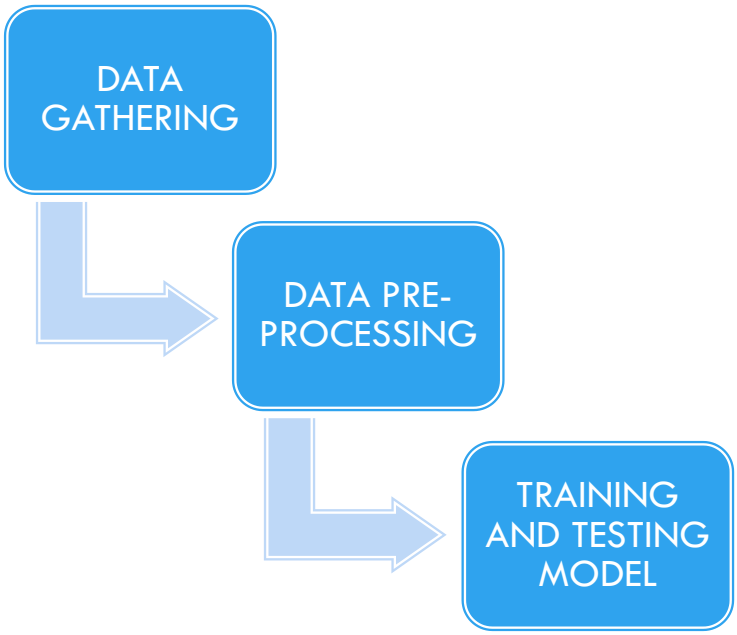
• PROBLEM STATEMENT

Create a project using transfer learning solving various problems like Face Recognition and Image Classification, using existing Deep Learning models like VGG16.

• IMPLEMENTED SOLUTION

In transfer learning, the knowledge of an already trained machine learning model is applied to a different but related problem. With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another.

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.



