REPORT

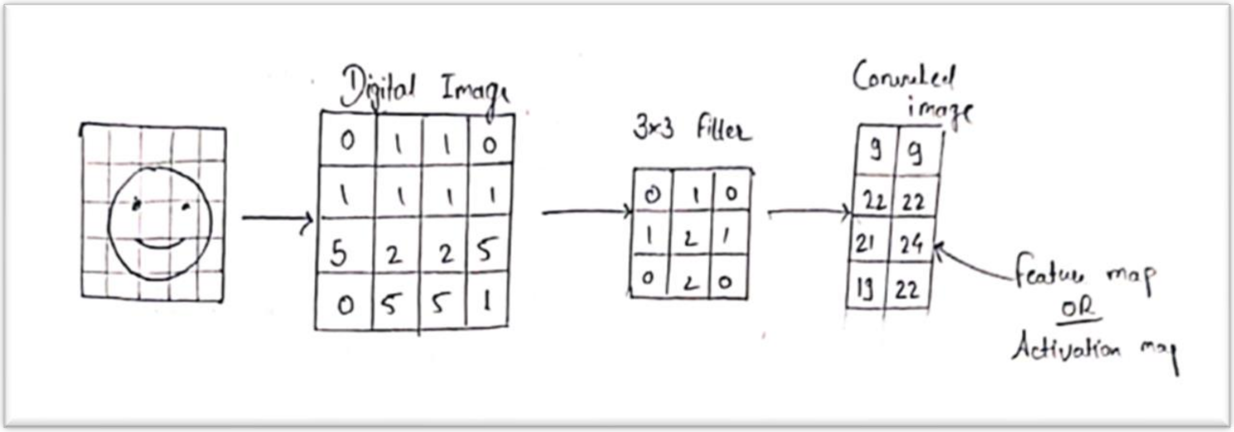
Members: - Shubham Sontakke (BT19CSE107)

Topic: - Face Recognition

Theory: -

Currently face recognition *method is becoming a more and more important technique in our social lives. Face* recognition *has been an active research area with many successful traditional and deep learning methods. In our project we are introducing convolutional neural network method to tackle the face* recognition *to maintain a high accuracy.*

A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images.



0	1	1	0
1	1	1	1
5	2	2	5
0	5	5	1

0	1	0
1	2	1
0	2	0

9	9
22	22
21	24
19	22

Convolutional neural network (CNN) changed the way we used to learn images. It made it very network CNN mimics the way humans see images, by focusing on one portion of the image at a time and scanning the whole image.

CNN boils down every image as a vector of numbers, which can be learned by the fully connected Dense layers of ANN.

How does facial recognition work?

Many people are familiar with face recognition technology through the FaceID used to unlock iPhones (however, this is only one application of face recognition). Typically, facial recognition does not rely on a massive database of photos to determine an individual's identity — it simply identifies and recognizes one person as the sole owner of the device, while limiting access to others.

Beyond unlocking phones, facial recognition works by matching the faces of people walking past special cameras, to images of people on a watch list. The watch lists can contain pictures of anyone, including people who are not suspected of any wrongdoing, and the images can come from anywhere — even from our social media accounts. Facial technology systems can vary, but in general, they tend to operate as follows:

Step 1: Face detection

The camera detects and locates the image of a face, either alone or in a crowd. The image may show the person looking straight ahead or in profile.

Step 2: Face analysis

Next, an image of the face is captured and analyzed. Most facial recognition technology relies on 2D rather than 3D images because it can more conveniently match a 2D image with public photos or those in a database. The software reads the geometry of your face. Key factors include the distance between your eyes, the depth of your eye sockets, the distance from forehead to chin, the shape of your cheekbones, and the contour of the lips, ears, and chin. The aim is to identify the facial landmarks that are key to distinguishing your face.

