Perform Facial Recognition with Deep Learning in Keras Using CNN

October 17, 2022

1 1. Input the required libraries

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout
from keras.optimizers import Adam
from sklearn.metrics import confusion_matrix,classification_report
import seaborn as sns

print('Tensorflow version:',tf.__version__)
```

```
Tensorflow version: 2.9.1
```

2 2. Load the dataset after loading the dataset, you have to normalize every image.

[3]:
```
# Load the dataset
data = np.load('ORL_faces.npz')
```

[4]:
```
# Load the train dataset
xtrain = data['trainX']

# Normalize the images
xtrain = np.array(xtrain,dtype='float32') / 255
```

[5]:
```
#Load the test dataset
xtest = data['testX']

# Normalize the images
```

```
xtest = np.array(xtest,dtype='float32') / 255
```

[6]:
```
# Load the labels of dataset
ytrain = data['trainY']
ytest = data['testY']
```

[7]:
```
print('xtrain :',xtrain)
print('\n ytrain :',ytrain)
```

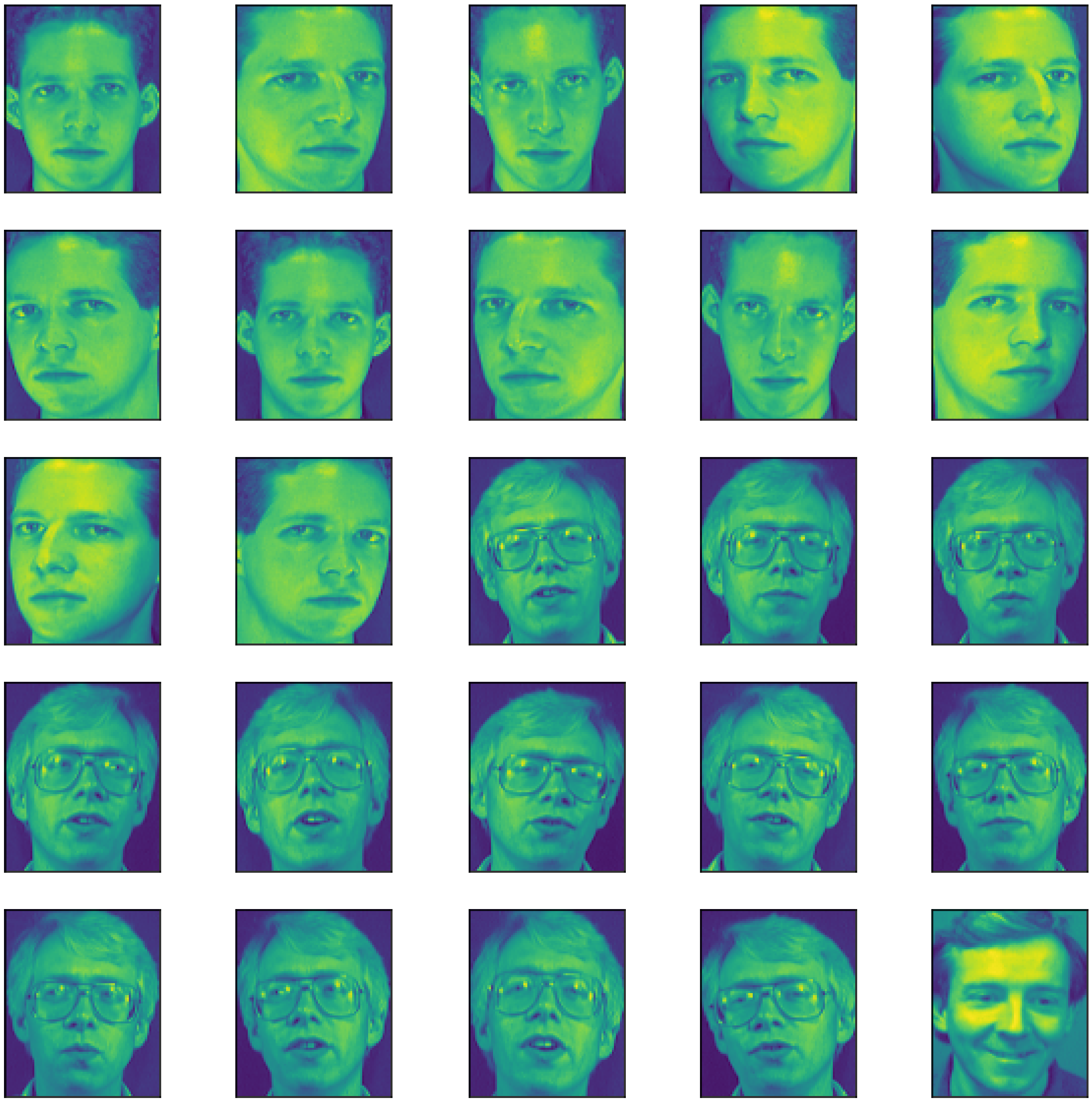
```
xtrain : [[0.1882353  0.19215687 0.1764706  … 0.18431373 0.18039216
0.18039216]
 [0.23529412 0.23529412 0.24313726 … 0.1254902  0.13333334 0.13333334]
 [0.15294118 0.17254902 0.20784314 … 0.11372549 0.10196079 0.11372549]
 …
 [0.44705883 0.45882353 0.44705883 … 0.38431373 0.3764706  0.38431373]
 [0.4117647  0.4117647  0.41960785 … 0.21176471 0.18431373 0.16078432]
 [0.45490196 0.44705883 0.45882353 … 0.37254903 0.39215687 0.39607844]]

