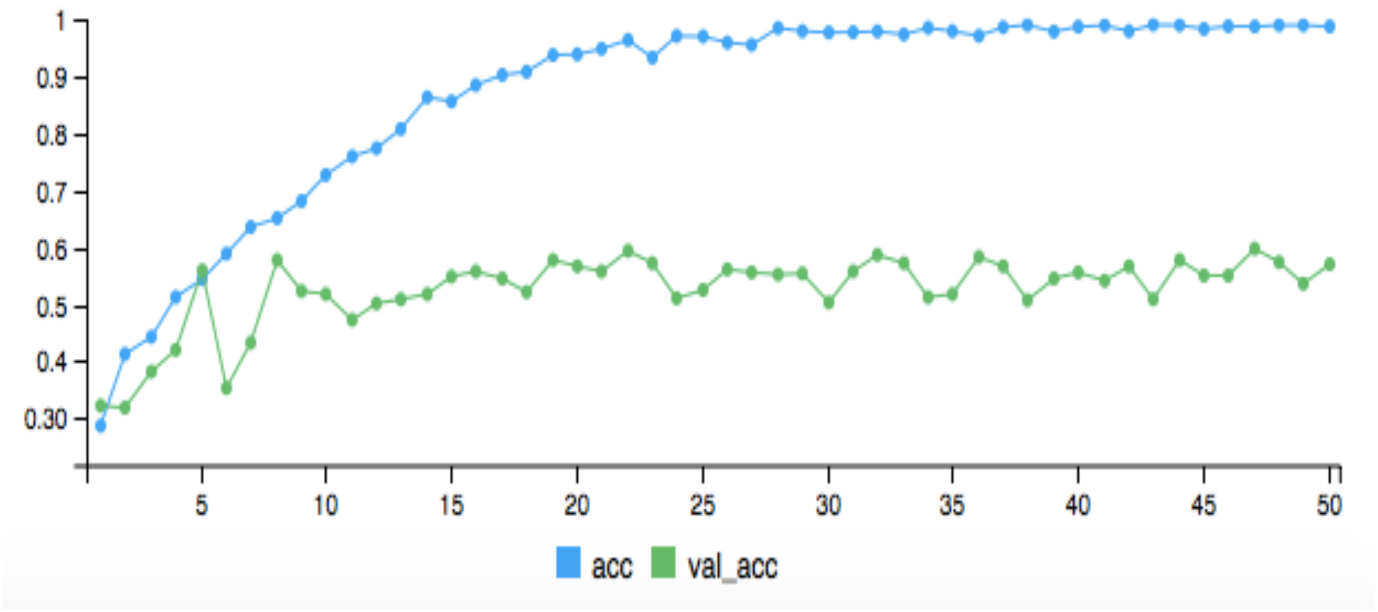
As mentioned in above, the model would only originally have 57% accuracy. However, after adding the image model generator, the accuracy for this model is around 71%, which the performance has increase around 14%.

Appendix 1: Screenshot on the accuracy for the model after adding image generator



```
Epoch 45/50
 - 332s - loss: 0.8095 - acc: 0.6982 - val_loss: 0.8584 - val_acc: 0.6800
Epoch 46/50
 - 361s - loss: 0.7937 - acc: 0.6996 - val_loss: 0.7411 - val_acc: 0.7182
Epoch 47/50
 - 324s - loss: 0.7882 - acc: 0.6965 - val_loss: 0.7695 - val_acc: 0.7036
Epoch 48/50
 - 347s - loss: 0.7888 - acc: 0.7130 - val_loss: 0.7591 - val_acc: 0.7091
Epoch 49/50
 - 277s - loss: 0.7828 - acc: 0.7164 - val_loss: 0.7646 - val_acc: 0.7055
Epoch 50/50
 - 276s - loss: 0.7666 - acc: 0.7132 - val_loss: 0.7815 - val_acc: 0.7127
```

Appendix 2: Graph versus accuracy without image generator


acc
val_acc