NITTE
EDUCATION TRUST

N.M.A.M. INSTITUTE OF TECHNOLOGY

(An Autonomous Institution under VTU, Belgaum)
AICTE approved, (ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
NITTE -574 110, Udupi District, KARNATAKA.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

June 2021

FACE EMOTION AND GENDER DETECTION

A MINI PROJECT REPORT SUBMITTED BY

Steve Aston D Almeida
4NM18CS188
VI Semester, D Section

Suman Manohar Shettigar
4NM18CS193
VI Semester, D section

UNDER THE GUIDANCE OF:
Ms. Keerthana B. Chigateri
Assistant professor GD II

Department of Computer Science and Engineering

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

Bachelor of Engineering in Computer Science & Engineering
From
Visvesvaraya Technological University, Belagavi

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

CERTIFICATE

"Emotion and Gender Detection" is a bonafide work carried out by Steve Aston D Almeida (4NM18CS188) and Suman Manohar Shettigar (4NM18CS193) in partial fulfilment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belagavi during the year 2020-2021.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The Mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide Signature of HOD

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to make this project successful.

First and foremost, our sincere thanks to our beloved principal, Dr. Niranjan N. Chiplunkar for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.

We sincerely thank Dr. Jyothi Shetty, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyanthaya Memorial Institute of Technology, Nitte.

We express our deep sense of gratitude and indebtedness to our guide Ms. Keerthana B. Chigateri , Assistant Professor GD II, Department of Computer Science and Engineering, for her inspiring guidance, constant encouragement, support and suggestions for improvement during the course of our project.

We thank all the teaching and non-teaching staff members of the Computer Science and Engineering Department and our parents and friends for their honest opinions and suggestions throughout the course of our project.

Finally, we thank all those who have supported us directly or indirectly throughout the project and make it a grand success.

Steve Aston D Almeida

(4NM18CS188)

Suman Manohar Shettigar

(4NM18CS193)

ABSTRACT

A facial expression is one or more motions or positions of the muscles beneath the skin of the face. It is the visible manifestation of the affective state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in interpersonal relations. People share a common set of emotions which are shown through facial expressions.

Facial expression recognition plays a crucial role in the area of human-machine interaction. Automatic facial expression recognition systems have many applications including human behaviour understanding, detection of mental disorders, robot expression generation and synthetic human expressions .

In this project we have applied machine learning to identify the key human emotions anger, happiness, sadness, surprise and neutrality. Reasonable accuracy is achieved with the predictor, dependent on the testing set and test emotions. Recognition of facial expression by computer with high recognition rate is still a challenging task.

TABLE OF CONTENTS

Introduction

Facial expression is the visible evidence of the state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in human relations. Human facial expressions can be easily classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Our facial emotions are expressed through activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind. These emotions have no cultural barriers or language barriers and hence are universal to human beings. The emotions can be classified by computers and can be used in fields of entertainment, social media, content analysis and healthcare .

Image processing is the field of signal processing where both the input and output signals are images. The objective of this project is to develop an Automatic Facial Expression Recognition System which can take human facial images containing some expression as input and recognize and classify it into different expression classes. Several Projects have already been done in this field and our goal will not only be to develop an Automatic Facial Expression Recognition System but also detecting the gender of the person.

In this project ,we have used Convolutional Neural Networks (CNN) to detect emotions. CNN was inspired by biological processes.

Literature Survey:

Real-time Convolutional Neural Networks for Emotion and Gender Classification - *Octavio Arriaga,Paul G. Pl¨oger,Matias Valdenegro*

In this paper we propose an implementation of a general convolutional neural network (CNN) building framework for designing real-time CNNs. We validate our models by creating a real-time vision system which accomplishes the tasks of face detection, gender classification and emotion classification simultaneously in one blended step using our proposed CNN architecture. After presenting the details of the training procedure setup we proceed to evaluate on standard benchmark sets. We report accuracies of 96% in the IMDB gender dataset and 66% in the FER-2013 emotion dataset. Along with this we also introduced the very recent real-time enabled guided backpropagation visualization technique. Guided back-propagation uncovers the dynamics of the weight changes and evaluates the learned features. We argue that the careful implementation of modern CNN architectures, the use of the current regularization methods and the visualization of previously hidden features are necessary in order to reduce the gap between slow performances and real-time architectures. Our system has been validated by its deployment on a Care-O-bot 3 robot used during RoboCup@Home competitions. All our code, demos and pretrained architectures have been released under an open-source license in our public repository.