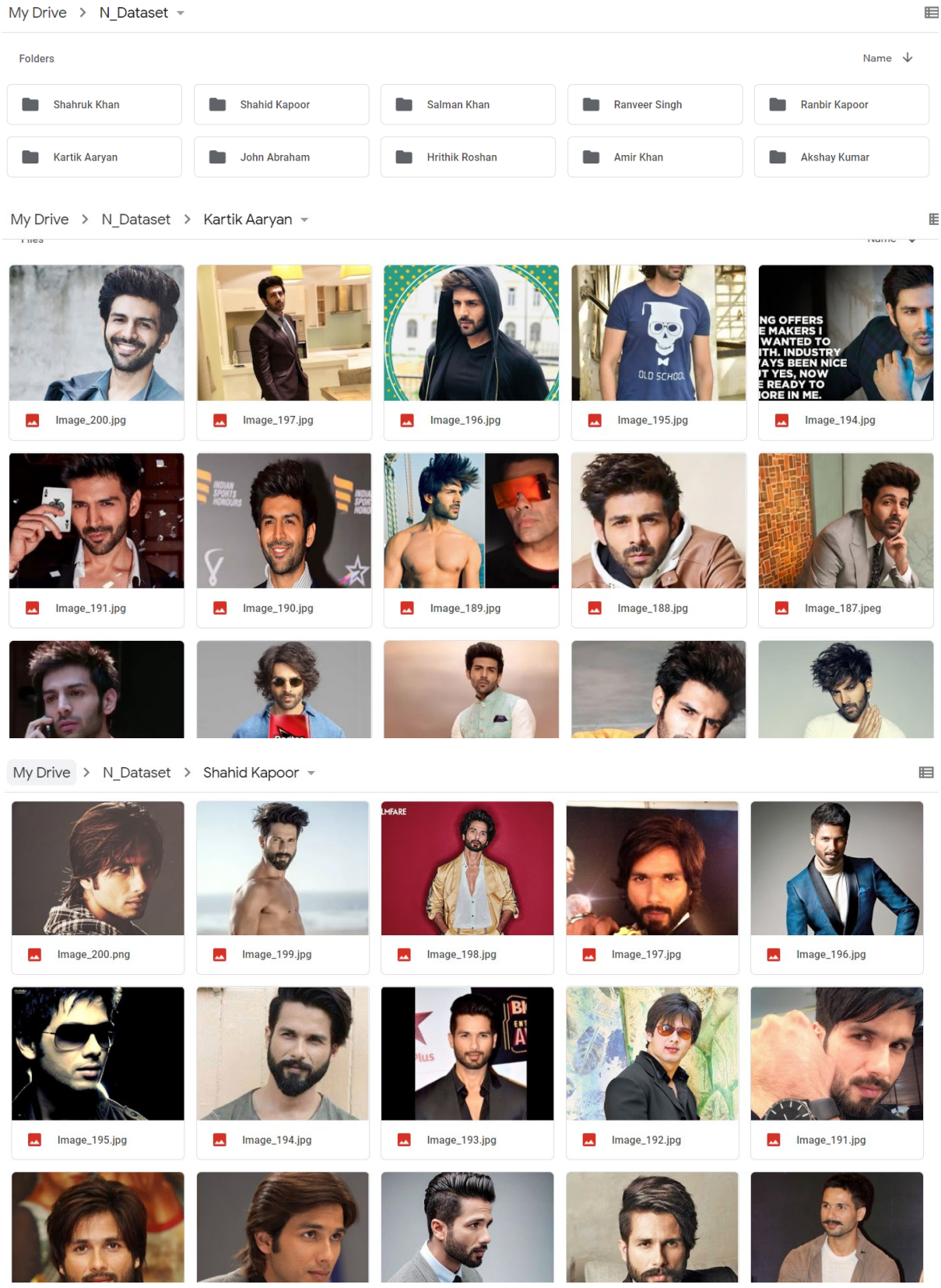
PHASE-I: DATA GATHERING

Used Bing Image Downloader library to download images for dataset. Python library to download bulk of images form Bing.com. This package uses async url, which makes it very fast while downloading.

This program lets you download tons of images from Bing. Downloading 200 images for each of 10 classes.

```
In [1]: !pip install bing-image-downloader

Collecting bing-image-downloader
  Downloading https://files.pythonhosted.org/packages/2c/f9/e827c690d0df1ec2f2f27cf0fb3d1f944c5c56253f8d3750ccaff051b3d3/bing_image_downloader-1.1.0-py3-none-any.whl
Installing collected packages: bing-image-downloader
Successfully installed bing-image-downloader-1.1.0

In [2]: from google.colab import drive
        drive.mount('/content/drive')

Mounted at /content/drive

In [3]: !ls '/content/drive'

MyDrive

In [13]: f_list = ['Akshay Kumar',
          'Amir Khan',
          'Hrithik Roshan',
          'John Abraham',
          'Kartik Aaryan',
          'Ranbir Kapoor',
          'Ranveer Singh',
          'Salman Khan',
          'Shahid Kapoor',
          'Shahruk Khan']
         f_list

Out[13]: ['Akshay Kumar',
          'Amir Khan',
          'Hrithik Roshan',
          'John Abraham',
          'Kartik Aaryan',
          'Ranbir Kapoor',
          'Ranveer Singh',
          'Salman Khan',
          'Shahid Kapoor',
          'Shahruk Khan']

In [9]: from bing_image_downloader import downloader

In [12]: for i in f_list:
           downloader.download(i, limit=200,  output_dir='/content/drive/MyDrive/N_Dataset')

Streaming output truncated to the last 5000 lines.
[%] Downloading Image #179 from https://images.indianexpress.com/2018/01/hrithik-gif.gif
[%] File Downloaded !

[%] Downloading Image #180 from https://www.bing.com/th/id/OGC.761d34788f8abaaf896d5dec14e464b3?pid=1.7&amp;rurl=https%3a%2f%2fimages.indianexpress.com%2f2018%2f01%2fhrithik-gif.gif&amp;ehk=Fk0J7yXACmvXM1j8949DmJUgfQiuunGJgc0zCikdNBM%3d
[!] Issue getting: https://www.bing.com/th/id/OGC.761d34788f8abaaf896d5dec14e464b3?pid=1.7&amp;rurl=https%3a%2f%2fimages.indianexpress.com%2f2018%2f01%2fhrithik-gif.gif&amp;ehk=Fk0J7yXACmvXM1j8949DmJUgfQiuunGJgc0zCikdNBM%3d
[!] Error:: HTTP Error 404: Not Found
[%] Downloading Image #180 from https://i.pinimg.com/originals/6f/2c/51/6f2c5199212938cdb0afd4ecf9e7ff40.jpg
[%] File Downloaded !

[%] Downloading Image #181 from https://i.ebayimg.com/images/g/lpEAAOSwHUhaDW8y/s-l400.jpg
[%] File Downloaded !

[%] Downloading Image #182 from https://images.indianexpress.com/2018/01/hrithik-roshan-7591.jpg

[%] File Downloaded !

[%] Downloading Image #198 from https://i.pinimg.com/originals/fe/84/0b/fe840b95cc6576070f7b0cd38bbe05f5.jpg
[%] File Downloaded !

[%] Downloading Image #199 from https://i.pinimg.com/originals/ab/86/e9/ab86e97ad3a268efab1c83e2667373bf.jpg
[%] File Downloaded !

[%] Downloading Image #200 from https://i.pinimg.com/originals/1f/dd/62/1fdd621367aa4faa152719a803fd4b29.jpg
[%] File Downloaded !


[%] Done. Downloaded 200 images.
================================================

Please show your support here
https://www.buymeacoffee.com/gurugaurav

================================================
```

