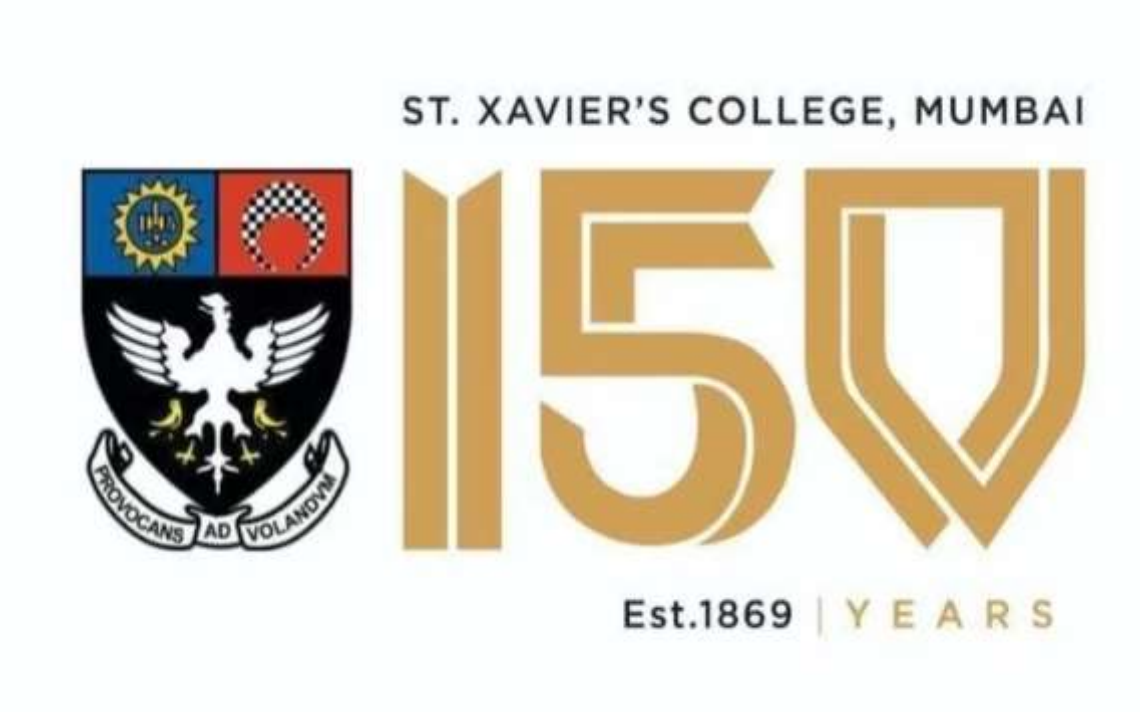
AGE AND GENDER DETECTION

IMAGE PROCESSING PROJECT REPORT

SUBMITTED BY:

Aakriti Agarwal (UID: 195254, Roll No: 32)



INTRODUCTION:

Image processing has two main steps followed by simple steps. The improvement of an image with the end goal of more good quality pictures; that can be adopted by other programs are called picture upgrades. The other procedure is the most pursued strategy utilized for the extraction of data from a picture. The division of images into certain parts is called segmentation.
To identify the age and gender of several faces' procedures, are followed by several methods. From the neural network, features are taken by the convolution network. In light of the prepared models, the image is processed into one of the age classes. The highlights are handled further and shipped off the preparation frameworks.

OBJECTIVE:

To develop a gender and age detector that can approximately guess the gender and age of the person (face) in a picture or through webcam using Adience dataset.

ABOUT THE PROJECT:

The predicted gender may be one of 'Male' and 'Female', and the predicted age may be one of the following ranges- (0 – 2), (4 – 6), (8 – 12), (15 – 20), (25 – 32), (38 – 43), (48 – 53), (60 – 100) (8 nodes in the final softmax layer). It is very difficult to accurately guess an exact age from a single image because of factors like makeup, lighting, obstructions, and facial expressions. And so, we make this a classification problem instead of making it one of regression.
Once you start treating age prediction as a regression problem, it becomes significantly harder for a model to accurately predict a single value representing that person's image.
However, if you treat it as a classification problem, defining buckets/age brackets for the model, our age predictor model becomes easier to train, often yielding substantially higher accuracy than regression-based prediction alone.
The process of visual age prediction is difficult, and we'd consider it subjective when either a computer or a person tries to guess someone's age.
In order to evaluate an age detector, you cannot rely on the person's actual age. Instead, you need to measure the accuracy between the predicted age and the perceived age

DATASET:

Adience dataset, this dataset serves as a benchmark for face photos and is inclusive of various real-world imaging conditions like noise, lighting, pose, and appearance. The images have been collected from Flickr albums and distributed under the Creative Commons (CC) license. It has a total of 26,580 photos of 2,284 subjects in eight age ranges (as mentioned above) and is about 1GB in size.

