LAB-03

Aim: To build a Convolutional Neural Network and use it to classify faces

```
!unzip /content/Face-Images
```

```
Archive:  /content/Face-Images.zip
   creating: Face Images/
  inflating: Face Images/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/Face Images/
  inflating: __MACOSX/Face Images/._.DS_Store
   creating: Face Images/Final Training Images/
  inflating: Face Images/Final Training Images/.DS_Store
   creating: __MACOSX/Face Images/Final Training Images/
  inflating: __MACOSX/Face Images/Final Training Images/._.DS_Store
   creating: Face Images/Final Training Images/face12/
  inflating: Face Images/Final Training
Images/face12/image_0281_Face_1.jpg
   creating: __MACOSX/Face Images/Final Training Images/face12/
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0281_Face_1.jpg
  inflating: Face Images/Final Training
Images/face12/image_0284_Face_1.jpg
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0284_Face_1.jpg
  inflating: Face Images/Final Training
Images/face12/image_0283_Face_2.jpg
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0283_Face_2.jpg
  inflating: Face Images/Final Training
Images/face12/image_0279_Face_1.jpg
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0279_Face_1.jpg
  inflating: Face Images/Final Training
Images/face12/image_0286_Face_1.jpg
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0286_Face_1.jpg
  inflating: Face Images/Final Training
Images/face12/image_0277_Face_1.jpg
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0277_Face_1.jpg
  inflating: Face Images/Final Training
Images/face12/image_0285_Face_1.jpg
  inflating: __MACOSX/Face Images/Final Training
Images/face12/._image_0285_Face_1.jpg
  inflating: Face Images/Final Training
```

```
Images/face1/._4face1.jpg
  inflating: Face Images/Final Testing Images/face1/2face1.jpg
  inflating: __MACOSX/Face Images/Final Testing
Images/face1/._2face1.jpg
  inflating: Face Images/Final Testing Images/face1/3face1.jpg
  inflating: __MACOSX/Face Images/Final Testing
Images/face1/._3face1.jpg
  inflating: Face Images/Final Testing Images/face1/1face1.jpg
  inflating: __MACOSX/Face Images/Final Testing
Images/face1/._1face1.jpg
  inflating: __MACOSX/Face Images/Final Testing Images/._face1
  inflating: __MACOSX/Face Images/._Final Testing Images
  inflating: __MACOSX/._Face Images
```

```
from keras.preprocessing.image import ImageDataGenerator
import warnings
warnings.filterwarnings("ignore")
trainpath='/content/Face Images/Final Training Images'
testpath='/content/Face Images/Final Testing Images'
train_datagen=ImageDataGenerator(shear_range=0.1, zoom_range=0.1,
horizontal_flip=True)

test_datagen = ImageDataGenerator()

training_set = train_datagen.flow_from_directory(trainpath,
target_size=(64, 64), batch_size=32, class_mode='categorical')
# training_set = training_set.repeat()
```

```
Found 244 images belonging to 16 classes.
```

```
test_set = test_datagen.flow_from_directory(testpath, target_size=(64,
64), batch_size=32, class_mode='categorical')
```

```
Found 64 images belonging to 16 classes.
```

```
test_set.class_indices
```

```
{'face1': 0,
 'face10': 1,
 'face11': 2,
 'face12': 3,
 'face13': 4,
 'face14': 5,
 'face15': 6,
 'face16': 7,
 'face2': 8,
 'face3': 9,
 'face4': 10,
 'face5': 11,
 'face6': 12,
 'face7': 13,
```

```
 'face8': 14,
 'face9': 15}

TrainClasses=training_set.class_indices

ResultMap={}
for faceValue,faceName in
zip(TrainClasses.values(),TrainClasses.keys()):
  ResultMap[faceValue]=faceName

import pickle
with open("ResultsMap.pkl", 'wb') as fileWriteStream:
  pickle.dump(ResultMap, fileWriteStream)

print("Mapping of Face and its ID",ResultMap)

Mapping of Face and its ID {0: 'face1', 1: 'face10', 2: 'face11', 3:
'face12', 4: 'face13', 5: 'face14', 6: 'face15', 7: 'face16', 8:
'face2', 9: 'face3', 10: 'face4', 11: 'face5', 12: 'face6', 13:
'face7', 14: 'face8', 15: 'face9'}

OutputNeurons=len(ResultMap)
print('\n The Number of output neurons: ', OutputNeurons)


 The Number of output neurons:  16