Sample Dataset gathered after phase-I:



Similar for remaining 8 classes.

PHASE-II: DATA PRE-PROCESSING

After collecting the required images, we extract faces from them and sort these in training and testing dataset.

```
from matplotlib import pyplot
from PIL import Image
from numpy import asarray
from mtcnn.mtcnn import MTCNN
import cv2
import numpy

face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

def face_extractor(photo):
    photo = cv2.imread(photo)
    gray_photo = cv2.cvtColor(photo, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_photo)

    if faces is ():
        return None

    else:
        # Crop all faces found
        for (x,y,w,h) in faces:
            cropped_face = photo[y:y+h, x:x+w]

        return cropped_face
```

```
data_path = '/content/drive/MyDrive/N_Dataset'
```

```
import os

folder_list = os.listdir(data_path)
folder_list
```

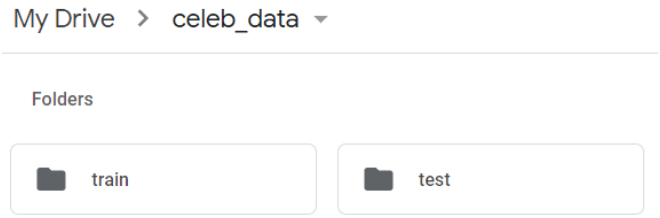
```
['Akshay Kumar',
 'Amir Khan',
 'Hrithik Roshan',
 'John Abraham',
 'Kartik Aaryan',
 'Ranbir Kapoor',
 'Ranveer Singh',
 'Shahruk Khan',
 'Shahid Kapoor',
 'Salman Khan']
```

```
f_list = os.listdir(data_path + '/' + folder_list[8])
len(f_list)
```