AGE, GENDER AND EMOTION DETECTION USING CNN - *Manasa S. B.; Abraham, Jeffy. S.; Sharma, Anjali; Himapoornashree K. S.*

Face is one of the most dominant features in our body. We can get a lot of information like age, gender, etc by analyzing the face of a human. In today's world, computer vision is being used to train machines to comprehend and understand the real world. Using several digital images from webcam, videos, cameras and with the help of deep learning, computers can correctly figure out and classify objects and then respond to what they "see" in the real world. There are various uses of identifying age and gender from faces like forensic testing, restricting access of alcohol from vending machines and adult content for young people. Emotion from a face can be used to predict human computer interaction, students/teacher's interest in class, advertisement bots etc.

Emotion Detection using Image Processing in Python - *Raghav Puri,Archit Gupta,Manas Sikri,Mohit Tiwarit.*

In this work, the user's emotion using its facial expressions will be detected. These expressions can be derived from the live feed via the system's camera or any pre-existing image available in the memory. Emotions possessed by humans can be recognized and has a vast scope of study in the computer vision industry upon which several researchers have already been done. The work has been implemented using Python (2.7, Open Source Computer Vision Library (OpenCV) and NumPy. The scanned image(testing dataset) is being compared to the training dataset and thus emotion is predicted. The objective of this paper is to develop a system which can analyze the image and predict the expression of the person. The study proves that this procedure is workable and produces valid results.

Design :

a. OpenCV:

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision.The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations. We use the OpenCV library to test our models with real time input of the subject.

b. Dataset Explanation:

We have used two datasets, one is fer2013 which consists of two folders namely train and validation, where each folder contains images referring to emotions, i.e. neutral,happy,fear,sad,angry, and surprised. Train folder’s images are used for training and the Validation folder’s images are used for fine tuning the model. The training dataset has 28821 images and the validation dataset has 7066 images. The other dataset is of gender based images consisting of two folders namely men and women. The men’s face folder has 1527 training images and the women’s face folder has 1379 training images.

c. CNN Architecture

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer. The architecture of the Convolution Neural Network used in the project is shown in the following figure.

Input Layer:

The input layer has predetermined, so the image must be pre-processed before it can be fed into the layer. An image of 48x48 pixels is used from the fer2013 dataset and an image of 96x96 pixels is used from the gender classification dataset. For testing purposes , the webcam is used which produces a high resolution image which must be cropped after detecting the face using OpenCV Haar Cascade Classifier and normalized .

What is Cascade Classifier?

Cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. Cascading classifiers are trained with several hundred "positive" images of a particular object and arbitrary "negative" images of the same size.

After the classifier is trained it can be applied to a region of an image and detect the object based on requirement.

The steps followed in our classifier for detecting faces is as follows :

1. Input image is passed to Cascade Classifier
2. ROI (region of interest) is extracted from the given image
3. Resize the image into 48 x 48 pixels for emotion prediction and 96 x 96 pixels for gender prediction.
4. Feed this image to the input layer .

Convolution and Pooling Layers:

The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighbourhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter

bank. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function. . Convolution and pooling is done based on batch processing. Each batch has N images. In each convolution layer, four dimensional convolution is calculated between image batch and feature maps for emotion prediction and three dimensions for gender prediction .

The purpose of the pooling layers is to reduce the spatial resolution of the feature maps. After each convolution layer the image is sampled down for dimensionality reduction. This process is called Pooling. Max pooling and Average Pooling are two famous pooling methods. In this project max pooling is done. Pool size of (2x2) is taken, which splits the image into a grid of blocks each of size 2x2 and takes a maximum of 4 pixels.

