Term Project Report

108502571 楊佳峻

A. Accuracy: 0.9847

```
54/54 [==============================] - 1s 13ms/step - loss: 0.1019 - accuracy: 0.9847
```

B. Explaning code line by line

- My model's summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 28, 64)        256
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 128)               401536
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 440,906
Trainable params: 440,650
Non-trainable params: 256
_________________________________________________________________
```

- Explanation (next page)

```python
import matplotlib.image as img
import numpy as np
import os
from keras.models import Sequential
from keras.layers import *
from keras.utils.np_utils import to_categorical
from keras.callbacks import ModelCheckpoint


def loadData(src):
    # output shape is [-1, 28, 28, 1]
    x = np.empty([0, 28, 28, 1])
    # store label of now bmp file
    y = np.empty([0])

    # os.walk for all directories and files in the given directory
    for root, dirs, files in os.walk(src):

        # root is going to be like: "./handwrite__detect/train_image\104602513\0" and "0" is the label
        root_split = root.split("\\")

        # if root get to be like: "./handwrite__detect/train_image\104602513", that's not what we need
        if len(root_split) == 3:
            # go through all files in this root
            for f in files:
                # the file's current position
                in_file = os.path.join(root, f)
                # add the image label to the array
                y = np.append(y, [root_split[2]], axis=0)

                # 1. image only black(255, 255, 255) or white(0, 0, 0), so just choose the first number
                # 2. img.imread(in_file) give shape with RGB and transparency
                #    , so I just choose to see the R's value
                np_img = img.imread(in_file)[:, :, 0].reshape([1, 28, 28, 1])
                # add image to the array
                x = np.append(x, np_img, axis=0)

    # x: image np_array
    # y: corresponding label(np_array) to x
    return x, y


def makeModel():
    # make a sequential model and add setting to it
    model = Sequential(
        [
            # 2D conv. using 3x3 kernel map , relu activation, and output a feature map numbered 64
            Conv2D(64, kernel_size=(3, 3), padding="same", input_shape=(28, 28, 1)),
```

```
            # use batch normalization instead of dropout
            BatchNormalization(),
            # relu activation
            LeakyReLU(),
            # Pooling layer choose Max value in 2D, pool_size=2x2
            MaxPooling2D(pool_size=(2, 2)),
            # 2D conv. using 3x3 kernel map , relu activation, and output a feature map numbered 64
            Conv2D(64, (3, 3), padding="same"),
            BatchNormalization(),
            LeakyReLU(),
            MaxPooling2D(pool_size=(2, 2)),
            # conv layer to fully connected layer
            Flatten(),
            # fully connected layer, and output a feature map numbered 128
            Dense(128),
            # avoid to overfit: 0.5 is ratio of feature we remain
            Dropout(0.5),
            LeakyReLU(),
            # fully connected using softmax activation, and output 10 which is about number of categories
            Dense(10, activation="softmax"),
        ]
    )

    return model


if __name__ == "__main__":
    # training data position
    train_dataset = "./handwrite__detect/train_image"
    # testing data position
    test_dataset = "./handwrite__detect/test_image"

    # load data
    # 1. x: image list, y: number list
    # 2. shape = [-1, 28, 28, 1]
    # ------------------------- #
    # get data from my function
    x_train, y_train = loadData(train_dataset)
    # get data from my function
    x_test, y_test = loadData(test_dataset)

    # normalization the color value: 0~1
    x_train = x_train / 255
    # normalization the color value: 0~1
    x_test = x_test / 255
    # make the label to 10 categories (definition step)
    y_train = to_categorical(y_train, 10)
    # make the label to 10 categories (definition step)
    y_test = to_categorical(y_test, 10)
```

```
98
99      # make a CNN model
100     model = makeModel()
101
102     # finish the model by adding:
103     #   1. loss: loss function, which is defined in "keras.losses"
104     #   2. optimizer: define in "keras.optimizers"
105     #   3. metrics: performance evaluation functions
106     model.compile(
107         loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
108     )
109
110     # training
111     model.fit(
112         # training bmp data
113         x_train,
114         # training label
115         y_train,
116         # batch size of data each time
117         batch_size=128,
118         # train model 200 times
119         epochs=200,
120         # add testing data to make model more accuracy
121         validation_split=0.1,
122     )
123
124     # accuracy viewing
125     # ---------------- #
126     # load the saved model
127     model.save("./models/chinese_number_identification_model_BN_policy.h5")
128     # evaluate this model
129     score = model.evaluate(x_test, y_test)
130
```