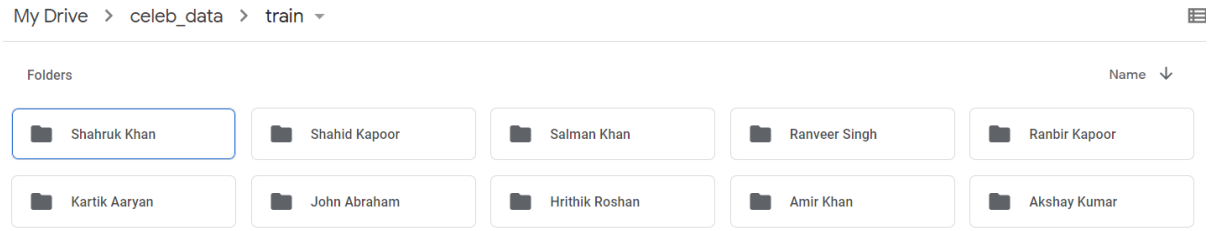
```
190
```

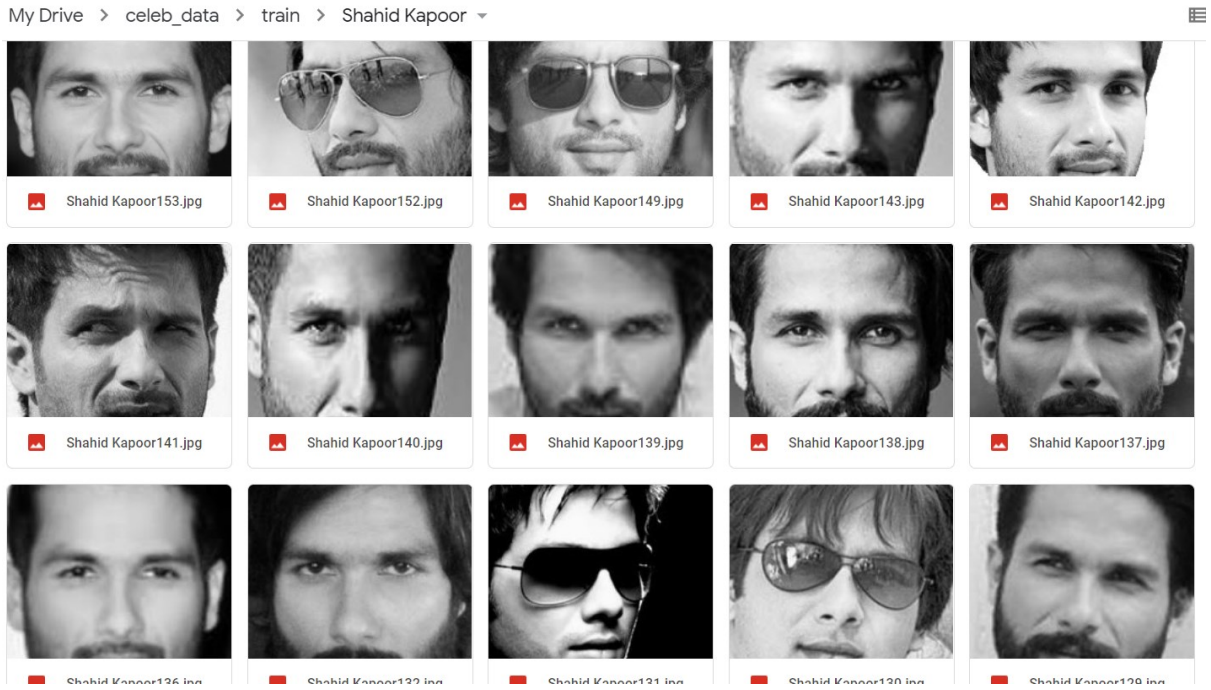
```
for i in folder_list:
  temp_list = os.listdir('/content/drive/MyDrive/N_Dataset/' + i)
  #print(i)
  count = 0
  for j in temp_list:
      photo = ('/content/drive/MyDrive/N_Dataset/' + i + '/' + j)
      #print(j)

      face = face_extractor(photo)

      if face_extractor(photo) is not None:
          count += 1

          face = cv2.resize(face_extractor(photo), (200, 200))
          face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

          # Save file in specified directory with unique name
          file_name_path = '/content/drive/MyDrive/celeb_data/face/' + i + str(count) + '.jpg'
          cv2.imwrite(file_name_path, face)

      else:
          pass
```

