FingerPrint_Classification

May 17, 2021

[63]:

```
import cv2
import os
from PIL import Image
from scipy import misc
import tensorflow as tf
import tensorflow.keras as keras
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import gzip
%matplotlib inline
from tensorflow.keras.layers import Input,Conv2D,MaxPooling2D,UpSampling2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import array_to_img
```

1 DATA Preprocessing

[64]:

```
#Data Preprocessing

def create_dataset(img_folder):

    img_data_array=[]
    class_name=[]

    for file in os.listdir(img_folder):

        image_path = os.path.join(img_folder, file)
        image = load_img(image_path, 'rb')
        image = img_to_array(image)

        if image.shape[2] == 3:
            image = image.mean(2)
        img_data_array.append(image)
        class_name.append(int(file[0]))
    return np.array(img_data_array), np.array(class_name)

def normalization(image):
    image = image / image.max()
    return image


training_path = "./01_finger_training"
test_path = './01_finger_test'

train_data, train_label = create_dataset(training_path)
test_data, _ = create_dataset(test_path)

train_data = normalization(train_data)
test_data = normalization(test_data)
```

[65]:

```
#preparing test_result_DataFrame

test_index = os.listdir(test_path)
temp = []

for index in test_index:
    index = index.split(".")
    temp.append(int(index[0]))

test_index = temp
```

[66]:

```
print('Shape of Train images :',train_data.shape)
print('Shape of Train labels : ', train_label.shape)
print('Shape of Test images : ', test_data.shape)
```

```
Shape of Train images : (80, 144, 144)
Shape of Train labels :  (80,)
Shape of Test images :  (80, 144, 144)
```

2 Model

[67]:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(144, 144)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(9, activation='softmax')
])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 20736)             0

_________________________________________________________________
dense_2 (Dense)              (None, 32)                663584

_________________________________________________________________
dense_3 (Dense)              (None, 9)                 297
=================================================================
Total params: 663,881
Trainable params: 663,881
Non-trainable params: 0

_________________________________________________________________
```

[68]:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[69]:

```
model.fit(train_data, train_label, epochs=10)
```

```
Epoch 1/10
3/3 [==============================] - 0s 9ms/step - loss: 3.1098 - accuracy:
0.1375
Epoch 2/10
3/3 [==============================] - 0s 9ms/step - loss: 1.0723 - accuracy:
0.6625
Epoch 3/10
3/3 [==============================] - 0s 7ms/step - loss: 0.6180 - accuracy:
0.8625
Epoch 4/10
3/3 [==============================] - 0s 7ms/step - loss: 0.2593 - accuracy:
0.9500
Epoch 5/10
3/3 [==============================] - 0s 9ms/step - loss: 0.1275 - accuracy:
0.9875
Epoch 6/10
3/3 [==============================] - 0s 9ms/step - loss: 0.0730 - accuracy:
0.9875
Epoch 7/10
3/3 [==============================] - 0s 10ms/step - loss: 0.0426 - accuracy:
1.0000
Epoch 8/10
3/3 [==============================] - 0s 9ms/step - loss: 0.0244 - accuracy:
1.0000
Epoch 9/10
3/3 [==============================] - 0s 10ms/step - loss: 0.0166 - accuracy:
1.0000
Epoch 10/10
3/3 [==============================] - 0s 9ms/step - loss: 0.0117 - accuracy:
```

```
1.0000
```

[69]:

```
<tensorflow.python.keras.callbacks.History at 0x1574d2668>
```

3 Training Validation

[70]:

```
predictions = model.predict(train_data)
```

[71]:

```
print(predictions.shape)
```

```
(80, 9)
```

[72]:

```
predict_label = []
for image in predictions:
    predict_label.append(np.argmax(image))
```

[73]:

```
predict_label = np.array(predict_label)
```

[74]:

```
def accuracy(original, prediction):
    accuracy = original == prediction
    accuracy = np.count_nonzero(accuracy)

    return accuracy / original.shape[0]

accuracy(train_label, predict_label)
```

[74]:

```
1.0
```

4 Test

[75]:

```
predictions = model.predict(test_data)
```

[76]:

```
print(predictions.shape)
```

```
(80, 9)
```

[77]:

```
test_label = []
for image in predictions:
    test_label.append(np.argmax(image))
```

[78]:

```
test_label = np.array(test_label)
```

[79]:

```
#
print(test_label)
```

```
[7 3 8 2 5 2 3 6 8 5 3 3 3 8 3 2 3 8 3 3 5 3 2 5 8 3 5 2 3 3 3 3 3 8 8 7 4
 3 8 3 2 3 3 5 1 3 3 4 3 4 3 2 2 5 5 2 3 3 3 8 7 8 3 3 3 7 5 3 8 3 3 6 2 1
 3 1 7 8 8 5]
```

5 Extract Pandas DataFrame

[80]:

```
test_result_df = pd.DataFrame([x for x in zip(test_index, test_label)],␣
 ↪columns=['Image', 'Answer'])
```

[81]:

```
test_result_df.head()
```

[81]:

```
   Image  Answer
0      1       7
1     10       3
2     11       8
3     12       2
4     13       5
```

[83]:

```
test_result_df = test_result_df.sort_values(by="Image")
test_result_df.head()
```

[83]:

```
    Image  Answer
0       1       7
11      2       3
22      3       2
33      4       8
44      5       1
```

[84]:

```
test_result_df.to_csv("./result.csv", index=False)
```