Fully Connected Layer

Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. This layer is inspired by the way neurons transmit signals through the brain. It takes a large number of input features and transforms features . The layers are connected with trainable weights. The weights of these layers are calculated by forward propagation of training data then backward propagation of its errors. Back propagation starts from evaluating the difference between prediction and true value, and back calculates the weight adjustment needed to every layer before. We can control the training speed and the complexity of the architecture by tuning the hyper-parameters, such as learning rate and network density. Hyper-parameters for this layer are learning rate, momentum, regularization parameter, and decay.

Output of the pooling layer is flattened and sent to the first hidden layer whose output is sent to the second hidden layer followed by the output layer .

Output Layer

Output from the second hidden layer is connected to the output layer having seven distinct classes. Using Softmax/Sigmoid activation function, output is obtained using the probabilities for each of the seven classes of emotion and 2 classes of gender respectively. The class with the highest probability is the predicted class.

Implementation:

A. Creating Emotion Classification Model:

a. Importing libraries

```
import matplotlib.pyplot as plt
import os

# Importing Deep Learning Libraries

from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormalization,Activation,MaxPooling2D
from keras.models import Model,Sequential
from keras.optimizers import Adam,SGD,RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
```

b. Displaying images

```
picture_size = 48
folder_path = "../input/face-expression-recognition-dataset/images/"

expression = 'disgust'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i) #3x3 matrix kind of display
    img = load_img(folder_path+"train/"+expression+"/"+
                   os.listdir(folder_path + "train/" + expression)[i], target_size=(picture_size, picture_size))
    plt.imshow(img)
plt.show()
```

c. Setting up Training data generator and Validation Data Generator

```
batch_size  = 128
#having training set and validation set
datagen_train  = ImageDataGenerator()
datagen_val = ImageDataGenerator()

train_set = datagen_train.flow_from_directory(folder_path+"train",#go inside training folder
                                              target_size = (picture_size,picture_size),#standardise the size
                                              color_mode = "grayscale",#for better accuracy
                                              batch_size=batch_size,#128
                                              class_mode='categorical',#7 different category of emotions
                                              shuffle=True)

test_set = datagen_val.flow_from_directory(folder_path+"validation",
                                              target_size = (picture_size,picture_size),
                                              color_mode = "grayscale",
                                              batch_size=batch_size,
                                              class_mode='categorical',
                                              shuffle=False)
```

d. Model Building (CNN)

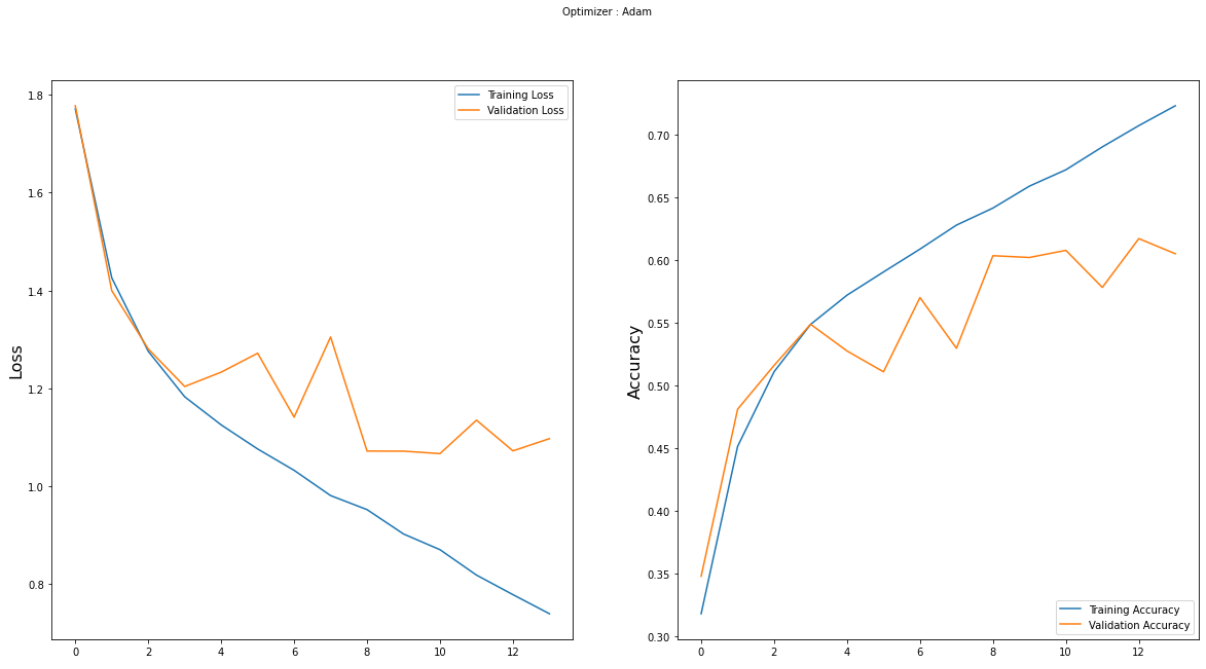
```
#Model Building
no_of_classes = 7 #7 different possible emotions
model = Sequential()#using sequential model

#1st CNN layer
model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#4th CNN layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())#2d to 1d image data because that is needed by ANN which starts next

#Fully connected 1st layer
model.add(Dense(256))#
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes, activation='softmax'))
opt = Adam(lr = 0.0001)#Adam Optimiser. It is used to update model based  on loss function. Used to minimize loss
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

e. Fitting the Model with Training and Validation Data