Sample Dataset gathered after phase-II:

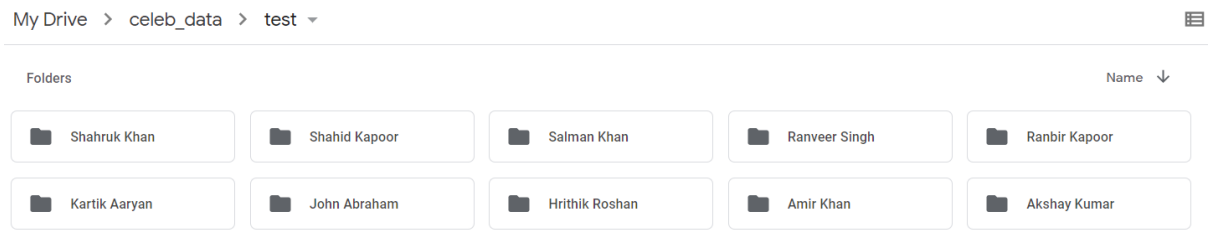


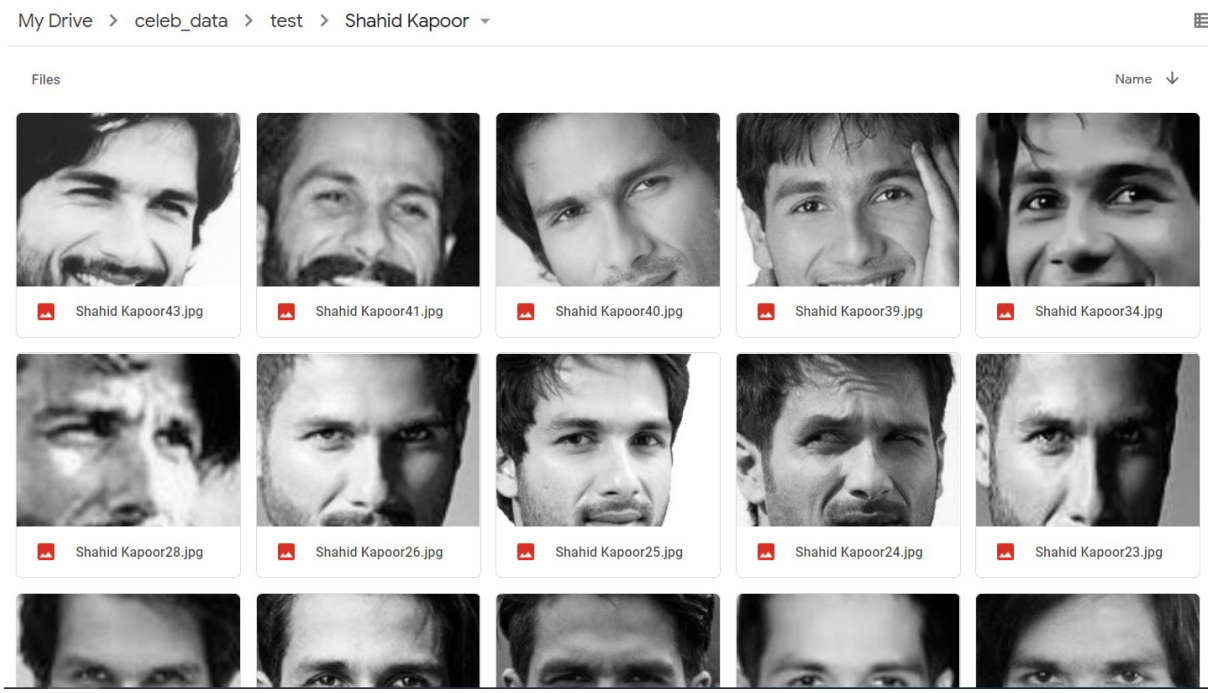
IN TRAIN DATASET, EACH CLASS HAS 75 FACES.