 ytrain : [ 0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1
1
  2  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3  3  3  3  3  3
  4  4  4  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  5  5  5
  6  6  6  6  6  6  6  6  6  6  6  6  7  7  7  7  7  7  7  7  7  7  7  7
  8  8  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9  9  9  9  9
 10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11 11 11
 12 12 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13 13 13 13 13 13
 14 14 14 14 14 14 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 15 15
 16 16 16 16 16 16 16 16 16 16 16 16 17 17 17 17 17 17 17 17 17 17 17 17
 18 18 18 18 18 18 18 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 19 19]
```

[8]:
```
print('Shape of xtrain :',xtrain.shape)
print('Shape of xtest :',xtest.shape)
print('Shape of ytrain :',ytrain.shape)
print('Shape of ytest :',ytest.shape)
```

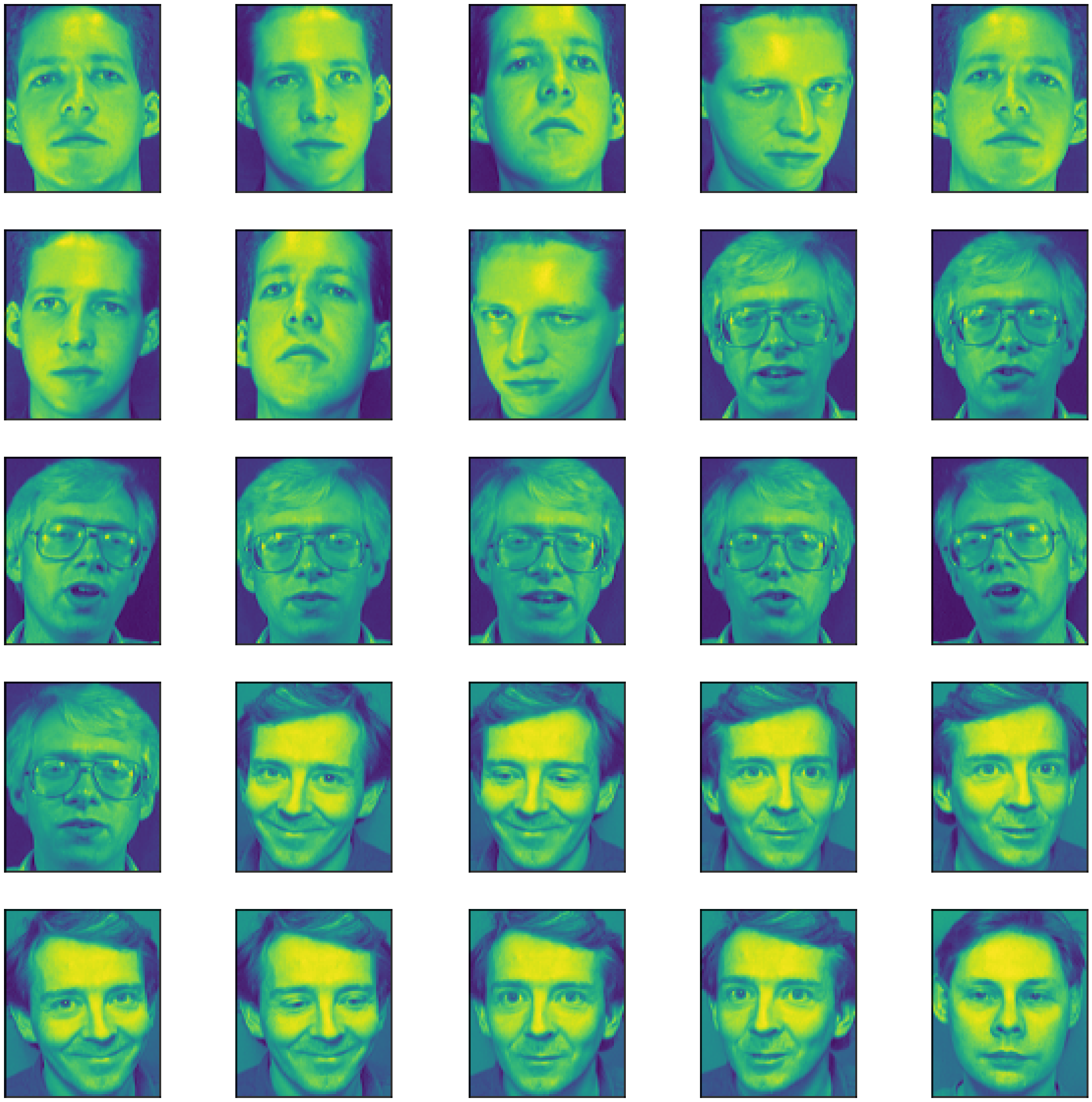
```
Shape of xtrain : (240, 10304)
Shape of xtest : (160, 10304)
Shape of ytrain : (240,)
Shape of ytest : (160,)
```

[9]:
```
# Visualize the train image
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.imshow(xtrain[i].reshape(112,92))
    plt.xticks([])
    plt.yticks([])
```

```
    plt.grid(False)
plt.show()
```



[10]:
```
# Visualize the test image
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.imshow(xtest[i].reshape(112,92))
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
plt.show()
```



3 3. Split the dataset

[11]:
```
xtrain,xvalid,ytrain,yvalid = train_test_split(xtrain,ytrain,test_size=0.
↪05,random_state=41)
```

[12]:
```
print('Shape of xtrain :',xtrain.shape)
print('Shape of xtest :',xvalid.shape)
print('Shape of ytrain :',ytrain.shape)
print('Shape of ytest :',yvalid.shape)
```