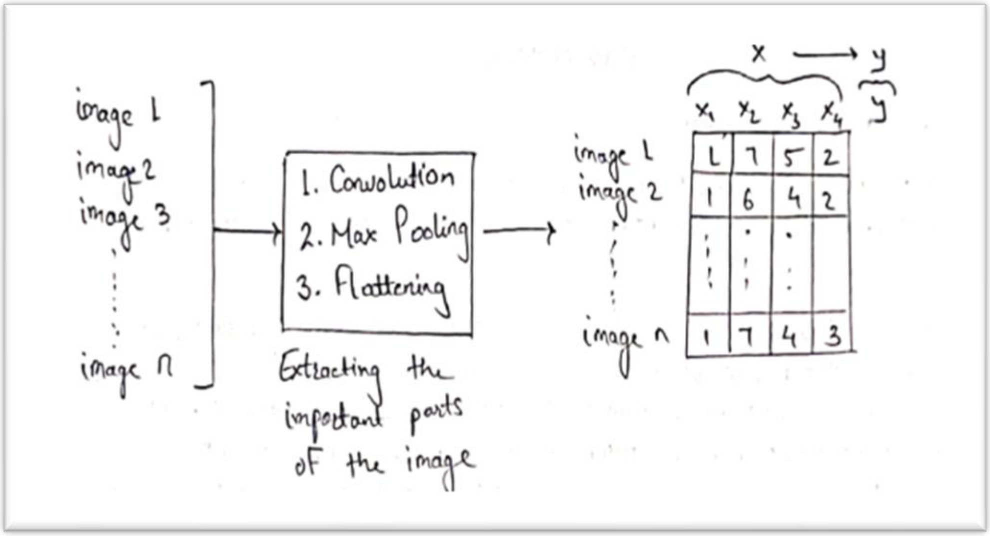
Step 3: Converting the image to data

The face capture process transforms analog information (a face) into a set of digital information (data) based on the person's facial features. Your face's analysis is essentially turned into a mathematical formula. The numerical code is called a faceprint. In the same way that thumbprints are unique, each person has their own faceprint.

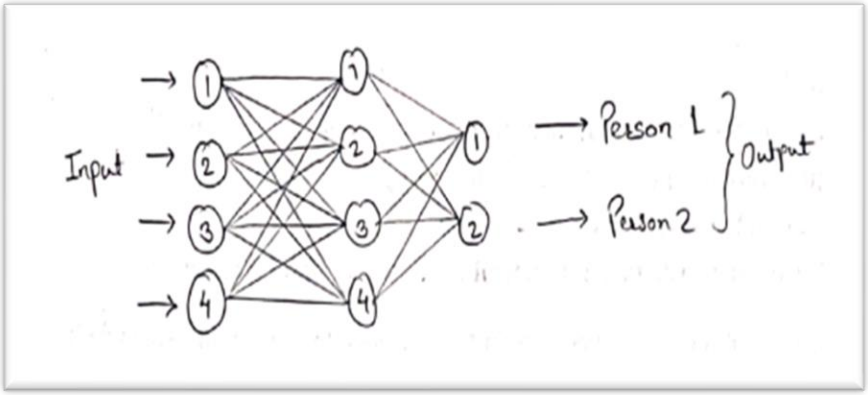
Step 4: Finding a match

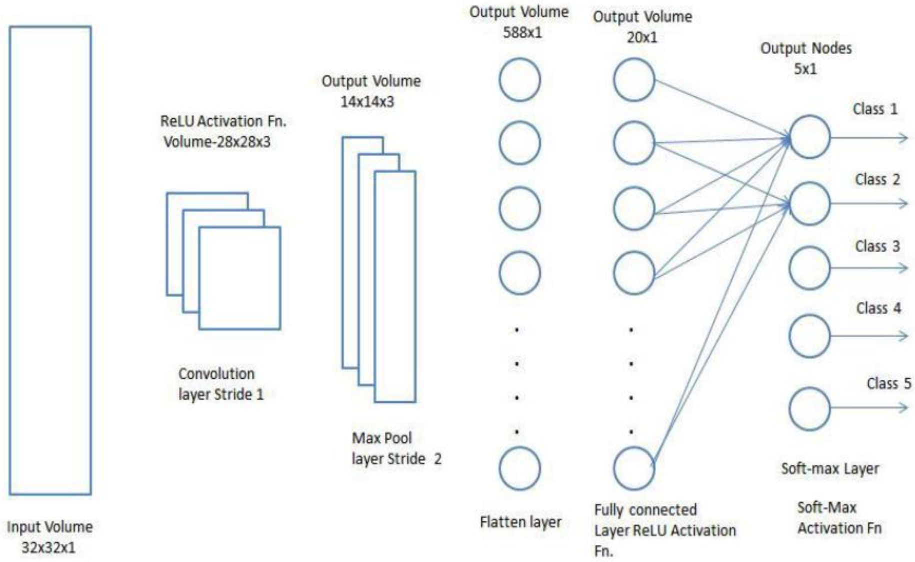
Your faceprint is then compared against a database of other known faces. On Facebook, any photo tagged with a person’s name becomes a part of Facebook's database, which may also be used for facial recognition. If your faceprint matches an image in a facial recognition database, then a determination is made.