```
#Fitting the Model with Training and Validation Data


checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc', verbose=1, save_best_only=True, mode='max')#Checks
model and saves the model. monitors validation accuracy. verbose is how the progress is shown to user.

early_stopping = EarlyStopping(monitor='val_loss',
                           min_delta=0,
                           patience=3,
                           verbose=1,
                           restore_best_weights=True
                           )


reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                                factor=0.2,
                                patience=3,
                                verbose=1,
                                min_delta=0.0001)


callbacks_list = [early_stopping,checkpoint,reduce_learningrate]

epochs = 48

history = model.fit_generator(generator=train_set,
                              steps_per_epoch=train_set.n//train_set.batch_size,
                              epochs=epochs,
                              validation_data = test_set,
                              validation_steps = test_set.n//test_set.batch_size,
                              callbacks=callbacks_list
                              )
```

f. Plotting Accuracy And Loss

```
#Plotting Accuracy & Loss

plt.style.use('dark_background')

plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



Training Loss and Validation Loss

Training Accuracy and Validation Accuracy

Creating Gender Classification Model:

a.Importing Modules

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization, Conv2D, MaxPooling2D, Activation, Flatten, Dropout, Dense
from tensorflow.keras import backend as K
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import random
import cv2
import os
import glob
```

b. Setting up initial parameter and loading image file

```
#Initial Parameter
epochs = 100 #number of cycles
lr = 1e-3 #learning rate
batch_size = 64 #number of images sent in a batch
img_dims = (96,96,3)#image dimension , 3 is channel rgb

data = []#append all images here
labels = []#man(0) or woman(1)

# load image files from the dataset
image_files = [f for f in glob.glob(r'C:\Files\gender_dataset_face' + "/**/*", recursive=True) if not
os.path.isdir(f)]
random.shuffle(image_files)
```

c. Converting images to arrays and labelling the categories

```
#Converting images to arrays and labelling the categories
for img in image_files:

    image = cv2.imread(img)#read image
    #we need uniform size for all images
    image = cv2.resize(image, (img_dims[0],img_dims[1]))#resize as per dimensions set aboce that is 96x96
    image = img_to_array(image)#convert image to array
    data.append(image)#inside data list image array is appended

    #split breaks down path, and from back woman/man will be at -2 . this id done to know which image is man/woman
    label = img.split(os.path.sep)[-2] # C:\Files\gender_dataset_face\woman\face_1162.jpg
    if label == "woman":
        label = 1
    else:
        label = 0
    #put the image category index in labels LIST
    labels.append([label]) # [[1], [0], [0], ...]
```

d. Preprocessing of the data

```
# pre-processing
#for deep learning we need arrays, so convert image to array
data = np.array(data, dtype="float") / 255.0
#list to array
labels = np.array(labels)

# split dataset for training and validation


(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2,
                                                  random_state=42)

#this step gives two neuron in our model and easier to process
#left digit is man,right is woman . which ever is true that is 1
trainY = to_categorical(trainY, num_classes=2) # [[1, 0], [0, 1], [0, 1], ...]
testY = to_categorical(testY, num_classes=2)
```

e. Model Building(CNN)