```
Shape of xtrain : (228, 10304)
Shape of xtest : (12, 10304)
```

```
Shape of ytrain : (228,)
Shape of ytest : (12,)
```

4 4. Transform the images to equal sizes to feed in CNN

[13]:
```
# Shape of image definition
rows = 112
columns = 92
image_shape = (rows,columns,1)
```

[14]:
```
xtrain = xtrain.reshape(xtrain.shape[0], *image_shape)
xtest = xtest.reshape(xtest.shape[0], *image_shape)
xvalid = xvalid.reshape(xvalid.shape[0], *image_shape)
```

[15]:
```
print('Shape of xtrain :',xtrain.shape)
print('Shape of xtest :',xtest.shape)
print('Shape of xvalid :',xvalid.shape)
```

```
Shape of xtrain : (228, 112, 92, 1)
Shape of xtest : (160, 112, 92, 1)
Shape of xvalid : (12, 112, 92, 1)
```

5 5. Build a CNN model that has 3 main layers:

i. Convolutional Layer

ii. Pooling Layer

iii. Fully Connected Layer

[16]:
```
model = Sequential()
model.
 ↪add(Conv2D(filters=32,kernel_size=7,activation='relu',input_shape=image_shape))
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(filters=64,kernel_size=5,activation='relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Flatten())
model.add(Dense(2024,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(20,activation='softmax'))
```

[17]:
```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=Adam(learning_rate=0.001),
              metrics=['accuracy'])
```

[18]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 106, 86, 32)       1600

 max_pooling2d (MaxPooling2D  (None, 53, 43, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 49, 39, 64)        51264

 max_pooling2d_1 (MaxPooling  (None, 24, 19, 64)       0
 2D)

 flatten (Flatten)           (None, 29184)             0

 dense (Dense)               (None, 2024)              59070440

 dropout (Dropout)           (None, 2024)              0

 dense_1 (Dense)             (None, 1024)              2073600

 dropout_1 (Dropout)         (None, 1024)              0

 dense_2 (Dense)             (None, 512)               524800

 dropout_2 (Dropout)         (None, 512)               0

 dense_3 (Dense)             (None, 20)                10260

=================================================================
Total params: 61,731,964
Trainable params: 61,731,964
Non-trainable params: 0
_________________________________________________________________
```