FEATURES:

- Detecting Gender and Age of face in Image Use Command:

python detect.py --image <image_name>

- Detecting Gender and Age of face through webcam Use Command:

python detect.py

LIBRARIES USED:

- OpenCV: OpenCV is a library of programming functions mainly aimed at real-time computer vision.

pip install opencv-python

- Argparse: We use the argparse library to create an argument parser so we can get the image argument from the command prompt. We make it parse the argument holding the path to the image to classify gender and age for.

pip install argparse

CONTENTS:

- opencv_face_detector.pbtxt
- opencv_face_detector_uint8.pb
- age_deploy.prototxt
- age_net.caffemodel
- gender_deploy.prototxt
- gender_net.caffemodel
- a few pictures to try the project on
- detect.py

For face detection, we have a .pb file- this is a protobuf file (protocol buffer); it holds the graph definition and the trained weights of the model. We can use this to run the trained model. And while a .pb file holds the protobuf in binary format, one with the .pbtxt extension holds it in text format. These are TensorFlow files. For age and gender, the .prototxt files describe the network configuration and the .caffemodel file defines the internal states of the parameters of the layers.

LIMITATIONS:

Our current age and gender detection system faces difficulty to accurately guess an exact age from a single image because of factors like:

- Make-up
- Lighting
- Obstructions
- Facial expressions

FUTURE SCOPE:

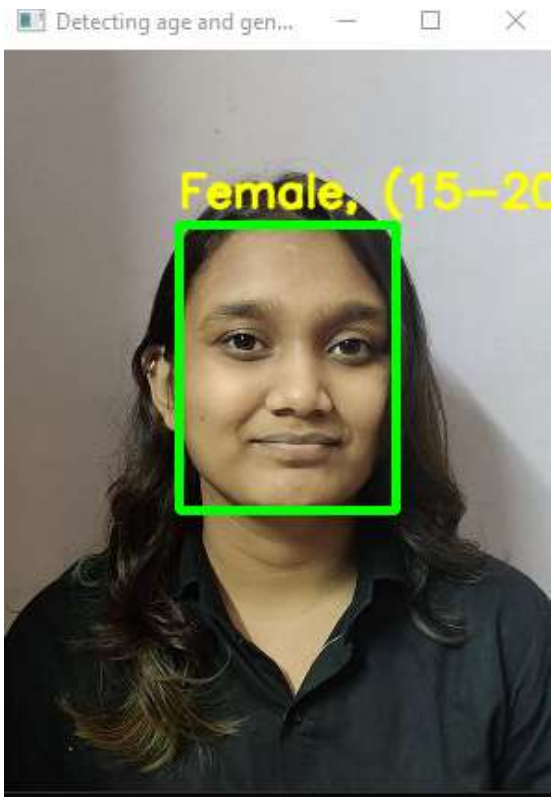
Potential uses of this application include uses in dating sites and social media, access to age-restricted content and for checkout at grocery stores when buying alcohol at the self-checkout.

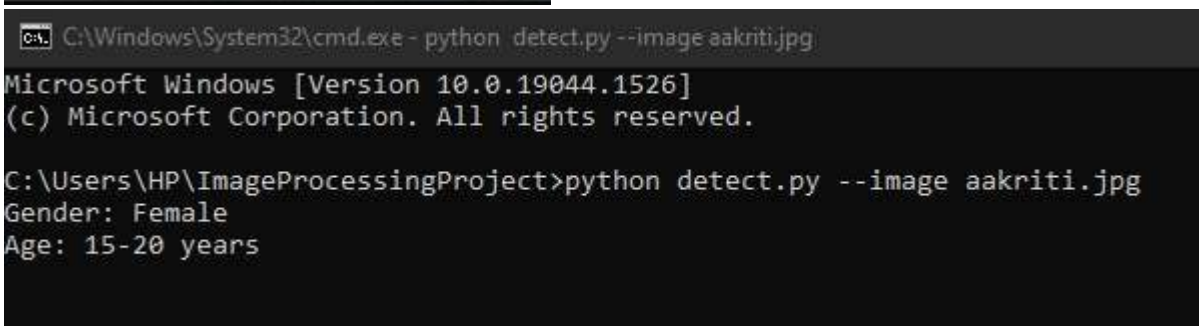
CONCLUSION:

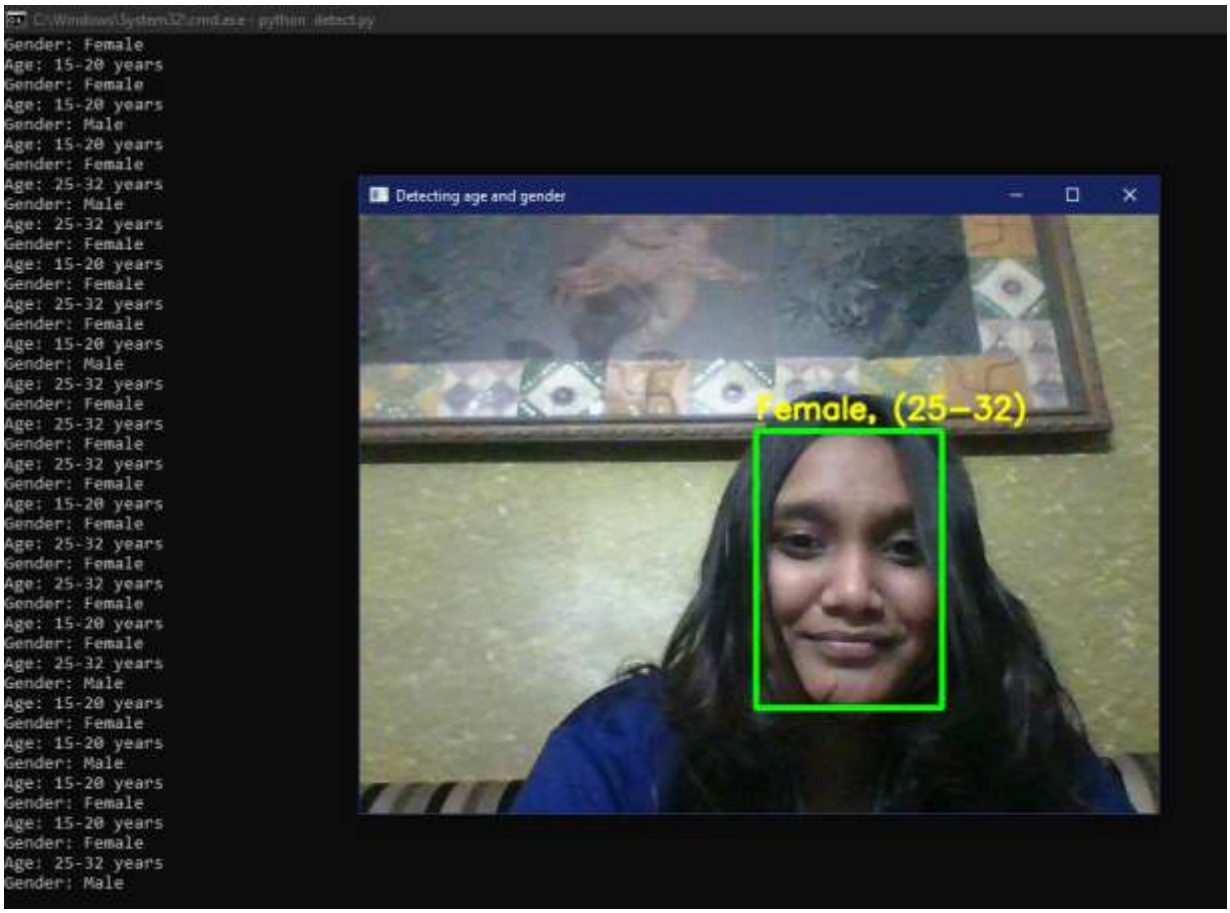
In this python project we implemented a Convolutional Neural Network (CNN) to detect gender and age from a single picture of a person. A Convolutional Neural Network is a deep neural network (DNN) widely used for the purposes of image recognition and processing.
Also known as a ConvNet, a CNN has input and output layers, and multiple hidden layers, many of which are convolutional. In a way, CNNs are regularized multilayer perceptrons.
Two important conclusions can be made from our results. First, CNN can be used to provide improved age and gen- der classification results, even considering the much smaller size of

contemporary unconstrained image sets labeled for age and gender. Second, the simplicity of our model im- plies that more elaborate systems using more training data may well be capable of substantially improving results be- yond those reported here.