```
# define model
def build(width, height, depth, classes):
    model = Sequential()
    inputShape = (height, width, depth)
    chanDim = -1 #normalising, to keep value in datset to common scale . above depth seems to be first argument in
some model . so to fix that

    #check channel
    if K.image_data_format() == "channels_first": #Returns a string, either 'channels_first' or 'channels_last'
        inputShape = (depth, height, width)
        chanDim = 1

    # The axis that should be normalized, after a Conv2D layer with data_format="channels_first",
    # set axis=1 in BatchNormalization.

    model.add(Conv2D(32, (3,3), padding="same", input_shape=inputShape))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))
    model.add(MaxPooling2D(pool_size=(3,3)))#basically to reduce noise
    model.add(Dropout(0.25))#to avoid overfitting. 25% of neurons are deactivated during front and back propogation

    model.add(Conv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))

    model.add(Conv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))

    model.add(Conv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))
    #2d to 1d
    model.add(Flatten())

    model.add(Dense(1024))#1024->neurons, dense is fully connected layer. this is last layers
    model.add(Activation("relu"))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))

    model.add(Dense(classes))#final output are two classes , men or women
    model.add(Activation("sigmoid"))#either use softmax or sigmoid as they are probability based functions

    return model

# build model
#img_dims is above
model = build(width=img_dims[0], height=img_dims[1], depth=img_dims[2],
                            classes=2)
```

f. Model Compiling

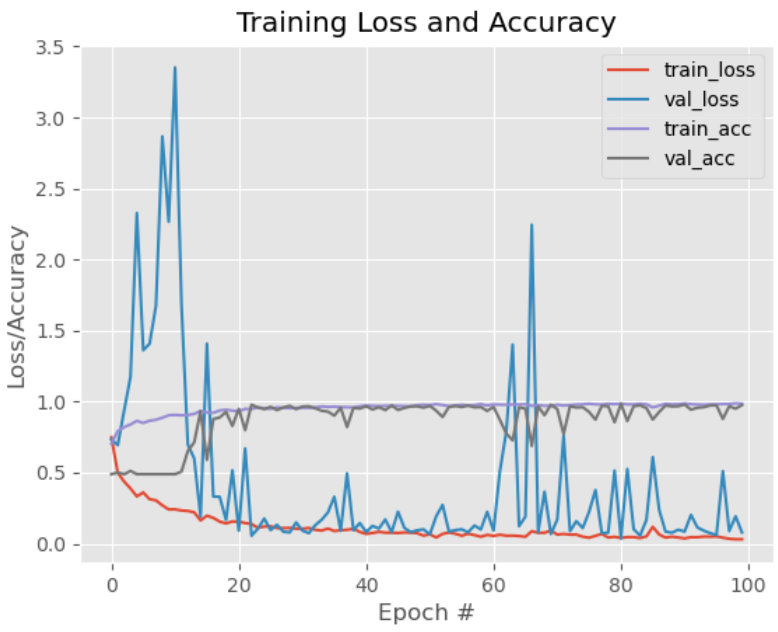
```
# compile the model

opt = Adam(lr=lr, decay=lr/epochs)

#specify the training configuration
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])
```

g. Model Fit and Save

```
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=batch_size),
                        validation_data=(testX,testY),
                        steps_per_epoch=len(trainX) // batch_size,
                        epochs=epochs, verbose=1)

model.save('gender_detection.model')
```

h. Plotting Training/Validation Loss/Accuracy

```
# plot training/validation loss/accuracy
plt.style.use("ggplot")
plt.figure()
N = epochs
plt.plot(np.arange(0,N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0,N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0,N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0,N), H.history["val_acc"], label="val_acc")

plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="upper right")