IN TEST DATASET, EACH CLASS HAS 25 FACES





PHASE-III: TRAINING AND TESTING MODEL:

1. Importing libraries

In [3]:
```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

2. Loading dataset

In [4]:
```
#Give dataset path
train_path = '/content/drive/MyDrive/celeb_data/train'
test_path = '/content/drive/MyDrive/celeb_data/test'
```

In [5]:
```
IMAGE_SIZE = [224, 224]
```

In [6]:
```
from PIL import Image
import os
from IPython.display import display
from IPython.display import Image as _Imgdis
# creating a object

folder = train_path+'/Shahruk Khan'

y = [f for f in os.listdir(folder) if os.path.isfile(os.path.join(folder, f))]
print("Working with {0} images".format(len(y)))
print("Image examples: ")

for i in range(5):
    print(y[i])
    display(_Imgdis(filename=folder + "/" + y[i], width=240, height=240))
```

```
Working with 75 images
Image examples:
Shahruk Khan138.jpg
```

```
Shahruk Khan141.jpg
```

3. Creating VGG-16 model

```
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
vgg.input
```

```
<KerasTensor: shape=(None, 224, 224, 3) dtype=float32 (created by layer 'input_5')>
```

Using Fine tuning upto last three layers

```
#Fine Tuning
for layer in vgg.layers[:-3]:
  layer.trainable = False
```

```
folders = glob('/content/drive/MyDrive/celeb_data/train/*')
print(len(folders))
```

```
10
```

```
from keras.layers import Dense, Dropout, Flatten, GlobalAveragePooling2D


x = GlobalAveragePooling2D()(vgg.output)
x = Dense(256,activation='relu')(x)
x = Dropout(0.2)(x) #to avoid overfitting
prediction = Dense(len(folders), activation='softmax')(x)

model = Model(inputs=vgg.input, outputs=prediction)
model.summary()
```

Model Compilation:

```
from keras import optimizers

adam = optimizers.Adam(learning_rate=0.0001)
model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=['accuracy'])
```

Model Summary:

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d_2 (	(None, 512)	0
dense_11 (Dense)	(None, 256)	131328
dropout_5 (Dropout)	(None, 256)	0
dense_12 (Dense)	(None, 10)	2570

Total params: 14,848,586
Trainable params: 4,853,514
Non-trainable params: 9,995,072

4. Data Augmentation

```
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    validation_split=0.15,
    horizontal_flip=True,
    fill_mode='nearest')
```

```
test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
```

```
class_subset = class_subset = sorted(os.listdir('/content/drive/MyDrive/celeb_data/train/'))
```

```
BATCH_SIZE = 64
traingen = train_datagen.flow_from_directory(train_path,
                                             target_size=(224, 224),
                                             class_mode='categorical',
                                             classes=class_subset,
                                             subset='training',
                                             batch_size=BATCH_SIZE,
                                             shuffle=True,
                                             seed=42)

validgen = train_datagen.flow_from_directory(train_path,
                                             target_size=(224, 224),
                                             class_mode='categorical',
                                             classes=class_subset,
                                             subset='validation',
                                             batch_size=BATCH_SIZE,
                                             shuffle=True,
                                             seed=42)

testgen = test_datagen.flow_from_directory(test_path,
                                           target_size=(224, 224),
                                           class_mode=None,
                                           classes=class_subset,
                                           batch_size=1,
                                           shuffle=False,
                                           seed=42)
```

```
Found 638 images belonging to 10 classes.
Found 110 images belonging to 10 classes.
Found 246 images belonging to 10 classes.
```