from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPool2D
from keras.layers import Flatten
from keras.layers import Dense

classifier= Sequential()
classifier.add(Convolution2D(32, kernel_size=(5, 5), strides=(1, 1),
input_shape=(64,64,3), activation='relu'))
classifier.add(MaxPool2D(pool_size=(2,2)))
classifier.add(Convolution2D(64, kernel_size=(5, 5), strides=(1, 1),
activation='relu'))
classifier.add(MaxPool2D(pool_size=(2,2)))
classifier.add(Flatten())
classifier.add(Dense(64, activation='relu'))
classifier.add(Dense(OutputNeurons, activation='softmax'))
classifier.compile(loss='categorical_crossentropy', optimizer =
'adam', metrics=["accuracy"])

import time
StartTime=time.time()
classifier.fit( training_set, steps_per_epoch=len(training_set),
epochs=20, validation_data=test_set, validation_steps=len(test_set))
EndTime=time.time()
```

```
Epoch 1/20
8/8 [==============================] - 4s 482ms/step - loss: 3.2526 -
accuracy: 0.0738 - val_loss: 2.8454 - val_accuracy: 0.1094
Epoch 2/20
8/8 [==============================] - 3s 320ms/step - loss: 2.7659 -
accuracy: 0.0902 - val_loss: 2.6836 - val_accuracy: 0.1094
Epoch 3/20
8/8 [==============================] - 3s 315ms/step - loss: 2.6392 -
accuracy: 0.1025 - val_loss: 2.5382 - val_accuracy: 0.1875
Epoch 4/20
8/8 [==============================] - 4s 443ms/step - loss: 2.4145 -
accuracy: 0.1844 - val_loss: 2.2007 - val_accuracy: 0.3594
Epoch 5/20
8/8 [==============================] - 3s 324ms/step - loss: 2.1442 -
accuracy: 0.3607 - val_loss: 1.8385 - val_accuracy: 0.4531
Epoch 6/20
8/8 [==============================] - 3s 319ms/step - loss: 1.9226 -
accuracy: 0.3566 - val_loss: 1.7587 - val_accuracy: 0.6250
Epoch 7/20
8/8 [==============================] - 3s 301ms/step - loss: 1.4037 -
accuracy: 0.5492 - val_loss: 1.2449 - val_accuracy: 0.6406
Epoch 8/20
8/8 [==============================] - 3s 305ms/step - loss: 0.8128 -
accuracy: 0.7541 - val_loss: 0.7229 - val_accuracy: 0.8750
Epoch 9/20
8/8 [==============================] - 2s 299ms/step - loss: 0.6314 -
accuracy: 0.8074 - val_loss: 0.4700 - val_accuracy: 0.9375
Epoch 10/20
8/8 [==============================] - 4s 472ms/step - loss: 0.2749 -
accuracy: 0.9385 - val_loss: 0.4018 - val_accuracy: 0.8750
Epoch 11/20
8/8 [==============================] - 3s 314ms/step - loss: 0.2607 -
accuracy: 0.9221 - val_loss: 0.4983 - val_accuracy: 0.8594
Epoch 12/20
8/8 [==============================] - 3s 304ms/step - loss: 0.3857 -
accuracy: 0.8852 - val_loss: 0.1844 - val_accuracy: 0.9531
Epoch 13/20
8/8 [==============================] - 3s 311ms/step - loss: 0.2303 -
accuracy: 0.9180 - val_loss: 0.2019 - val_accuracy: 0.9375
Epoch 14/20
8/8 [==============================] - 3s 306ms/step - loss: 0.1888 -
accuracy: 0.9631 - val_loss: 0.2038 - val_accuracy: 0.9375
Epoch 15/20
8/8 [==============================] - 3s 313ms/step - loss: 0.1561 -
accuracy: 0.9590 - val_loss: 0.1614 - val_accuracy: 0.9375
Epoch 16/20
8/8 [==============================] - 3s 301ms/step - loss: 0.1422 -
accuracy: 0.9631 - val_loss: 0.1285 - val_accuracy: 0.9531
Epoch 17/20
8/8 [==============================] - 4s 451ms/step - loss: 0.1378 -
```

```
accuracy: 0.9508 - val_loss: 0.2485 - val_accuracy: 0.9219
Epoch 18/20
8/8 [==============================] - 3s 318ms/step - loss: 0.1776 -
accuracy: 0.9590 - val_loss: 0.1727 - val_accuracy: 0.9531
Epoch 19/20
8/8 [==============================] - 3s 311ms/step - loss: 0.0931 -
accuracy: 0.9754 - val_loss: 0.0921 - val_accuracy: 0.9844
Epoch 20/20
8/8 [==============================] - 3s 321ms/step - loss: 0.0789 -
accuracy: 0.9795 - val_loss: 0.0393 - val_accuracy: 0.9844
```

```
import numpy as np
from keras.preprocessing import image
ImagePath='/content/Face Images/Final Testing
Images/face12/1face12.jpg'
test_image=image.load_img(ImagePath,target_size=(64, 64))
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
result=classifier.predict(test_image,verbose=0)
print('Prediction is: ',ResultMap[np.argmax(result)])
```

```
Prediction is:  face12
```