Below diagram summarizes the overall flow of CNN algorithm.



	x_1	x_2	x_3	x_4
image 1	1	7	5	2
image 2	1	6	4	2
⋮	⋮	⋮	⋮	
image n	1	7	4	3

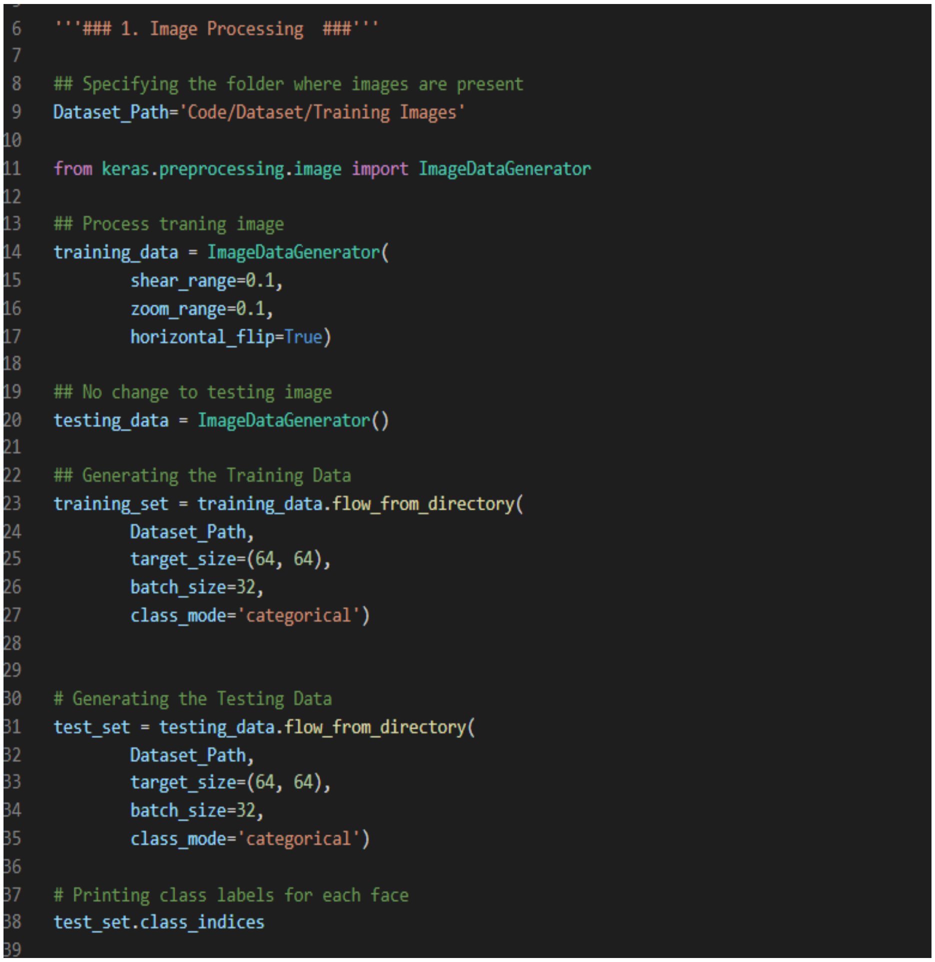




Code:

1. *Image Processing*-
    - Dataset folder is specified
    - Defining pre-processing transformations on raw images of training data. These hyper parameters help 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
    - Defining pre-processing transformations on raw images of testing data. No extra transformations are done on the testing images
    - Generating the Testing & training Data using flow_from_directory()
    - Printing class labels for each face

