Capstone Project Report

Face Detection And Recognition

Name: Pavan Tiwari
Course: AI and ML
(Batch 4)

- **Problem Statement**

Face recognition such as that used in our phones, relies on two important steps. First step is face detection, which can be implemented using simple classifiers (such as Haar Cascade) or CNNs. Second part involves the recognition of the identity of the person, which further uses a CNN for this task. Implement a face recognition network using Haar Cascade for detection, followed by using VGG-19 for the task of recognition. Make sure you use a pre-trained VGG model and freeze the weights of starting layers before fine tuning the mode
Face detection can be regarded as a specific case of object-class detection, which focuses on the detection of frontal human faces. Once the facial region is obtained, we can use deep learning methods such as CNNs to extract a wide range of features from images. Deep neural networks can be used to produce a bunch of numbers each of which describes a face (known as face encodings) and can be used for both facial recognition and search

Prerequisites

- Software:
  - Python 3 (Use anaconda as your python distributor as well)

- Tools:
  - Pandas
  - Numpy
  - Matplotlib
  - Seaborn
  - OpenCv
  - Deep Learning

- Dataset: Custom Dataset

- **Method Used**

In this case we gonna use the deep learning model (CNN) to train our model on image and at end we gonna freeze the weights trained and then we use that trained weight for any process . For better accuracy we can use VGG19 model which has its own CNN layer inbuilt

- **Implementation:**

1. Load all required libraries

In [1]:
```
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D,Input,Dense,ZeroPadding2D,BatchNormalization,Activation,MaxPooling2D,Flatten,Dropout
from keras.models import Model, load_model
from keras.callbacks import TensorBoard,ModelCheckpoint
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

2. Data Augmentation And Reading Dataset

In [7]:
```
TRAINING_DIR = './dataset/train'
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,batch_size = 10,target_size = (150,150))
```

```
Found 100 images belonging to 2 classes.
```

In [8]:
```
VALIDATION_DIR = "./dataset/test"
validation_datagen = ImageDataGenerator(rescale=1.0/255)

validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                              batch_size=10,
                                                              target_size=(150, 150))
```

```
Found 70 images belonging to 2 classes.
```

In [9]:
```
checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
```

3. Model Building

In [12]:
```
vgg19 = VGG19(weights='imagenet',include_top=False,input_shape=(128,128,3))

for layer in vgg19.layers:
    layer.trainable = False

model = Sequential()
model.add(vgg19)
model.add(Flatten())
model.add(Dense(2,activation='sigmoid'))
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_
ls_notop.h5
80142336/80134624 [==============================] - 21s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 4, 4, 512)         20024384
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 16386
=================================================================
Total params: 20,040,770
Trainable params: 16,386
Non-trainable params: 20,024,384
```


4. Compiling Model and Training Weights

In [13]:
```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics ="accuracy")
```

In [17]:
```
history = model.fit(train_generator,
                    epochs=20,validation_data=validation_generator,
                    validation_steps=len(validation_generator)//32)
```

```
Epoch 1/20
10/10 [==============================] - 20s 2s/step - loss: 0.6698 - accuracy: 0.5600
Epoch 2/20
10/10 [==============================] - 19s 2s/step - loss: 0.4431 - accuracy: 0.7800
Epoch 3/20
10/10 [==============================] - 18s 2s/step - loss: 0.1253 - accuracy: 1.0000
Epoch 4/20
10/10 [==============================] - 20s 2s/step - loss: 0.0348 - accuracy: 1.0000
Epoch 5/20
10/10 [==============================] - 18s 2s/step - loss: 0.0187 - accuracy: 1.0000
Epoch 6/20
10/10 [==============================] - 18s 2s/step - loss: 0.0055 - accuracy: 1.0000
```

After Running through 20 Epochs we have reached to the point with good Accuracy now we will save the model weight or freeze for further use

In [18]:
```
from keras.models import load_model
model.save("model-10.h5")
```

Now we will use the opencv library and gonna use hard cascade yml file to detect face and use our model weight to predict the faces

In [1]:
```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("model-10.h5")

labels_dict={0:'Shikha',1:'Pavan'}
color_dict={0:(0,0,255),1:(0,255,0)}

size = 4
webcam = cv2.VideoCapture(0) #Use camera 0

# We load the xml file
classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

while True:
    (rval, im) = webcam.read()
    im=cv2.flip(im,1,1) #Flip to act as a mirror

    # Resize the image to speed up detection
    mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] // size))

    # detect MultiScale / faces
    faces = classifier.detectMultiScale(mini)
    # Draw rectangles around each face
    for f in faces:
        (x, y, w, h) = [v * size for v in f] #Scale the shapesize backup
        #Save just the rectangle faces in SubRecFaces
        face_img = im[y:y+h, x:x+w]
        resized=cv2.resize(face_img,(128,128))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,128,128,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)
        #print(result)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(im, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

    # Show the image
    cv2.imshow('LIVE',   im)
    key = cv2.waitKey(10)
    # if Esc key is press then break out of the loop
    if key == 27: #The Esc key
        break
# Stop video
webcam.release()
```

OUTPUT