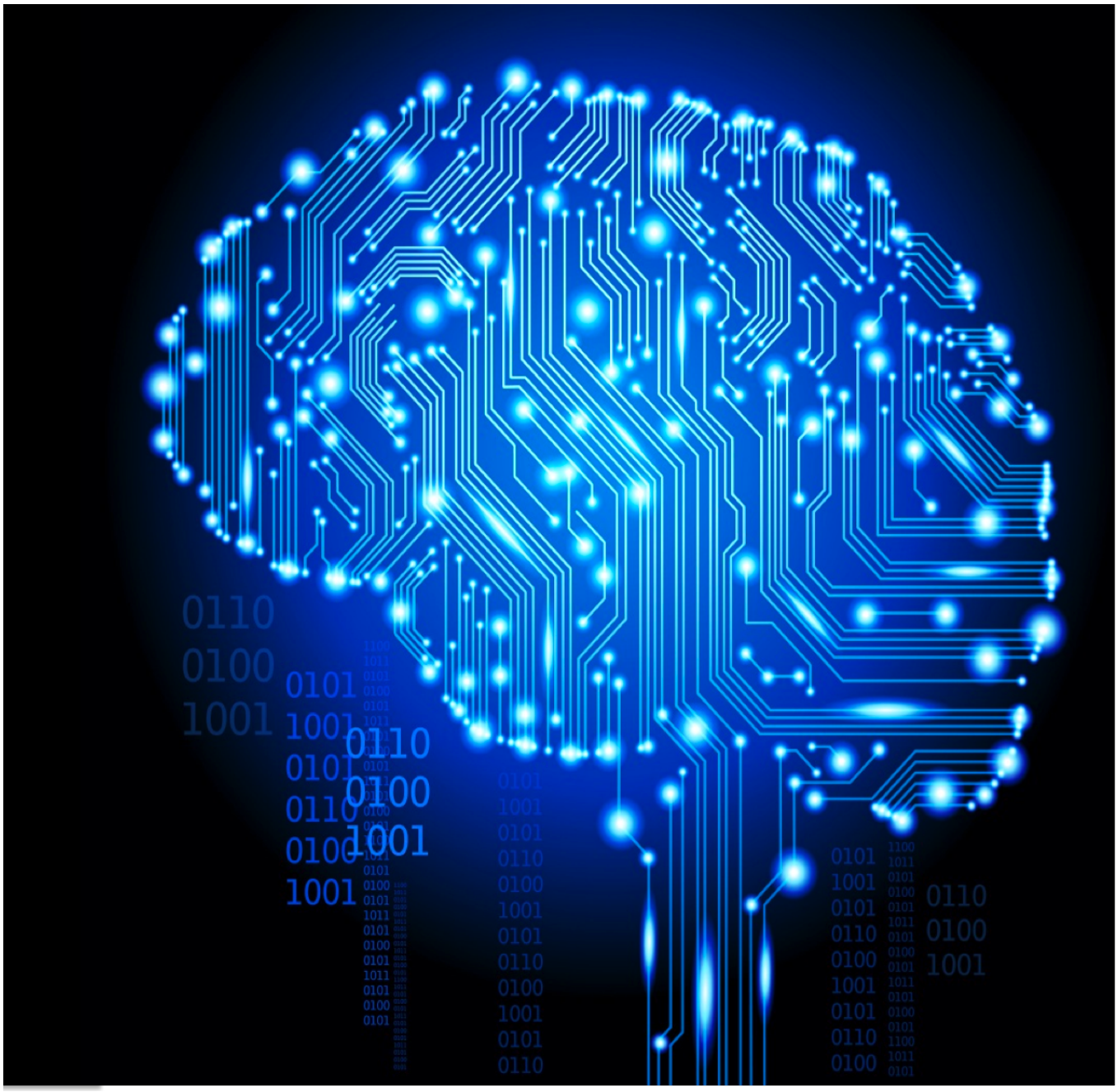

MSBD 6000B PROJECT 2

REPORT FOR CNN MODEL USING AND PRE-/POST-PROCESSING AND MATHODS

MSBD 6000B

PROJECT 2

REPORT FOR CNN MODEL USING AND PRE-/POST-PROCESSING AND MATHODS

TASK

This project is to train a deep model on the image training data set with CNN. With the trained model, predict the class labels for the test data set and put the result into a txt file.

Training data: train.txt – which is the path of the training image data and the labels are indicated after each image file path

Class label: 0 – daisy; 1 – dandelion; 2 – roses; 3 – sunflowers; 4 – tulips

Validation data: val.txt - which is the path of the validation image data and the labels are indicated after each image file path

Testing data: test.txt - which is the path of the test image data

Data Preprocessing

As I used Keras to implement the training model, to simplify the coding process with “ImageDataGenerator”, I put the Training data images into the directory /data3/preview with the subdirectory of the class labels. Also, the same for the validation image data.

Model Training

First I set the image width and height to 64, 64 to standardize the training and validation image data. Then I build the model.

This is the summary of the model

```
_________________________________________________________________\
Layer (type)                 Output Shape              Param #   \
=================================================================\
conv2d_1 (Conv2D)            (None, 62, 62, 64)        1792      \
_________________________________________________________________\
activation_1 (Activation)    (None, 62, 62, 64)        0         \
_________________________________________________________________\
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 64)        0         \
_________________________________________________________________\
conv2d_2 (Conv2D)            (None, 29, 29, 32)        18464     \
_________________________________________________________________\
activation_2 (Activation)    (None, 29, 29, 32)        0         \
_________________________________________________________________\
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0         \
_________________________________________________________________\
dropout_1 (Dropout)          (None, 14, 14, 32)        0         \
_________________________________________________________________\
conv2d_3 (Conv2D)            (None, 12, 12, 64)        18496     \
_________________________________________________________________\
activation_3 (Activation)    (None, 12, 12, 64)        0         \
_________________________________________________________________\
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0         \
_________________________________________________________________\
flatten_1 (Flatten)          (None, 2304)              0         \
_________________________________________________________________\
dense_1 (Dense)              (None, 128)               295040    \
_________________________________________________________________\
activation_4 (Activation)    (None, 128)               0         \
_________________________________________________________________\
dropout_2 (Dropout)          (None, 128)               0         \
_________________________________________________________________\
dense_2 (Dense)              (None, 5)                 645       \
_________________________________________________________________\
activation_5 (Activation)    (None, 5)                 0         \
=================================================================\
Total params: 334,437\
Trainable params: 334,437\
Non-trainable params: 0\
```

And I used epoch 150 to train my model. Then I saved the model in to “smart_pig_model_4.json” and the corresponding weigh into “sp_try.h5”.

As run it in my own PC is too slow. Nearly took 3 to 4 hours to run it once, I then put my codes and data to a AWS EC2 instance with a higher spec to run it to shorten the time needed.

I tried some different model but the best one is the above one. It’s get the accuracy 97.64%, which I think is quite impressive.

I found that the accuracy would be affected if I do the Batch Normalization between each layer. But I still haven't confirmed the reason of it.

Results

After training the model, I pull the model files back to local PC to predict the result. saved the result in the "project2_20057566.txt" file.