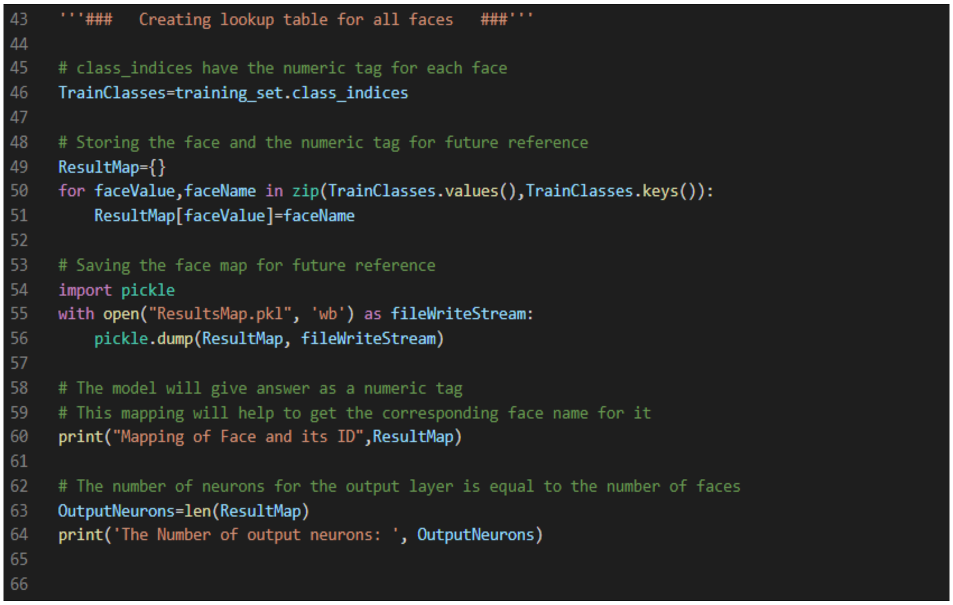
The data contains cropped face images of 6 people divided into Training and testing. We will train the CNN model using the images in the dataset folder and then test the model by using the any random image of dataset folder, to check if the model is able to recognize the face name of the images or not.



```
'''### 1. Image Processing  ###'''

## Specifying the folder where images are present
Dataset_Path='Code/Dataset/Training Images'

from keras.preprocessing.image import ImageDataGenerator

## Process traning image
training_data = ImageDataGenerator(
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True)

## No change to testing image
testing_data = ImageDataGenerator()

## Generating the Training Data
training_set = training_data.flow_from_directory(
        Dataset_Path,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')


# Generating the Testing Data
test_set = testing_data.flow_from_directory(
        Dataset_Path,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

# Printing class labels for each face
test_set.class_indices

```

2. *Create lookup table for all face: -*

- Giving number & label to each face. class_indices have the numeric tag for each face. The class_index dictionary has face names as keys and the numeric mapping as values. We need to swap it, because the classifier model will return the answer as the numeric mapping and we need to get the face-name out of it.
- Storing the face and the numeric tag for future reference
- Saving the face map for future reference
- Note:- The model will give answer as a numeric tag. This mapping will help to get the corresponding face name for it.
- We are counting the number of unique faces, as that will be used as the number of output neurons in the output layer of fully connected CNN classifier. At last display the number of neurons for the output layer i.e. equal to the number of faces