```
n_steps = traingen.samples // BATCH_SIZE
n_val_steps = validgen.samples // BATCH_SIZE
n_epochs = 100
```

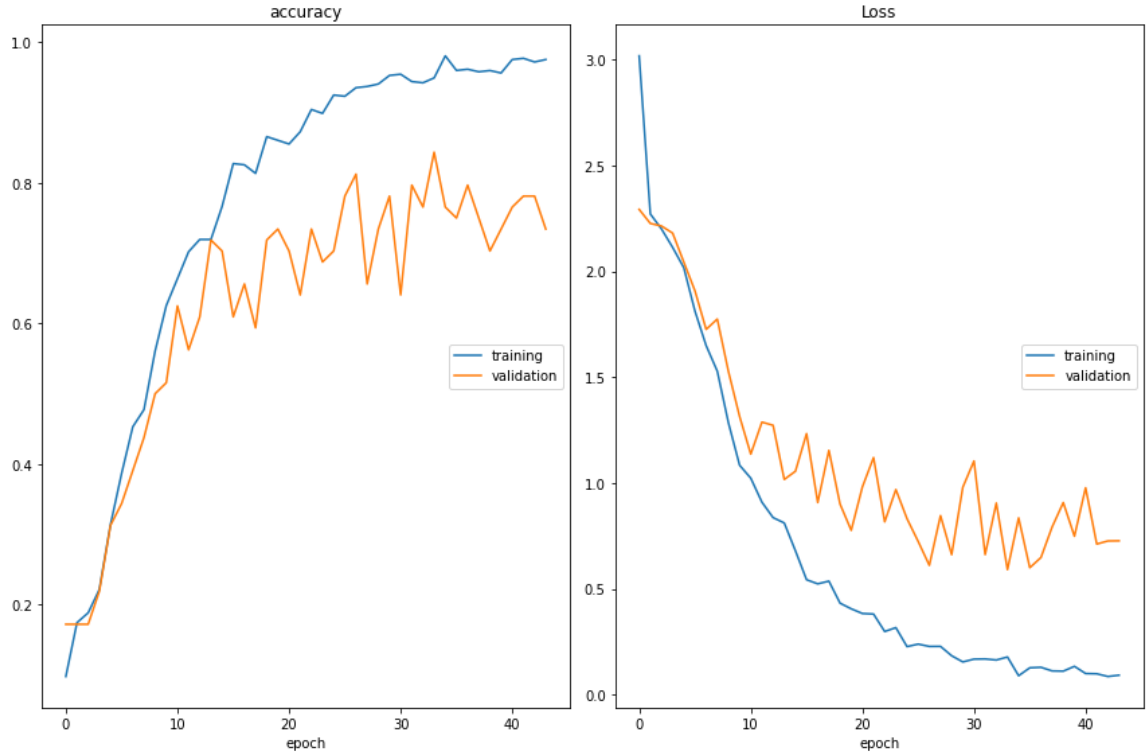
5. Livelossplot setup

```
Successfully installed livelossplot-0.5.4
```

```
from livelossplot.inputs.keras import PlotLossesCallback
from keras.callbacks import ModelCheckpoint, EarlyStopping

plot_loss_75 = PlotLossesCallback()

# ModelCheckpoint callback - save best weights
tl_checkpoint_1 = ModelCheckpoint(filepath='tl_model_v1.weights.best.hdf5',
                                  save_best_only=True,
                                  verbose=1)

# EarlyStopping
early_stop = EarlyStopping(monitor='val_loss',
                           patience=10,
                           restore_best_weights=True,
                           mode='min')
```

6. Fitting the model

```
vgg_history = model.fit(traingen,#validgen
                        batch_size=BATCH_SIZE,
                        epochs=n_epochs,
                        validation_data=validgen,
                        steps_per_epoch=n_steps,
                        validation_steps=n_val_steps,
                        callbacks=[tl_checkpoint_1, early_stop, plot_loss_75],
                        verbose=1)
```




```
accuracy
        training                (min:    0.098, max:    0.981, cur:    0.976)
        validation              (min:    0.172, max:    0.844, cur:    0.734)
Loss
        training                (min:    0.086, max:    3.017, cur:    0.092)
        validation              (min:    0.590, max:    2.292, cur:    0.727)
```

7. Generating Predictions

