Homework3_report

Xia Chengyu: A0218977J Zhang Haoran: A0218995J

1. Dataset Clean

The existence of dirty data often affects the classification accuracy of the model, so the first thing we do is to clean out the dirty data in the dataset provided. In fact, in our data set, many scenes have high similarity, so it is difficult to distinguish them by simple CNN, and it is even difficult for human eyes to distinguish the corresponding pictures of these scenes. For example, the image_0271.jpg in OpenCountry can also be classified as forest.

Pic1. OpenCountry _image_0271.jpg

In order to reduce the impact of the dirty data on the model, we clean and delete the dirty data in the dataset before importing.

2. Dataset Optimization

We also apply data augmentation in following strategy:

Strategy	Method
BrightnessEnhancement	1.5 times
ContrastEnhancement	1.5 times

RandomRotation	Randomly in [-90, +90, 180]
Flip	Filp image in left and right

After data augmentation, we will generate a new data directory './augmented'

3. Model Design

Our model implements a classic 3 convolution layer CNN model to fit the train data as following:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 18496)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1183808
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 15)                975
=================================================================
Total params: 1,212,847
Trainable params: 1,212,847
Non-trainable params: 0
```

Pic2. Three-layer CNN model

In the training procedure, we choose "Adam" as our optimizer. Adam is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. It first published in 2014[1]

On the other hand, we choose “categorical_crossentropy” as our loss function. Cross-entropy loss increases as the predicted probability diverges from the actual label.[2]

4. Overfitting

The biggest problem we face in our task is overfitting, so we applied drop out layer in our model. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.[3]

5. Result Analysis

The final accuracy of our model can be seen as below. Here we randomly choose 15% of image as our validation data. With the procession of training, we obtained an accuracy of 93% on training set and 60% on the validation set. There are two reasons for the low accuracy of the prediction of validation set. The main reason is because the number of images of the training set is not enough, which leads to overfitting. Another reason is because of the similarity of different categories， such as forest and OpenCountry, bedroom and living room.

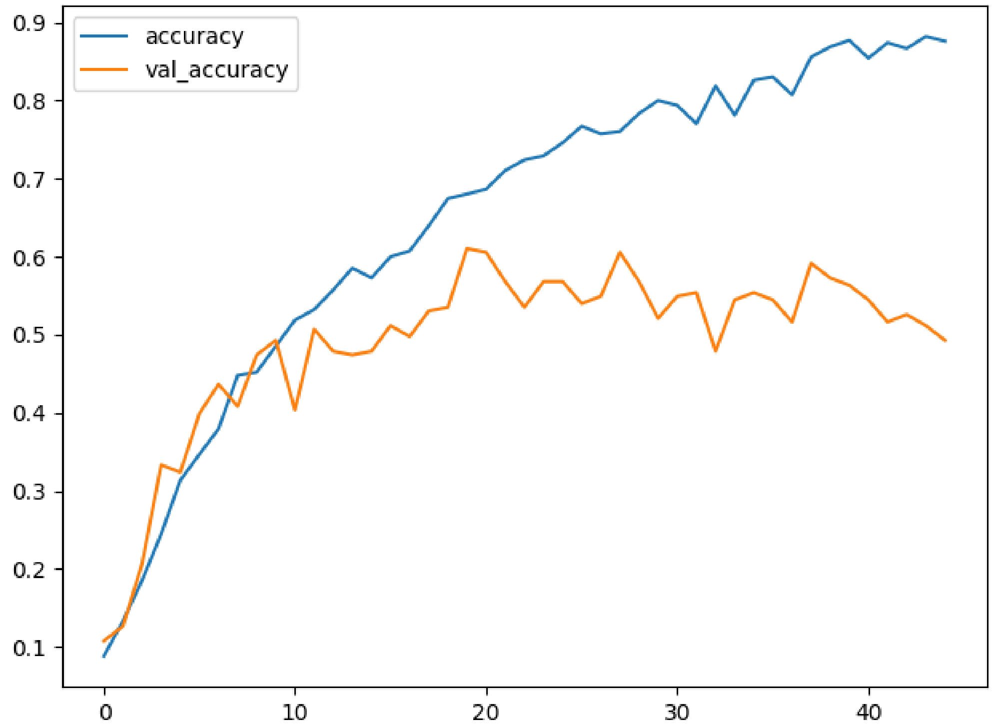


Fig3. Accuracy on training set and validation set

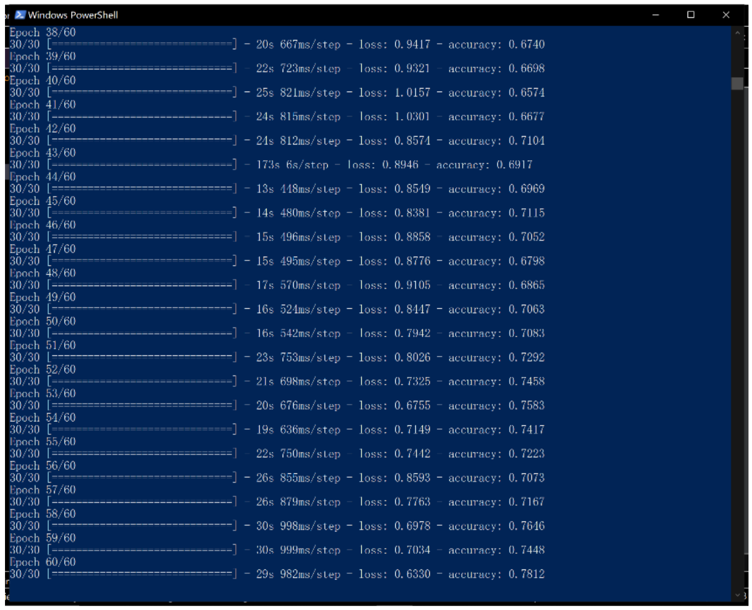
6. Implementation

The directory we submitted contains following files.

File	Function
dataAugmentation.py and dataPreprocessing.py	Data cleaning and Data Augmentation
Data_cleaning.txt	The image list we discard
My_classify_model.h5	Keras Model
Scene_recog_cnn.py	Main function

The usage of Scene_recog_cnn.py in my environment

python .\scene_recog_cnn.py --phase="test" --train_data_dir=".\train\" --test_data_dir='.\test\' --model_dir='.\'



```
Windows PowerShell
Epoch 38/60
30/30 [==============================] - 20s 667ms/step - loss: 0.9417 - accuracy: 0.6740
Epoch 39/60
30/30 [==============================] - 22s 723ms/step - loss: 0.9321 - accuracy: 0.6698
Epoch 40/60
30/30 [==============================] - 25s 821ms/step - loss: 1.0157 - accuracy: 0.6574
Epoch 41/60
30/30 [==============================] - 24s 815ms/step - loss: 1.0301 - accuracy: 0.6677
Epoch 42/60
30/30 [==============================] - 24s 812ms/step - loss: 0.8574 - accuracy: 0.7104
Epoch 43/60
30/30 [==============================] - 173s 6s/step - loss: 0.8946 - accuracy: 0.6917
Epoch 44/60
30/30 [==============================] - 13s 448ms/step - loss: 0.8549 - accuracy: 0.6969
Epoch 45/60
30/30 [==============================] - 14s 480ms/step - loss: 0.8381 - accuracy: 0.7115
Epoch 46/60
30/30 [==============================] - 15s 496ms/step - loss: 0.8858 - accuracy: 0.7052
Epoch 47/60
30/30 [==============================] - 15s 495ms/step - loss: 0.8776 - accuracy: 0.6798
Epoch 48/60
30/30 [==============================] - 17s 570ms/step - loss: 0.9105 - accuracy: 0.6865
Epoch 49/60
30/30 [==============================] - 16s 524ms/step - loss: 0.8447 - accuracy: 0.7063
Epoch 50/60
30/30 [==============================] - 16s 542ms/step - loss: 0.7942 - accuracy: 0.7083
Epoch 51/60
30/30 [==============================] - 23s 753ms/step - loss: 0.8026 - accuracy: 0.7292
Epoch 52/60
30/30 [==============================] - 21s 698ms/step - loss: 0.7325 - accuracy: 0.7458
Epoch 53/60
30/30 [==============================] - 20s 676ms/step - loss: 0.6755 - accuracy: 0.7583
Epoch 54/60
30/30 [==============================] - 19s 636ms/step - loss: 0.7149 - accuracy: 0.7417
Epoch 55/60
30/30 [==============================] - 22s 750ms/step - loss: 0.7442 - accuracy: 0.7223
Epoch 56/60
30/30 [==============================] - 26s 855ms/step - loss: 0.8593 - accuracy: 0.7073
Epoch 57/60
30/30 [==============================] - 26s 879ms/step - loss: 0.7763 - accuracy: 0.7167
Epoch 58/60
30/30 [==============================] - 30s 998ms/step - loss: 0.6978 - accuracy: 0.7646
Epoch 59/60
30/30 [==============================] - 30s 999ms/step - loss: 0.7034 - accuracy: 0.7448
Epoch 60/60
30/30 [==============================] - 29s 982ms/step - loss: 0.6330 - accuracy: 0.7812
```

7. Contribution

Chengyu optimized CNN network, changed the two-layer neural network to a three layers neural network, and improved the quality of the model.
Meanwhile, Chengyu used image processing to expand our training set and applied the dropout layer to eliminate the impact of overfitting and merged the code into a format that meets the requirements.
Haoran was responsible for dataset cleaning and successfully batched out dirty data. For the model part, Haoran built the two-layer CNN model for image training and test the dataset. In addition, Haoran was also responsible for the result analysis and writing of the report.

Reference:

[1] Adam — latest trends in deep learning optimization.(Oct 2018)
https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[2] Loss Functions
https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[3] drop out layer https://keras.io/api/layers/regularization_layers/dropout/