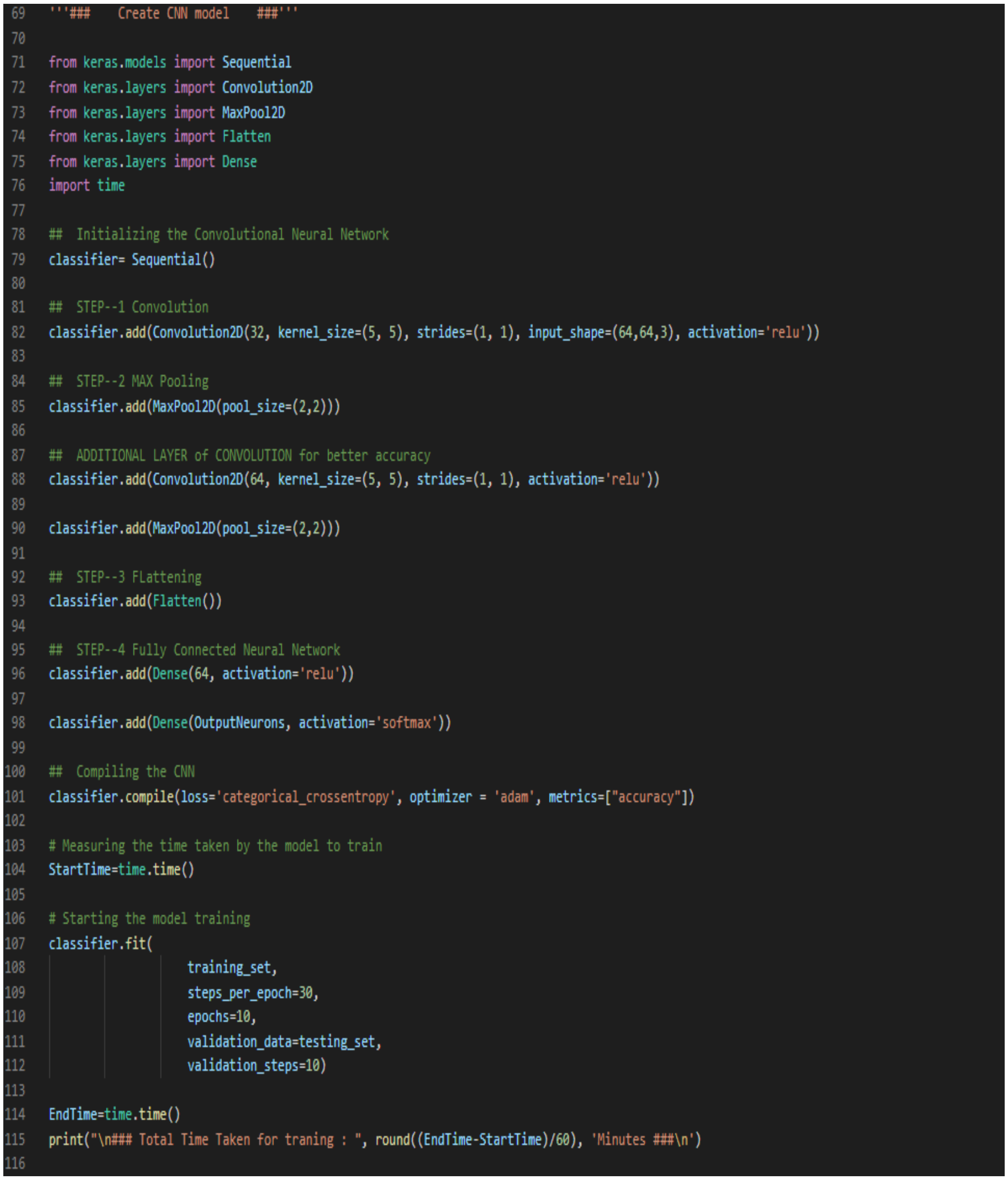


```
'''###   Creating lookup table for all faces   ###'''

# 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('The Number of output neurons: ', OutputNeurons)
```

3. *Create CNN model: -*

- Initializing the Convolutional Neural Network
- 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(1st hidden layers of convolution)
- Adding MAX Pooling(2 hidden layers of max pooling)
- Apply ADDITIONAL LAYER of CONVOLUTION for better accuracy(2nd hidden layers of convolution)
- FLattening for 2D Image(1 layer of flattening)
- Also 1 Hidden ANN layer & 1 output layer with 6-neurons
- Fully Connected Neural Network
- Compiling the CNN using compile()