6 6. Train the model

[19]:
```
history = model.fit(np.array(xtrain),np.array(ytrain),
                    batch_size=512,
                    epochs=100,
                    verbose=1,
                    validation_data=(np.array(xvalid),np.array(yvalid)))
```

```
Epoch 1/100
1/1 [==============================] - 250s 250s/step - loss: 3.0142 - accuracy:
0.0439 - val_loss: 3.2184 - val_accuracy: 0.0000e+00
Epoch 2/100
1/1 [==============================] - 47s 47s/step - loss: 3.4576 - accuracy:
0.0263 - val_loss: 3.0100 - val_accuracy: 0.1667
Epoch 3/100
1/1 [==============================] - 67s 67s/step - loss: 3.0677 - accuracy:
0.0307 - val_loss: 3.0096 - val_accuracy: 0.0000e+00
Epoch 4/100
1/1 [==============================] - 21s 21s/step - loss: 2.9916 - accuracy:
0.0658 - val_loss: 2.9966 - val_accuracy: 0.0833
Epoch 5/100
1/1 [==============================] - 14s 14s/step - loss: 2.9897 - accuracy:
0.0439 - val_loss: 2.9936 - val_accuracy: 0.0833
Epoch 6/100
1/1 [==============================] - 22s 22s/step - loss: 2.9834 - accuracy:
0.0746 - val_loss: 2.9991 - val_accuracy: 0.0833
Epoch 7/100
1/1 [==============================] - 16s 16s/step - loss: 2.9696 - accuracy:
0.1053 - val_loss: 3.0050 - val_accuracy: 0.0000e+00
Epoch 8/100
1/1 [==============================] - 15s 15s/step - loss: 2.9593 - accuracy:
0.1096 - val_loss: 3.0066 - val_accuracy: 0.0833
Epoch 9/100
1/1 [==============================] - 43s 43s/step - loss: 2.9302 - accuracy:
0.1140 - val_loss: 3.0022 - val_accuracy: 0.0000e+00
Epoch 10/100
1/1 [==============================] - 14s 14s/step - loss: 2.9037 - accuracy:
0.1316 - val_loss: 2.9971 - val_accuracy: 0.0000e+00
Epoch 11/100
1/1 [==============================] - 13s 13s/step - loss: 2.8460 - accuracy:
0.1447 - val_loss: 2.9749 - val_accuracy: 0.0000e+00
Epoch 12/100
1/1 [==============================] - 14s 14s/step - loss: 2.7666 - accuracy:
0.1535 - val_loss: 2.9002 - val_accuracy: 0.0000e+00
Epoch 13/100
1/1 [==============================] - 14s 14s/step - loss: 2.6749 - accuracy:
0.2061 - val_loss: 2.8856 - val_accuracy: 0.0833
Epoch 14/100
```

```
1/1 [==============================] - 15s 15s/step - loss: 2.6770 - accuracy:
0.1930 - val_loss: 2.7084 - val_accuracy: 0.4167
Epoch 15/100
1/1 [==============================] - 14s 14s/step - loss: 2.5348 - accuracy:
0.2982 - val_loss: 2.5950 - val_accuracy: 0.4167
Epoch 16/100
1/1 [==============================] - 14s 14s/step - loss: 2.3803 - accuracy:
0.4123 - val_loss: 2.3941 - val_accuracy: 0.2500
Epoch 17/100
1/1 [==============================] - 14s 14s/step - loss: 2.2255 - accuracy:
0.3465 - val_loss: 2.2334 - val_accuracy: 0.3333
Epoch 18/100
1/1 [==============================] - 14s 14s/step - loss: 1.9986 - accuracy:
0.3728 - val_loss: 2.1388 - val_accuracy: 0.4167
Epoch 19/100
1/1 [==============================] - 14s 14s/step - loss: 1.7954 - accuracy:
0.4737 - val_loss: 1.9395 - val_accuracy: 0.4167
Epoch 20/100
1/1 [==============================] - 15s 15s/step - loss: 1.6493 - accuracy:
0.5132 - val_loss: 1.6654 - val_accuracy: 0.4167
Epoch 21/100
1/1 [==============================] - 17s 17s/step - loss: 1.5132 - accuracy:
0.5351 - val_loss: 1.5417 - val_accuracy: 0.5000
Epoch 22/100
1/1 [==============================] - 15s 15s/step - loss: 1.3199 - accuracy:
0.6404 - val_loss: 1.3591 - val_accuracy: 0.5833
Epoch 23/100
1/1 [==============================] - 21s 21s/step - loss: 1.1984 - accuracy:
0.6009 - val_loss: 1.0536 - val_accuracy: 0.8333
Epoch 24/100
1/1 [==============================] - 99s 99s/step - loss: 0.9657 - accuracy:
0.7018 - val_loss: 0.7833 - val_accuracy: 0.8333
Epoch 25/100
1/1 [==============================] - 17s 17s/step - loss: 0.9013 - accuracy:
0.7281 - val_loss: 0.6451 - val_accuracy: 1.0000
Epoch 26/100
1/1 [==============================] - 14s 14s/step - loss: 0.8276 - accuracy:
0.7149 - val_loss: 0.6413 - val_accuracy: 1.0000
Epoch 27/100
1/1 [==============================] - 17s 17s/step - loss: 0.6983 - accuracy:
0.8202 - val_loss: 0.5249 - val_accuracy: 1.0000
Epoch 28/100
1/1 [==============================] - 14s 14s/step - loss: 0.5310 - accuracy:
0.8640 - val_loss: 0.3803 - val_accuracy: 1.0000
Epoch 29/100
1/1 [==============================] - 14s 14s/step - loss: 0.5579 - accuracy:
0.8202 - val_loss: 0.3172 - val_accuracy: 1.0000
Epoch 30/100
```

```
1/1 [==============================] - 14s 14s/step - loss: 0.5010 - accuracy:
0.8465 - val_loss: 0.2431 - val_accuracy: 1.0000
Epoch 31/100
1/1 [==============================] - 14s 14s/step - loss: 0.3160 - accuracy:
0.9167 - val_loss: 0.2308 - val_accuracy: 1.0000
Epoch 32/100
1/1 [==============================] - 14s 14s/step - loss: 0.2647 - accuracy:
0.9211 - val_loss: 0.1811 - val_accuracy: 1.0000
Epoch 33/100
1/1 [==============================] - 14s 14s/step - loss: 0.2714 - accuracy:
0.9167 - val_loss: 0.0888 - val_accuracy: 1.0000
Epoch 34/100
1/1 [==============================] - 14s 14s/step - loss: 0.1644 - accuracy:
0.9781 - val_loss: 0.0611 - val_accuracy: 1.0000
Epoch 35/100
1/1 [==============================] - 14s 14s/step - loss: 0.1141 - accuracy:
0.9737 - val_loss: 0.0473 - val_accuracy: 1.0000
Epoch 36/100
1/1 [==============================] - 14s 14s/step - loss: 0.1055 - accuracy:
0.9737 - val_loss: 0.0483 - val_accuracy: 1.0000
Epoch 37/100
1/1 [==============================] - 14s 14s/step - loss: 0.0676 - accuracy:
0.9868 - val_loss: 0.0676 - val_accuracy: 1.0000
Epoch 38/100
1/1 [==============================] - 19s 19s/step - loss: 0.1158 - accuracy:
0.9693 - val_loss: 0.0466 - val_accuracy: 1.0000
Epoch 39/100
1/1 [==============================] - 16s 16s/step - loss: 0.0743 - accuracy:
0.9693 - val_loss: 0.0196 - val_accuracy: 1.0000
Epoch 40/100
1/1 [==============================] - 15s 15s/step - loss: 0.0658 - accuracy:
0.9737 - val_loss: 0.0135 - val_accuracy: 1.0000
Epoch 41/100
1/1 [==============================] - 15s 15s/step - loss: 0.0643 - accuracy:
0.9868 - val_loss: 0.0121 - val_accuracy: 1.0000
Epoch 42/100
1/1 [==============================] - 14s 14s/step - loss: 0.0727 - accuracy:
0.9781 - val_loss: 0.0074 - val_accuracy: 1.0000
Epoch 43/100
1/1 [==============================] - 15s 15s/step - loss: 0.0374 - accuracy:
0.9868 - val_loss: 0.0064 - val_accuracy: 1.0000
Epoch 44/100
1/1 [==============================] - 14s 14s/step - loss: 0.0142 - accuracy:
1.0000 - val_loss: 0.0096 - val_accuracy: 1.0000
Epoch 45/100
1/1 [==============================] - 14s 14s/step - loss: 0.0392 - accuracy:
0.9912 - val_loss: 0.0243 - val_accuracy: 1.0000
Epoch 46/100
```

```
1/1 [==============================] - 14s 14s/step - loss: 0.0363 - accuracy:
0.9912 - val_loss: 0.0264 - val_accuracy: 1.0000
Epoch 47/100
1/1 [==============================] - 15s 15s/step - loss: 0.0521 - accuracy:
0.9825 - val_loss: 0.0077 - val_accuracy: 1.0000
Epoch 48/100
1/1 [==============================] - 15s 15s/step - loss: 0.0230 - accuracy:
0.9912 - val_loss: 0.0059 - val_accuracy: 1.0000
Epoch 49/100
1/1 [==============================] - 14s 14s/step - loss: 0.0426 - accuracy:
0.9912 - val_loss: 0.0065 - val_accuracy: 1.0000
Epoch 50/100
1/1 [==============================] - 14s 14s/step - loss: 0.0281 - accuracy:
0.9956 - val_loss: 0.0070 - val_accuracy: 1.0000
Epoch 51/100
1/1 [==============================] - 14s 14s/step - loss: 0.0215 - accuracy:
0.9956 - val_loss: 0.0065 - val_accuracy: 1.0000
Epoch 52/100
1/1 [==============================] - 13s 13s/step - loss: 0.0230 - accuracy:
0.9956 - val_loss: 0.0051 - val_accuracy: 1.0000
Epoch 53/100
1/1 [==============================] - 13s 13s/step - loss: 0.0204 - accuracy:
0.9956 - val_loss: 0.0034 - val_accuracy: 1.0000
Epoch 54/100
1/1 [==============================] - 13s 13s/step - loss: 0.0152 - accuracy:
1.0000 - val_loss: 0.0026 - val_accuracy: 1.0000
Epoch 55/100
1/1 [==============================] - 13s 13s/step - loss: 0.0465 - accuracy:
0.9912 - val_loss: 0.0022 - val_accuracy: 1.0000
Epoch 56/100
1/1 [==============================] - 12s 12s/step - loss: 0.0116 - accuracy:
1.0000 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 57/100
1/1 [==============================] - 12s 12s/step - loss: 0.0237 - accuracy:
0.9956 - val_loss: 0.0020 - val_accuracy: 1.0000
Epoch 58/100
1/1 [==============================] - 13s 13s/step - loss: 0.0046 - accuracy:
1.0000 - val_loss: 0.0020 - val_accuracy: 1.0000
Epoch 59/100
1/1 [==============================] - 13s 13s/step - loss: 0.0117 - accuracy:
0.9912 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 60/100
1/1 [==============================] - 13s 13s/step - loss: 0.0076 - accuracy:
1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 61/100
1/1 [==============================] - 12s 12s/step - loss: 0.0145 - accuracy:
0.9956 - val_loss: 6.9563e-04 - val_accuracy: 1.0000
Epoch 62/100
```

```
1/1 [==============================] - 12s 12s/step - loss: 0.0073 - accuracy:
0.9956 - val_loss: 4.6428e-04 - val_accuracy: 1.0000
Epoch 63/100
1/1 [==============================] - 13s 13s/step - loss: 0.0061 - accuracy:
1.0000 - val_loss: 3.1933e-04 - val_accuracy: 1.0000
Epoch 64/100
1/1 [==============================] - 13s 13s/step - loss: 0.0020 - accuracy:
1.0000 - val_loss: 2.3133e-04 - val_accuracy: 1.0000
Epoch 65/100
1/1 [==============================] - 13s 13s/step - loss: 0.0082 - accuracy:
0.9956 - val_loss: 1.9001e-04 - val_accuracy: 1.0000
Epoch 66/100
1/1 [==============================] - 13s 13s/step - loss: 0.0029 - accuracy:
1.0000 - val_loss: 1.6877e-04 - val_accuracy: 1.0000
Epoch 67/100
1/1 [==============================] - 13s 13s/step - loss: 0.0266 - accuracy:
0.9956 - val_loss: 1.9083e-04 - val_accuracy: 1.0000
Epoch 68/100
1/1 [==============================] - 13s 13s/step - loss: 0.0028 - accuracy:
1.0000 - val_loss: 2.3311e-04 - val_accuracy: 1.0000
Epoch 69/100
1/1 [==============================] - 13s 13s/step - loss: 0.0044 - accuracy:
1.0000 - val_loss: 3.1910e-04 - val_accuracy: 1.0000
Epoch 70/100
1/1 [==============================] - 12s 12s/step - loss: 0.0081 - accuracy:
0.9956 - val_loss: 4.1139e-04 - val_accuracy: 1.0000
Epoch 71/100
1/1 [==============================] - 12s 12s/step - loss: 0.0046 - accuracy:
1.0000 - val_loss: 4.8869e-04 - val_accuracy: 1.0000
Epoch 72/100
1/1 [==============================] - 13s 13s/step - loss: 0.0048 - accuracy:
1.0000 - val_loss: 5.5797e-04 - val_accuracy: 1.0000
Epoch 73/100
1/1 [==============================] - 12s 12s/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 6.3142e-04 - val_accuracy: 1.0000
Epoch 74/100
1/1 [==============================] - 12s 12s/step - loss: 0.0022 - accuracy:
1.0000 - val_loss: 6.7084e-04 - val_accuracy: 1.0000
Epoch 75/100
1/1 [==============================] - 13s 13s/step - loss: 0.0038 - accuracy:
1.0000 - val_loss: 6.7499e-04 - val_accuracy: 1.0000
Epoch 76/100
1/1 [==============================] - 13s 13s/step - loss: 0.0015 - accuracy:
1.0000 - val_loss: 6.3653e-04 - val_accuracy: 1.0000
Epoch 77/100
1/1 [==============================] - 13s 13s/step - loss: 0.0018 - accuracy:
1.0000 - val_loss: 5.2073e-04 - val_accuracy: 1.0000
Epoch 78/100
```

```
1/1 [==============================] - 14s 14s/step - loss: 0.0017 - accuracy:
1.0000 - val_loss: 3.6967e-04 - val_accuracy: 1.0000
Epoch 79/100
1/1 [==============================] - 13s 13s/step - loss: 0.0112 - accuracy:
0.9956 - val_loss: 3.6260e-04 - val_accuracy: 1.0000
Epoch 80/100
1/1 [==============================] - 12s 12s/step - loss: 0.0119 - accuracy:
0.9956 - val_loss: 3.9327e-04 - val_accuracy: 1.0000
Epoch 81/100
1/1 [==============================] - 12s 12s/step - loss: 0.0013 - accuracy:
1.0000 - val_loss: 4.3586e-04 - val_accuracy: 1.0000
Epoch 82/100
1/1 [==============================] - 13s 13s/step - loss: 0.0045 - accuracy:
1.0000 - val_loss: 4.6614e-04 - val_accuracy: 1.0000
Epoch 83/100
1/1 [==============================] - 13s 13s/step - loss: 0.0023 - accuracy:
1.0000 - val_loss: 4.5756e-04 - val_accuracy: 1.0000
Epoch 84/100
1/1 [==============================] - 14s 14s/step - loss: 0.0021 - accuracy:
1.0000 - val_loss: 4.1266e-04 - val_accuracy: 1.0000
Epoch 85/100
1/1 [==============================] - 42s 42s/step - loss: 0.0016 - accuracy:
1.0000 - val_loss: 3.5652e-04 - val_accuracy: 1.0000
Epoch 86/100
1/1 [==============================] - 13s 13s/step - loss: 0.0041 - accuracy:
1.0000 - val_loss: 3.0152e-04 - val_accuracy: 1.0000
Epoch 87/100
1/1 [==============================] - 12s 12s/step - loss: 0.0012 - accuracy:
1.0000 - val_loss: 2.5359e-04 - val_accuracy: 1.0000
Epoch 88/100
1/1 [==============================] - 13s 13s/step - loss: 0.0109 - accuracy:
0.9956 - val_loss: 2.2350e-04 - val_accuracy: 1.0000
Epoch 89/100
1/1 [==============================] - 14s 14s/step - loss: 3.5423e-04 -
accuracy: 1.0000 - val_loss: 2.0042e-04 - val_accuracy: 1.0000
Epoch 90/100
1/1 [==============================] - 12s 12s/step - loss: 6.5771e-04 -
accuracy: 1.0000 - val_loss: 1.8055e-04 - val_accuracy: 1.0000
Epoch 91/100
1/1 [==============================] - 12s 12s/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 1.6035e-04 - val_accuracy: 1.0000
Epoch 92/100
1/1 [==============================] - 13s 13s/step - loss: 4.1956e-04 -
accuracy: 1.0000 - val_loss: 1.4273e-04 - val_accuracy: 1.0000
Epoch 93/100
1/1 [==============================] - 12s 12s/step - loss: 6.4156e-04 -
accuracy: 1.0000 - val_loss: 1.2553e-04 - val_accuracy: 1.0000
Epoch 94/100
```

```
1/1 [==============================] - 12s 12s/step - loss: 0.0014 - accuracy:
1.0000 - val_loss: 1.1043e-04 - val_accuracy: 1.0000
Epoch 95/100
1/1 [==============================] - 12s 12s/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 9.4731e-05 - val_accuracy: 1.0000
Epoch 96/100
1/1 [==============================] - 12s 12s/step - loss: 0.0026 - accuracy:
1.0000 - val_loss: 8.2398e-05 - val_accuracy: 1.0000
Epoch 97/100
1/1 [==============================] - 12s 12s/step - loss: 0.0026 - accuracy:
1.0000 - val_loss: 6.9011e-05 - val_accuracy: 1.0000
Epoch 98/100
1/1 [==============================] - 12s 12s/step - loss: 0.0080 - accuracy:
0.9956 - val_loss: 7.2327e-05 - val_accuracy: 1.0000
Epoch 99/100
1/1 [==============================] - 12s 12s/step - loss: 0.0018 - accuracy:
1.0000 - val_loss: 7.5236e-05 - val_accuracy: 1.0000
Epoch 100/100
1/1 [==============================] - 12s 12s/step - loss: 0.0091 - accuracy:
0.9956 - val_loss: 7.3824e-05 - val_accuracy: 1.0000
```