SCREENSHOTS:







CODE:

```
#A Gender and Age Detection program
#Import all the required libraries.
import cv2
import math
import argparse

#highlightFace() function that is used to get the coordinates of the face
def highlightFace(net, frame, conf_threshold=0.7):
#shallow copy of the frame
frameOpencvDnn=frame.copy()
#height detection
frameHeight=frameOpencvDnn.shape[0]
#weight detection
frameWidth=frameOpencvDnn.shape[1]
#blob construction
        blob=cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123],
True, False)
```

```
 #pass the blob through the network to obtain the face detections
 net.setInput(blob)
 detections=net.forward()
 faceBoxes=[]
 #loop over the detections and extract the face coordinates
 for i in range(detections.shape[2]):
        confidence=detections[0,0,i,2]
        if confidence>conf_threshold:
        x1=int(detections[0,0,i,3]*frameWidth)
        y1=int(detections[0,0,i,4]*frameHeight)
        x2=int(detections[0,0,i,5]*frameWidth)
        y2=int(detections[0,0,i,6]*frameHeight)
        faceBoxes.append([x1,y1,x2,y2])

 # use these coordinates to create a rectangle around the face
                cv2.rectangle(frameOpencvDnn, (x1,y1), (x2,y2), (0,255,0),
int(round(frameHeight/150)), 8)
 return frameOpencvDnn,faceBoxes


 parser=argparse.ArgumentParser()

 #command for executing
 parser.add_argument('--image')

 args=parser.parse_args()

 #dataset and model creation of different datasets
 faceProto="opencv_face_detector.pbtxt"
 faceModel="opencv_face_detector_uint8.pb"
 ageProto="age_deploy.prototxt"
 ageModel="age_net.caffemodel"
 genderProto="gender_deploy.prototxt"
 genderModel="gender_net.caffemodel"

 MODEL_MEAN_VALUES=(78.4263377603, 87.7689143744, 114.895847746)

 #list of data for age classification
 ageList=['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)',
'(60-100)']

 #list of data for gender detection
 genderList=['Male','Female']

 # Load the face detection, age detection, and gender detection models
 faceNet=cv2.dnn.readNet(faceModel,faceProto)
 ageNet=cv2.dnn.readNet(ageModel,ageProto)
 genderNet=cv2.dnn.readNet(genderModel,genderProto)

 video=cv2.VideoCapture(args.image if args.image else 0)
 padding=20
```

```
 while cv2.waitKey(1)<0 :
 #call the highlightFace() function with the faceNet and frame parameters.
 #And what these returns will be stored in resultimg and faceboxes variables
 hasFrame,frame=video.read()
 if not hasFrame:
        cv2.waitKey()
        break

 resultImg,faceBoxes=highlightFace(faceNet,frame)
 if not faceBoxes:
        print("No face detected")

 for faceBox in faceBoxes:
        face=frame[max(0,faceBox[1]-padding):
               min(faceBox[3]+padding,frame.shape[0]-1),max(0,faceBox[0]-padding)
               :min(faceBox[2]+padding, frame.shape[1]-1)]

 #we create a 4-dimensional blob from the image. In doing this,
 we scale it, resize it, and pass in the mean values.
        blob=cv2.dnn.blobFromImage(face, 1.0, (227,227), MODEL_MEAN_VALUES,
swapRB=False)

 #pass the blob through the gender model
        genderNet.setInput(blob)
        genderPreds=genderNet.forward()
        gender=genderList[genderPreds[0].argmax()]
        print(f'Gender: {gender}')

 #pass the blob through the age model
        ageNet.setInput(blob)
        agePreds=ageNet.forward()
        age=ageList[agePreds[0].argmax()]
        print(f'Age: {age[1:-1]} years')

        #add the gender and age to the resulting image using cv2.putText()
function and display it with imshow().
        cv2.putText(resultImg, f'{gender}, {age}', (faceBox[0], faceBox[1]-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 2, cv2.LINE_AA)
        cv2.imshow("Detecting age and gender", resultImg)
```