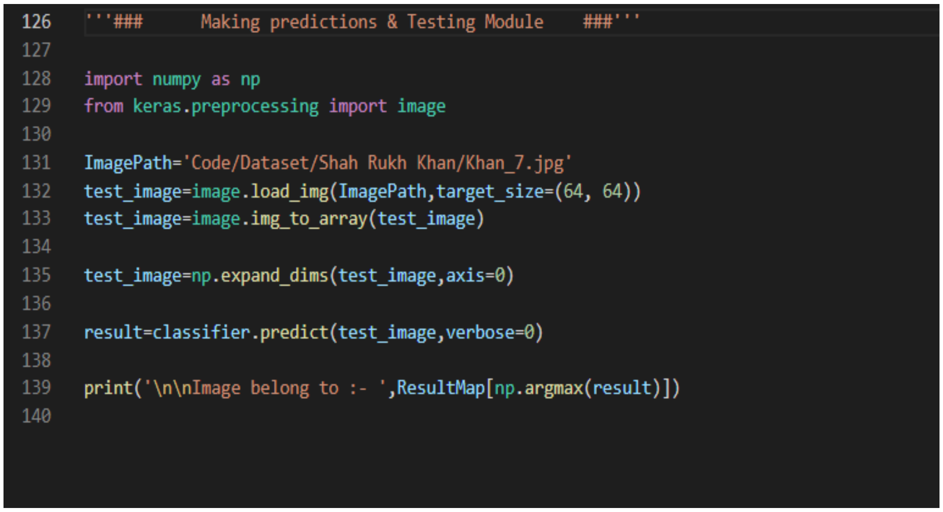
- Starting the model training using module.fit()
- Measuring & printing the time taken by the model to train using time()



```python
'''###    Create CNN model    ###'''

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
import time

##  Initializing the Convolutional Neural Network
classifier= Sequential()

##  STEP--1 Convolution
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='categorical_crossentropy', optimizer = 'adam', metrics=["accuracy"])

# Measuring the time taken by the model to train
StartTime=time.time()

# Starting the model training
classifier.fit(
                    training_set,
                    steps_per_epoch=30,
                    epochs=10,
                    validation_data=testing_set,
                    validation_steps=10)

EndTime=time.time()
print("\n### Total Time Taken for traning : ", round((EndTime-StartTime)/60), 'Minutes ###\n')
```

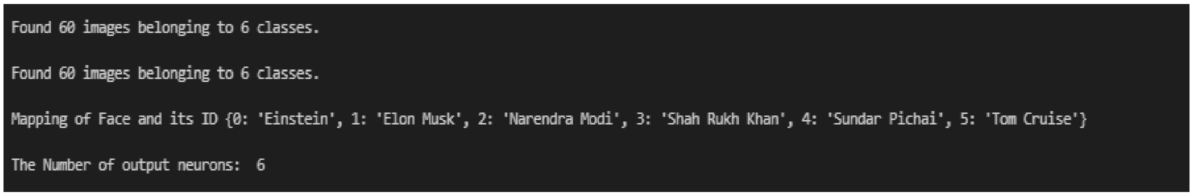
4. *Finally testing model by Making single predictions:-*

- Specify image path & load image
- After that convert to array data
- And predict the image using predict()
- Display result

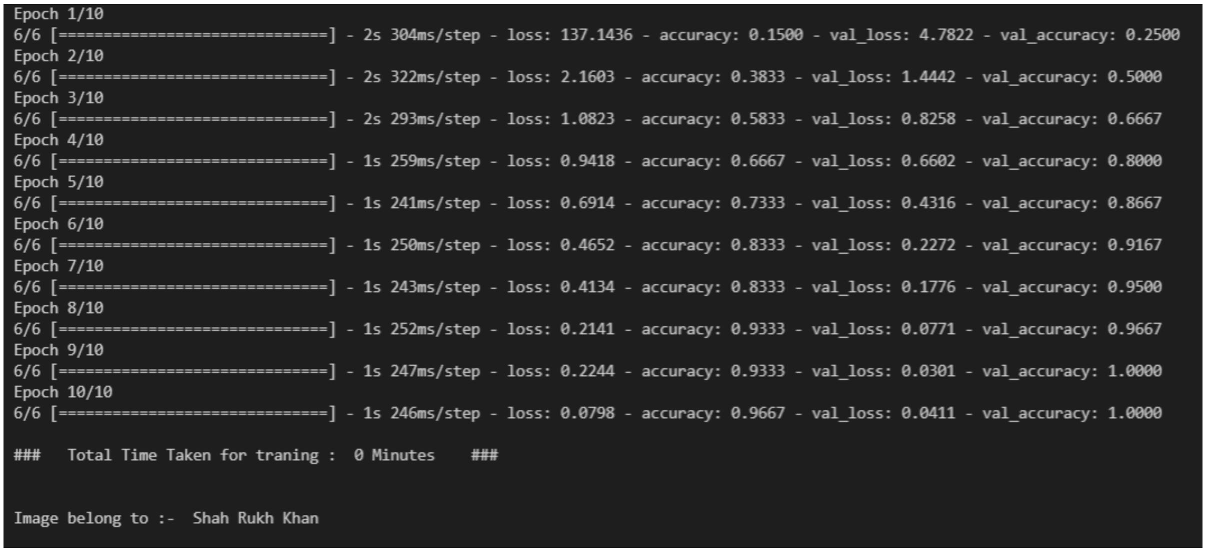


```python
'''###       Making predictions & Testing Module     ###'''

import numpy as np
from keras.preprocessing import image

ImagePath='Code/Dataset/Shah Rukh Khan/Khan_7.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('\n\nImage belong to :- ',ResultMap[np.argmax(result)])