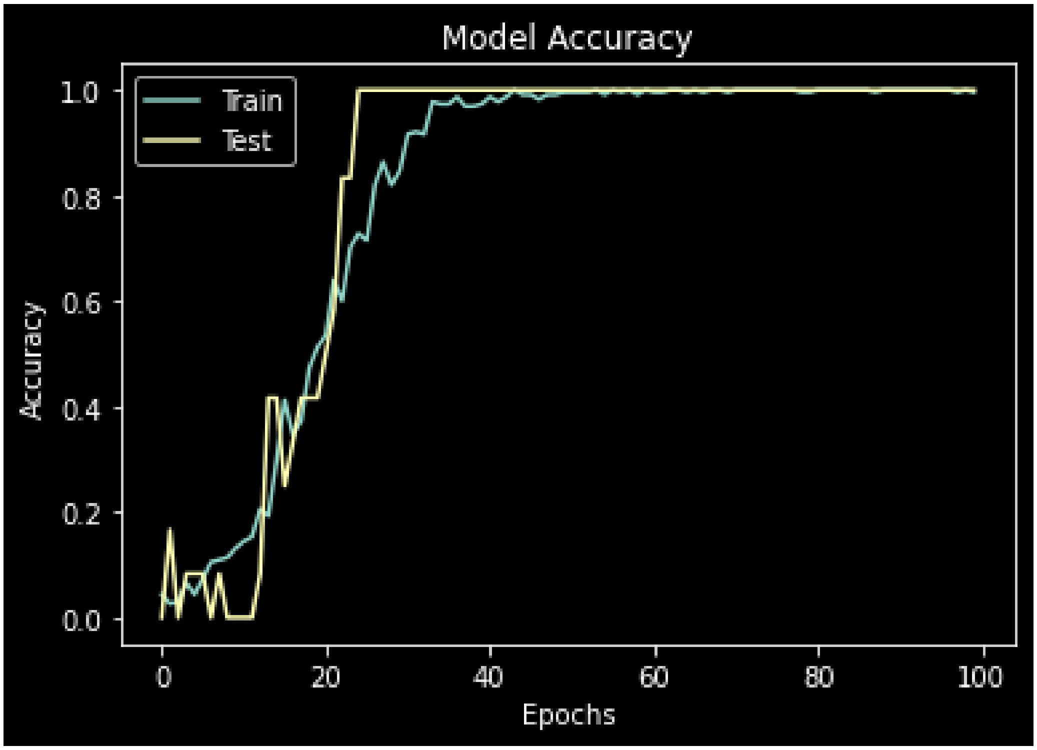
[20]:
```python
score = model.evaluate(np.array(xtest),np.array(ytest),verbose=1)

print('Test Loss {:.4f}'.format(score[0]))
print('Test Accuracy {:.4f}'.format(score[1]))
```

