```
import cv2
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.flush_and_unmount()
```

```
from keras.preprocessing.image import ImageDataGenerator
```

```
# Defining pre-processing transformations on raw images of training data
# These hyper parameters helps to generate slightly twisted versions
# of the original image, which leads to a better model, since it learns
# on the good and bad mix of images

train_datagen = ImageDataGenerator(
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True)
```

```
#we wont apply any pre processimg on the raw images of the test dataset

test_datagen = ImageDataGenerator()
```

```
trainingImagePath = '/content/test/face/Face Images/Final Training Images'
testImagePath = '/content/test/face/Face Images/Final Testing Images'
```

```
# Generating the Training Data
training_set = train_datagen.flow_from_directory(
        trainingImagePath,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')
```

```
Found 256 images belonging to 17 classes.
```

```
# Generating the Testing Data
test_set = test_datagen.flow_from_directory(
        testImagePath,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')
```

```
Found 66 images belonging to 17 classes.
```

```
# Printing class labels for each face
test_set.class_indices
```

```
{'face1': 0,
 'face10': 1,
 'face11': 2,
 'face12': 3,
 'face13': 4,
 'face14': 5,
 'face15': 6,
 'face16': 7,
 'face17': 8,
 'face2': 9,
 'face3': 10,
 'face4': 11,
 'face5': 12,
 'face6': 13,
 'face7': 14,
 'face8': 15,
 'face9': 16}
```

```
# class_indices have the numeric tag for each face
TrainClasses=training_set.class_indices

# Storing the face and the numeric tag for future reference
ResultMap={}
for faceValue,faceName in zip(TrainClasses.values(),TrainClasses.keys()):
    ResultMap[faceValue]=faceName

# Saving the face map for future reference
import pickle
with open("ResultsMap.pkl", 'wb') as fileWriteStream:
    pickle.dump(ResultMap, fileWriteStream)

# The model will give answer as a numeric tag
# This mapping will help to get the corresponding face name for it
print("Mapping of Face and its ID",ResultMap)

# The number of neurons for the output layer is equal to the number of faces
OutputNeurons=len(ResultMap)
print('\n The Number of output neurons: ', OutputNeurons)
```

```
Mapping of Face and its ID {0: 'face1', 1: 'face10', 2: 'face11', 3: 'face12', 4: 'face13', 5: 'face14', 6: 'face15', 7: 'face16', 8

 The Number of output neurons:  17
```

```
# so in our CNN model we would have:
#2 hideen convolutional layers
#2 hidden pooling layers
#16 neurons in the output layer since we have 17 classes
# and 1 flattening layer
```

```
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
```

```
'''Initializing the Convolutional Neural Network'''
classifier= Sequential()

''' STEP--1 Convolution
# Adding the first layer of CNN
# we are using the format (64,64,3) because we are using TensorFlow backend
# It means 3 matrix of size (64X64) pixels representing Red, Green and Blue components of pixels
'''
classifier.add(Convolution2D(32, kernel_size=(5, 5), strides=(1, 1), input_shape=(64,64,3), activation='relu'))

'''STEP--2 MAX Pooling'''
classifier.add(MaxPool2D(pool_size=(2,2)))

''' ADDITIONAL LAYER of CONVOLUTION for better accuracy'''
classifier.add(Convolution2D(64, kernel_size=(5, 5), strides=(1, 1), activation='relu'))

classifier.add(MaxPool2D(pool_size=(2,2)))

''' STEP--3 FLattening'''
classifier.add(Flatten())

'''STEP--4 Fully Connected Neural Network'''
classifier.add(Dense(64, activation='relu'))

classifier.add(Dense(OutputNeurons, activation='softmax'))

'''Compiling the CNN'''
#classifier.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
classifier.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics=["accuracy"])



# Starting the model training
classifier.fit(
    training_set,
    steps_per_epoch=8,         #number of steps per epoch = (Total number of training samples / Batch size), here i have 244 training in
    epochs=15,
    validation_data=test_set,
    validation_steps=10
)
```

```
Epoch 1/15
8/8 [==============================] - ETA: 0s - loss: 63.5780 - accuracy: 0.0586WARNING:tensorflow:Your input ran out of data; inte
```

```
8/8 [==============================] - 5s 518ms/step - loss: 63.5780 - accuracy: 0.0586 - val_loss: 2.8539 - val_accuracy: 0.0303
Epoch 2/15
8/8 [==============================] - 4s 543ms/step - loss: 2.7385 - accuracy: 0.1367
Epoch 3/15
8/8 [==============================] - 3s 364ms/step - loss: 2.6010 - accuracy: 0.2109
Epoch 4/15
8/8 [==============================] - 4s 482ms/step - loss: 2.3415 - accuracy: 0.2773
Epoch 5/15
8/8 [==============================] - 5s 547ms/step - loss: 1.8141 - accuracy: 0.4688
Epoch 6/15
8/8 [==============================] - 3s 369ms/step - loss: 1.3341 - accuracy: 0.6016
Epoch 7/15
8/8 [==============================] - 3s 383ms/step - loss: 0.8897 - accuracy: 0.7305
Epoch 8/15
8/8 [==============================] - 4s 544ms/step - loss: 1.1695 - accuracy: 0.6562
Epoch 9/15
8/8 [==============================] - 4s 407ms/step - loss: 1.1351 - accuracy: 0.6406
Epoch 10/15
8/8 [==============================] - 3s 363ms/step - loss: 0.7475 - accuracy: 0.7539
Epoch 11/15
8/8 [==============================] - 4s 423ms/step - loss: 0.5683 - accuracy: 0.8242
Epoch 12/15
8/8 [==============================] - 4s 509ms/step - loss: 0.4082 - accuracy: 0.8711
Epoch 13/15
8/8 [==============================] - 3s 378ms/step - loss: 0.3045 - accuracy: 0.9102
Epoch 14/15
8/8 [==============================] - 3s 373ms/step - loss: 0.1307 - accuracy: 0.9609
Epoch 15/15
8/8 [==============================] - 5s 582ms/step - loss: 0.1210 - accuracy: 0.9570
<keras.src.callbacks.History at 0x7d9c9f3ecdf0>
```

```
'''Making single predictions'''
import numpy as np
from keras.preprocessing import image

ImagePath='/content/test/face/Face Images/Final Testing Images/face17/PXL_20240319_081525750.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(training_set.class_indices)


print('Prediction is: ',ResultMap[np.argmax(result)])
```

```
Prediction is:  face17
```

```

```