```

Console Result:-



```
Found 60 images belonging to 6 classes.

Found 60 images belonging to 6 classes.

Mapping of Face and its ID {0: 'Einstein', 1: 'Elon Musk', 2: 'Narendra Modi', 3: 'Shah Rukh Khan', 4: 'Sundar Pichai', 5: 'Tom Cruise'}

The Number of output neurons:  6
```



```
Epoch 1/10
6/6 [==============================] - 2s 304ms/step - loss: 137.1436 - accuracy: 0.1500 - val_loss: 4.7822 - val_accuracy: 0.2500
Epoch 2/10
6/6 [==============================] - 2s 322ms/step - loss: 2.1603 - accuracy: 0.3833 - val_loss: 1.4442 - val_accuracy: 0.5000
Epoch 3/10
6/6 [==============================] - 2s 293ms/step - loss: 1.0823 - accuracy: 0.5833 - val_loss: 0.8258 - val_accuracy: 0.6667
Epoch 4/10
6/6 [==============================] - 1s 259ms/step - loss: 0.9418 - accuracy: 0.6667 - val_loss: 0.6602 - val_accuracy: 0.8000
Epoch 5/10
6/6 [==============================] - 1s 241ms/step - loss: 0.6914 - accuracy: 0.7333 - val_loss: 0.4316 - val_accuracy: 0.8667
Epoch 6/10
6/6 [==============================] - 1s 250ms/step - loss: 0.4652 - accuracy: 0.8333 - val_loss: 0.2272 - val_accuracy: 0.9167
Epoch 7/10
6/6 [==============================] - 1s 243ms/step - loss: 0.4134 - accuracy: 0.8333 - val_loss: 0.1776 - val_accuracy: 0.9500
Epoch 8/10
6/6 [==============================] - 1s 252ms/step - loss: 0.2141 - accuracy: 0.9333 - val_loss: 0.0771 - val_accuracy: 0.9667
Epoch 9/10
6/6 [==============================] - 1s 247ms/step - loss: 0.2244 - accuracy: 0.9333 - val_loss: 0.0301 - val_accuracy: 1.0000
Epoch 10/10
6/6 [==============================] - 1s 246ms/step - loss: 0.0798 - accuracy: 0.9667 - val_loss: 0.0411 - val_accuracy: 1.0000

###   Total Time Taken for traning :  0 Minutes    ###


Image belong to :-  Shah Rukh Khan
```

- Training images & classes
- Testing images & classes
- Mapping of Number tag & Name
- Total output possible
- Then process & time for training
- At last, the model test result.

Conclusion: -

In this project we have done a lot of research on related algorithms for face recognition using CNN before we actually started to implement our own version of neural network, and then we chose to extract some good aspects of these well-developed algorithms and made our own. Since all these mentioned methods have their own strengths and drawbacks, we would like to combine them to achieve an optimal performance to achieve our goal for higher accuracy and lower run time. And through the process of doing the project, we have managed to overcome problems of accuracy (>=95%).