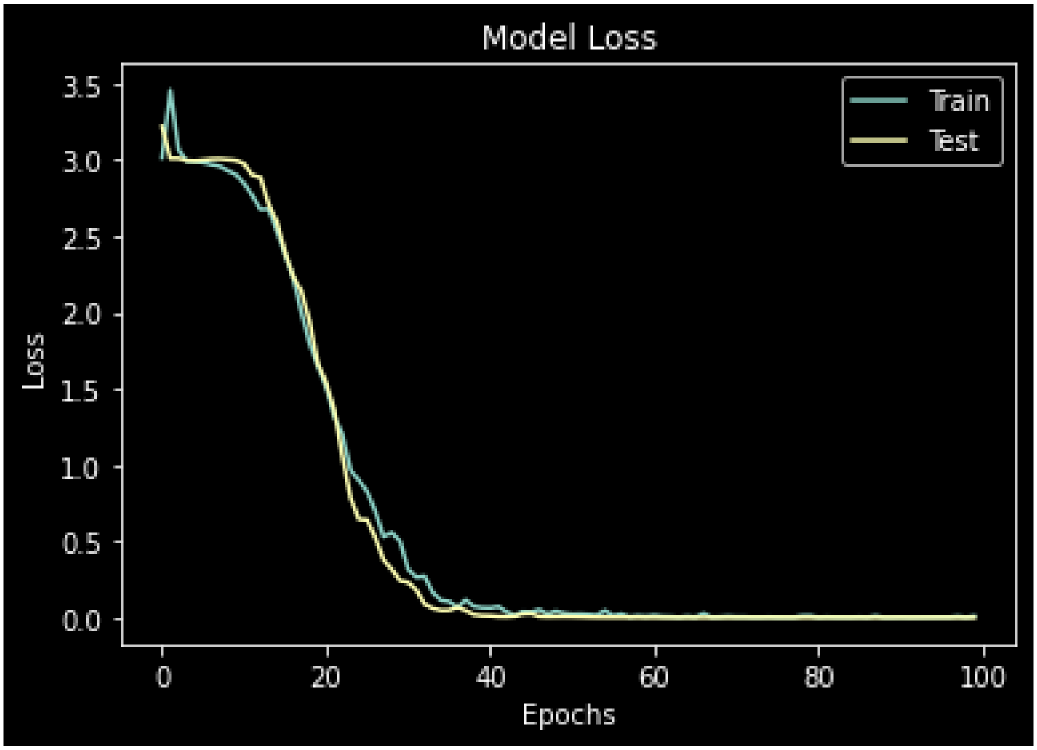
```
5/5 [==============================] - 7s 631ms/step - loss: 0.5774 - accuracy:
0.9250
Test Loss 0.5774
Test Accuracy 0.9250
```

7 7. Plot the result

[21]:
```python
with plt.style.context('dark_background'):
    # Summarize history for Accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend(['Train','Test'],loc='best')
    plt.show()

    # Summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Train','Test'],loc='best')
plt.show()
```