```
# Generate predictions
model.load_weights('tl_model_v1.weights.best.hdf5') # initialize the best trained weights

true_classes = testgen.classes
class_indices = traingen.class_indices
class_indices = dict((v,k) for k,v in class_indices.items())

vgg_preds = model.predict(testgen)
vgg_pred_classes = np.argmax(vgg_preds, axis=1)
```

8. Results

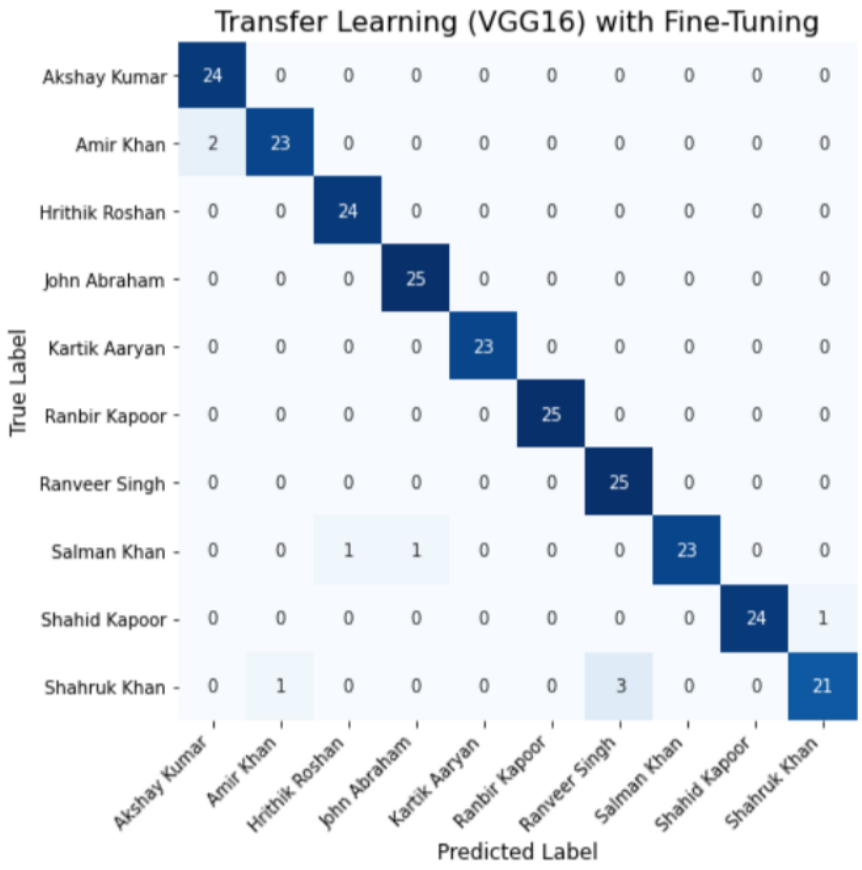
```
from sklearn.metrics import confusion_matrix , classification_report

print(classification_report(true_classes,vgg_pred_classes))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        24
           1       0.96      0.92      0.94        25
           2       0.96      1.00      0.98        24
           3       0.96      1.00      0.98        25
           4       1.00      1.00      1.00        23
           5       1.00      1.00      1.00        25
           6       0.89      1.00      0.94        25
           7       1.00      0.92      0.96        25
           8       1.00      0.96      0.98        25
           9       0.95      0.84      0.89        25

    accuracy                           0.96       246
   macro avg       0.97      0.96      0.96       246
weighted avg       0.96      0.96      0.96       246
```

```
from sklearn.metrics import accuracy_score

vgg_acc = accuracy_score(true_classes, vgg_pred_classes)
print("VGG16 Model(with fine tuning) Accuracy: {:.2f}%".format(vgg_acc * 100))
```

```
VGG16 Model(with fine tuning) Accuracy: 96.34%
```

Confusion Matrix:



9. Future scope/Suggestions:
    - Try different combination of layers to see changes in accuracy.
    - Use coloured images instead of black and white, and analysing the changes in prediction.
    - Implement this model on video inputs.

Project link:
https://github.com/Anjali0305/Face-Identification-Model