# save plot to disk
plt.savefig('plot.png')
```



Testing the models:

a.Importing Modules

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from keras.models import load_model
from keras.preprocessing.image import img_to_array
import numpy as np
import cv2
import cvlib as cv
```

b. Load model and prediction labels

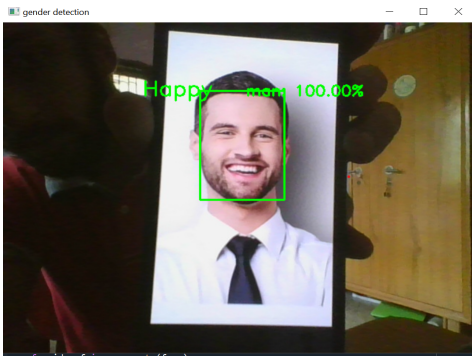
```
# load model

#load gender classification model
model = load_model('gender_detection.model')
#load emotion classification model
face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
classifier =load_model('model.h5')

#prediction labels
classes = ['man','woman']
emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sad', 'Surprise']
```

c. Using OpenCV to capture real time input for testing the models.

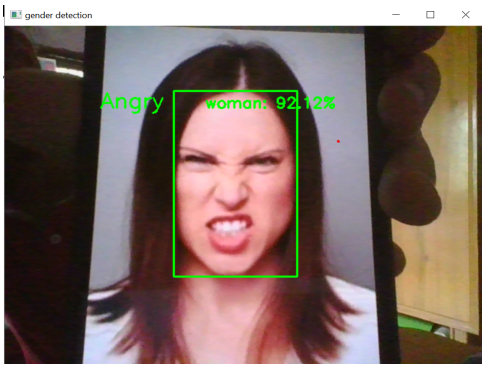
```
webcam = cv2.VideoCapture(0)
# loop through frames
while webcam.isOpened():

    # read frame from webcam
    status, frame = webcam.read() #status to check whether being recorded or not

    # apply face detection
    face, confidence = cv.detect_face(frame)#to detect face we use detect_face from cvlib module

    # loop through detected faces
    for idx, f in enumerate(face):

        #emotion empty label
        labels = []

        # get corner points of face rectangle
        (startX, startY) = f[0], f[1]
        (endX, endY) = f[2], f[3]
```

```
         # draw rectangle over face
        cv2.rectangle(frame, (startX,startY), (endX,endY), (0,255,0), 2)#2->thickness

        # crop the detected face region
        face_crop = np.copy(frame[startY:endY,startX:endX])

        if (face_crop.shape[0]) < 10 or (face_crop.shape[1]) < 10:
            continue

        #emotion test
        roi_gray = cv2.cvtColor(face_crop,cv2.COLOR_BGR2GRAY)
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)#trained model is in 48x48,so we resize.
        roi = roi_gray.astype('float')/255.0 #standardise roi
        roi = img_to_array(roi)# convert to array because model is trained on arrays
        roi = np.expand_dims(roi,axis=0)#adds an dimension for batch sizes.eg (2)->(1,2)
        prediction = classifier.predict(roi)[0]#using model, it predicts emotion in ROI
        labela=emotion_labels[prediction.argmax()]

        #Emotion End
        #gender start
        # preprocessing for gender detection model
        face_crop = cv2.resize(face_crop, (96,96))#gender model is trained in 96x96. so resize
        face_crop = face_crop.astype("float") / 255.0 #standardise face_crop
        face_crop = img_to_array(face_crop) # convert to array because model is trained on arrays
        face_crop = np.expand_dims(face_crop, axis=0) #adds an dimension for batch sizes.eg (2)->(1,2)

        # apply gender detection on face
        conf = model.predict(face_crop)[0] # model.predict return a 2D matrix, ex: [[9.9993384e-01 7.4850512e-05]]

        # get label with max accuracy
        idx = np.argmax(conf)
        label = classes[idx]#finding the prediction using index

        #store prediction of gender for display
        label = "{}: {:.2f}%".format(label, conf[idx] * 100)#conf[idx] is percentage
        #gender end

        # just coordinates to print predictions
        Y = startY - 10 if startY - 10 > 10 else startY + 10

        #write label and confidence above face rectangle
        cv2.putText(frame, label, (startX, Y),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)
        cv2.putText(frame, labela, (startX, endY+20),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)

    # display output
    cv2.imshow("gender detection", frame)

    # press "Q" to stop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# release resources
webcam.release()
cv2.destroyAllWindows()
```

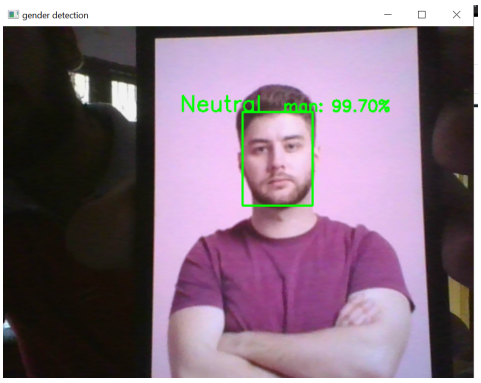
Result:



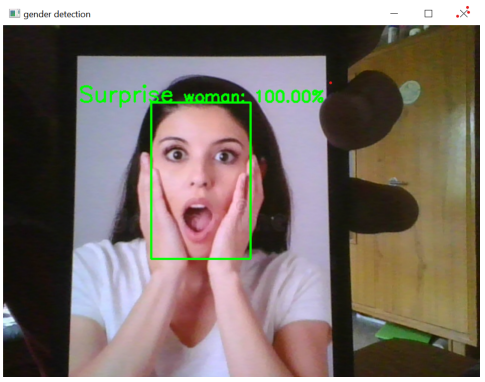
Gender: Male **Emotion**: Happy



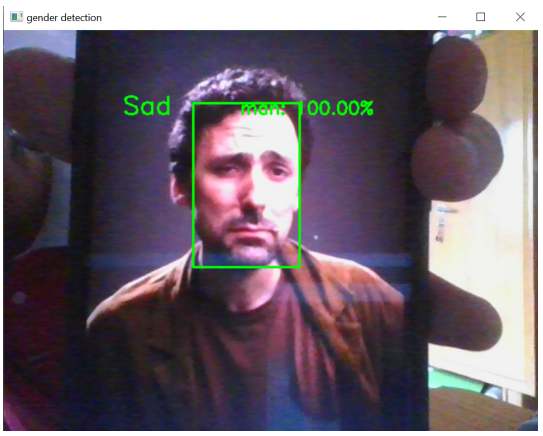
Gender: Female **Emotion**: Angry



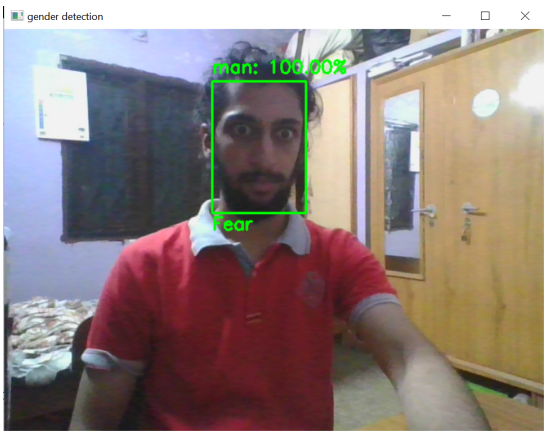
Gender: Male **Emotion**: Neutral



Gender: Female **Emotion**: Surprise



Gender: Male **Emotion**: Sad



Gender: Male **Emotion**: Fear

Conclusion:

Image processing and classification has been achieved through the use of Convolutional Neural Network Architecture and other useful libraries . The classifier is trained using images of faces. It is able to predict the emotions through processing of facial expressions i.e: happy, sad, surprise, fear, anger, disgust and neutral . It is also able to predict the gender of the person accurately.

The program is successful and predicts facial emotions that the system has been trained for. However, the system fails to detect emotions that do not clearly belong to one of the seven basic emotions. The model is able to provide an accuracy of 67% with respect to emotions as per the validation data test. The model is able to predict the gender of the person with almost 99% accuracy as per the validation data test.

The model can be improved in the future by using more training images from various datasets, investigation of more accurate algorithms that are cost efficient and considering the classification of more emotions.

References:

- www.stackoverflow.com
- Machine Learning (https://www.geeksforgeeks.org)
- https://towardsdatascience.com/emotion-detection-a-machine-learning-project-f7431f652b1f
- https://www.c-sharpcorner.com/article/real-time-emotion-detection-using-python/
- https://learnopencv.com/age-gender-classification-using-opencv-deep-learning-c-python/