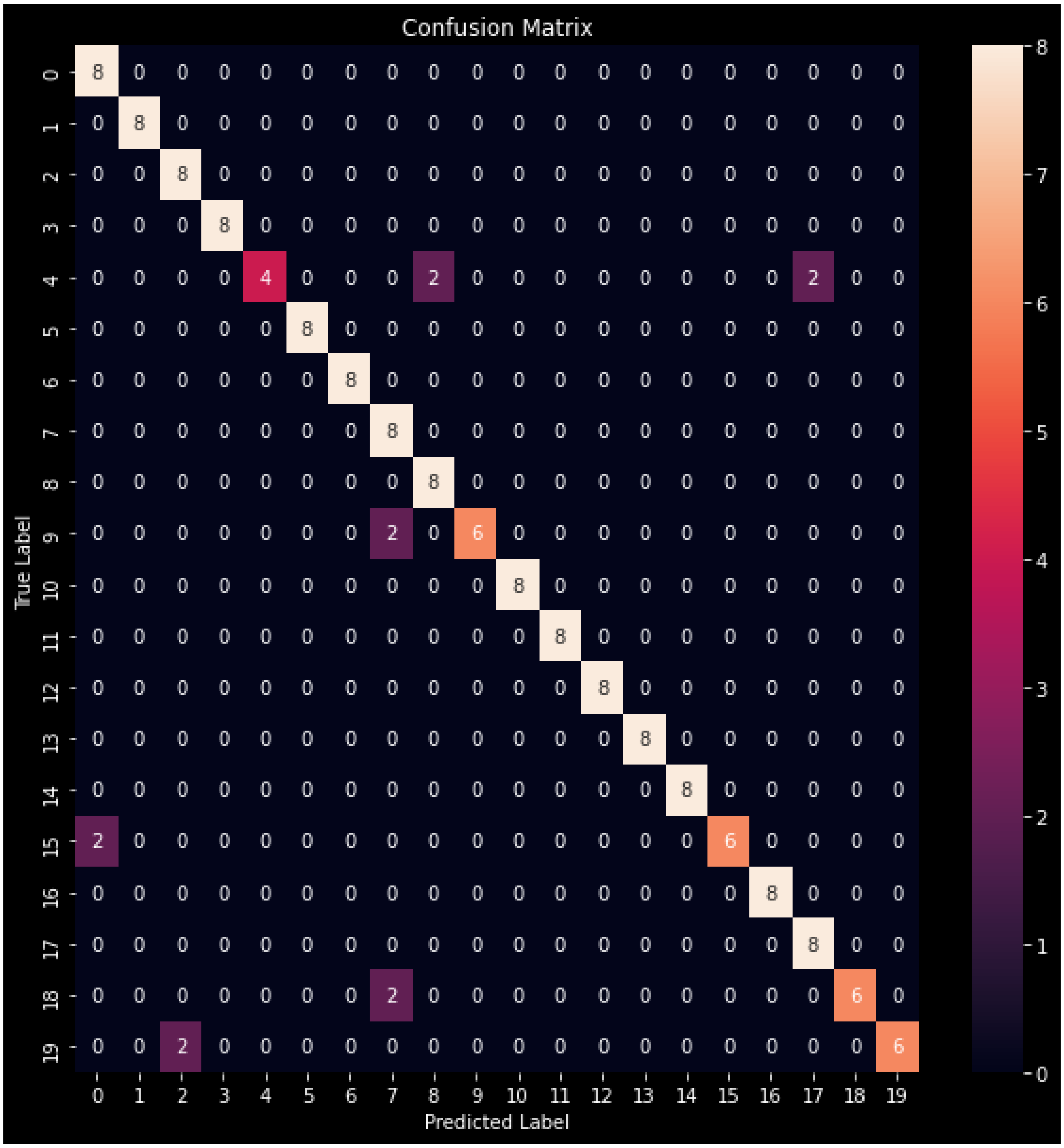


8 8. Iterate the model until the accuracy is above 90%

[22]:
```
y_pred = model.predict(xtest)
y_pred = np.argmax(y_pred,axis=1)
```

```
5/5 [==============================] - 12s 542ms/step
```

[23]:
```
with plt.style.context('dark_background'):
    cm = confusion_matrix(ytest,y_pred)
    plt.figure(figsize=(10,10))
    f = sns.heatmap(cm,annot=True,fmt='d')
    f.set_title('Confusion Matrix',color='white')
    plt.xlabel('Predicted Label',color='white')
    plt.ylabel('True Label',color='white')
    plt.show()
print(classification_report(ytest,y_pred))
```



	precision	recall	f1-score	support
0	0.80	1.00	0.89	8
1	1.00	1.00	1.00	8
2	0.80	1.00	0.89	8
3	1.00	1.00	1.00	8
4	1.00	0.50	0.67	8
5	1.00	1.00	1.00	8
6	1.00	1.00	1.00	8
7	0.67	1.00	0.80	8
8	0.80	1.00	0.89	8

```
           9       1.00      0.75      0.86         8
          10       1.00      1.00      1.00         8
          11       1.00      1.00      1.00         8
          12       1.00      1.00      1.00         8
          13       1.00      1.00      1.00         8
          14       1.00      1.00      1.00         8
          15       1.00      0.75      0.86         8
          16       1.00      1.00      1.00         8
          17       0.80      1.00      0.89         8
          18       1.00      0.75      0.86         8
          19       1.00      0.75      0.86         8

    accuracy                           0.93       160
   macro avg       0.94      0.93      0.92       160
weighted avg       0.94      0.93      0.92       160
```

[ ]: