國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

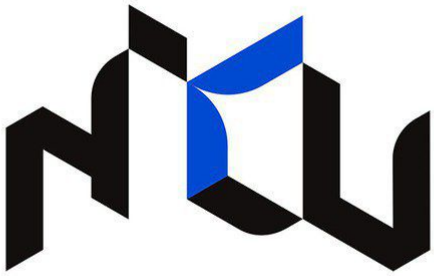

Image Processing
Programming Assignment #1

Name: 許子駿
Student ID: 311551166

Institute of Computer Science and Engineering

October 16, 2022

Contents

1 Description **2**

2 Analysis and comments **3**
2.1 Network structure 3
2.2 Analysis 4

3 Conclusion **5**

Appendices **6**

A Code **7**

Chapter 1

Description

Optical Character Reader (OCR) is a useful tool. In this project, an OCR to recognize music notes is implemented.
If a stave is input to be recognize, first, locate the five horizontal lines and capture the region. Second, extract the music notes from the captured image. Third, input the extracted music notes to the Convolutional Neural Network (CNN) model; then, the model will recognize each music note.
In this project, the most important part, the third part, is implemented.
For the selected data set, the accuracy rate can be **98%**.

Chapter 2

Analysis and comments

In this project, a simple CNN model is implemented to recognize music notes. This CNN model is developed by **Python3** and **Keras**.

2.1 Network structure

Network structure:

1. **32** convolutional layers with $4 * 4$ kernel size.
2. A **RELU** activation function.
3. **1** max pooling layer with $2 * 2$ kernel size.
4. Dropout with **25%** probability to prevent from over-fitting.
5. A **256**-dimensions fully connected (FC) layer.
6. A **RELU** activation function.
7. Dropout with **50%** probability.
8. A **9**-dimensions fully connected (FC) layer to classify.
9. A **Softmax** layer.

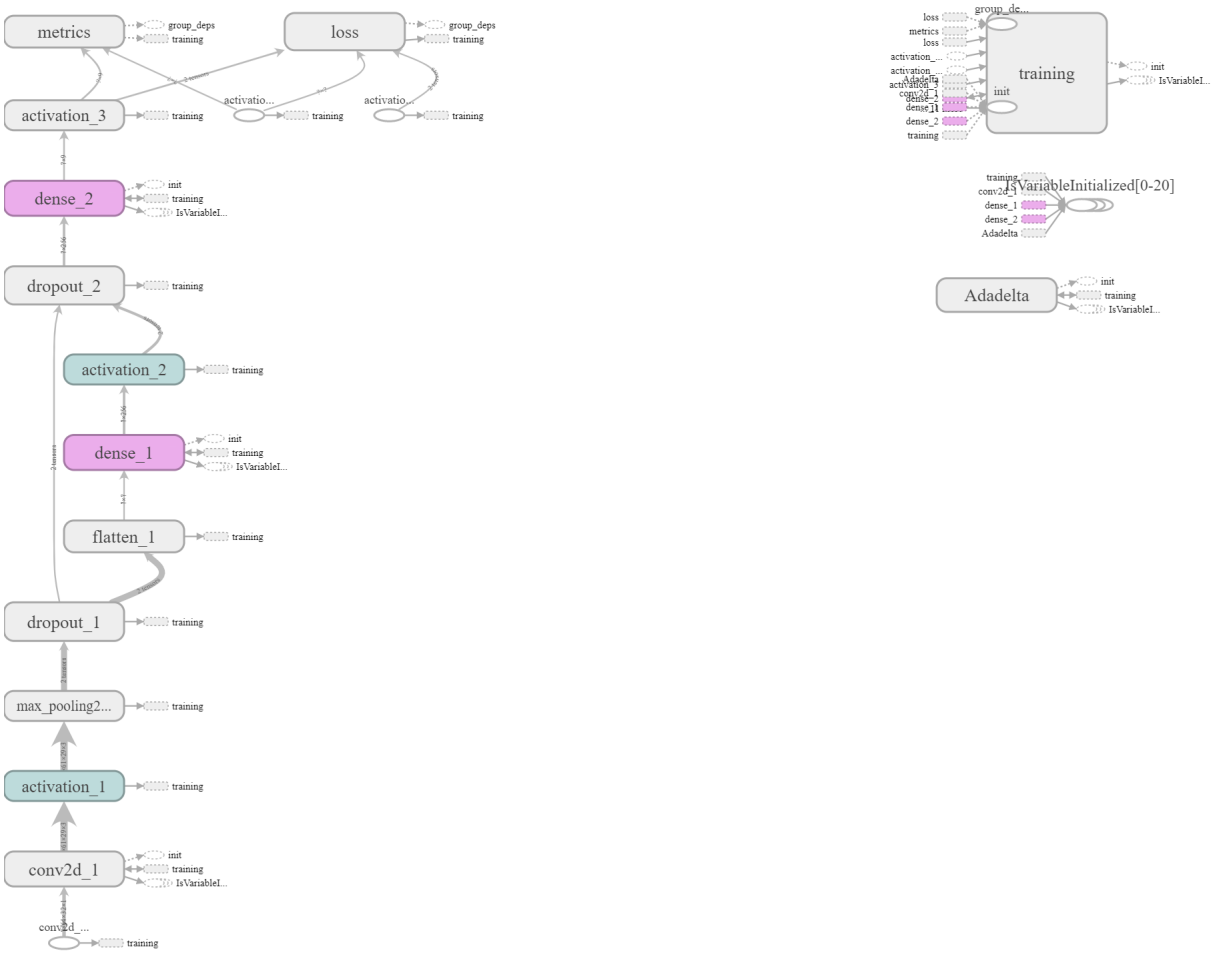

Figure 2.1: Tensorboard.

2.2 Analysis

In the figure 2.2, the accuracy rate can be **98%** in **8** epochs, and it shows the network structure works well for the testing data set.

```
Epoch 1/8
2832/2832 [==============================] - 17s 6ms/step - loss: 1.2819 - acc: 0.5876 - val_loss: 0.7760 - val_acc: 0.7400
Epoch 2/8
2832/2832 [==============================] - 16s 6ms/step - loss: 0.5354 - acc: 0.8231 - val_loss: 0.3122 - val_acc: 0.8900
Epoch 3/8
2832/2832 [==============================] - 17s 6ms/step - loss: 0.3069 - acc: 0.9025 - val_loss: 0.1630 - val_acc: 0.9500
Epoch 4/8
2832/2832 [==============================] - 19s 7ms/step - loss: 0.1921 - acc: 0.9400 - val_loss: 0.1320 - val_acc: 0.9500
Epoch 5/8
2832/2832 [==============================] - 19s 7ms/step - loss: 0.1106 - acc: 0.9668 - val_loss: 0.0776 - val_acc: 0.9700
Epoch 6/8
2832/2832 [==============================] - 17s 6ms/step - loss: 0.0755 - acc: 0.9774 - val_loss: 0.0606 - val_acc: 0.9800
Epoch 7/8
2832/2832 [==============================] - 17s 6ms/step - loss: 0.0466 - acc: 0.9898 - val_loss: 0.0314 - val_acc: 0.9900
Epoch 8/8
2832/2832 [==============================] - 16s 6ms/step - loss: 0.0356 - acc: 0.9912 - val_loss: 0.0354 - val_acc: 0.9800
Test score: 0.03541500225663185
Test accuracy: 0.98
```

Figure 2.2: Accuracy rate.

Chapter 3

Conclusion

The network structure for CNN is a problem, I have tried several times to make the network works better.
However, The network structure maybe work well only for this data set. If the network structure is not well-selected and the data set is too small, it may over-fit.
OCR is a cool implementation using deep learning and usually seen in normal life. It can be also used for another object and the result will be significant, and make lives more convenient.

Appendices

Appendix A

Code

```
# -*- coding: utf-8 -*-

import cv2
from keras import backend
from keras.callbacks import TensorBoard
from keras.layers.core import Activation, Dense, Dropout, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.models import load_model, save_model, Sequential
from keras.utils import np_utils
import numpy as np
from os import getcwd, listdir, path, remove
from os.path import join, isfile
from tempfile import mkstemp


def build_dataset():
    """
    Build a dataset, from existing dataset.
    """
    dataset = np.array([]);
    label = np.array([]);
    Len = 0
    for index in range(1, 10):
        rootdir = getcwd() + '\\dataset\\' + str(index)
        list = listdir(rootdir)
        Len += len(list)
        for i in range(len(list)):
            path = join(rootdir, list[i])
            if isfile(path):
                # Gray-scale image.
                src = cv2.imread(path, 0)

                ret, src = cv2.threshold(src, 127, 255, 0)
                kernel = np.array([[1],[1],[1]], dtype = 'uint8')
                src = cv2.dilate(src,kernel,iterations = 1)

                dataset = np.append(dataset, src)
                label = np.append(label, index - 1)

    dataset = dataset.reshape(Len, 64, 32)

    index = np.arange(Len)
```

```
    np.random.shuffle(index)

    dataset = dataset[index, :, :]
    label = label[index]

    trainDataset = dataset[:-100]
    trainLabel = label[:-100]
    testDataset = dataset[-100:]
    testLabel = label[-100:]

    return testDataset, testLabel, trainDataset, trainLabel

def cnn_model(testDataset, testLabel, trainDataset, trainLabel):
    """
    CNN model.
    """
    batch_size = 16
    nb_classes = 9
    nb_epoch = 8

    img_rows, img_cols = 64, 32
    # Number of filters.
    nb_filters = 32
    # Convolutional Kernel size.
    kernel_size = (4, 4)
    # Pooling kernel size.
    pool_size = (2, 2)

    # Load data.
    (X_train, y_train), (X_test, y_test) = (trainDataset, trainLabel), (
    testDataset, testLabel)

    if backend.image_dim_ordering() == 'th':
        # Theano: (conv_dim1, channels, conv_dim2, conv_dim3).
        X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
        X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
        input_shape = (1, img_rows, img_cols)
    else:
        # TensorFlow: (conv_dim1, conv_dim2, conv_dim3, channels).
        X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
        X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)
        input_shape = (img_rows, img_cols, 1)

    X_train = X_train.astype('float32')
    X_test = X_test.astype('float32')
    X_train /= 255
    X_test /= 255

    # Binary matrix.
    Y_train = np_utils.to_categorical(y_train, nb_classes)
    Y_test = np_utils.to_categorical(y_test, nb_classes)

    """
    32 convolutional layer, one max pooling layer, and two FC layers.
    Use dropout to prevent from overfitting.
    Use RELU as activation function.
    Use Softmax as Cost function.
    """
```

```
100     model = Sequential()
101     model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1],
        border_mode='valid', input_shape=input_shape))
102     model.add(Activation('relu'))
103     model.add(MaxPooling2D(pool_size=pool_size))
104     model.add(Dropout(0.25))
105     model.add(Flatten())
106     model.add(Dense(256))
107     model.add(Activation('relu'))
108     model.add(Dropout(0.5))
109     model.add(Dense(nb_classes))
110     model.add(Activation('softmax'))
111 
112     # Compile the model.
113     model.compile(loss='categorical_crossentropy', optimizer='adadelta',
        metrics=['accuracy'])
114 
115     # Tensorboard.
116     tbCallBack = TensorBoard(log_dir='.\\logs',
117         histogram_freq=0,
118         write_graph=True,
119         write_grads=True,
120         write_images=True,
121         embeddings_freq=0,
122         embeddings_layer_names=None,
123         embeddings_metadata=None)
124 
125     # Train the model.
126     model.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch, verbose
        =1, validation_data=(X_test, Y_test), callbacks=[tbCallBack])
127 
128     # Evaluate the model.
129     score = model.evaluate(X_test, Y_test, verbose=0)
130     print('Test score:', score[0])
131     print('Test accuracy:', score[1])
132 
133     # Save the model parameters.
134     _, fname = mkstemp('.h5', dir='.\\save')
135     save_model(model, fname)
136 
137 if __name__ == '__main__':
138     # Clear the logs.
139     for file in listdir(getcwd() + '\\logs'):
140         remove(getcwd() + '\\logs\\' + file)
141 
142     # Build dataset.
143     testDataset, testLabel, trainDataset, trainLabel = build_dataset()
144     # Train the model.
145     cnn_model(testDataset, testLabel, trainDataset, trainLabel)
```

Listing A.1: music_note_ocr.py