Savitribai Phule Pune University

Pune Vidyarthi Griha's College of Engineering and Technology and G K Pate (Wani) Institute of Management, Pune

Department of Computer Engineering
Academic Year 2022 – 2023

"High Performance Computing"
(2019 Course)

Project Title

"Gender and Age Detection"

TEAM MEMEBRS

SR.NO	NAME	ROLL NO:
1.	AYUSH BOLLA	77
2.	DEEP PAWAR	79
3.	PRANIT RATHOD	13
4.	VISHAKA MATKAR	07

❖ **TITLE:**

Implement Gender and Age Detection: predict if a person is a male or female and also their age

❖ **THEORY:**

Age, Gender and race detection using Convolutional Neural Network Automatic age and gender classification has become relevant to an increasing amount of applications, particularly since the rise of social platforms and social media. Nevertheless, performance of existing methods on real-world images is still significantly lacking, especially when compared to the tremendous leaps in performance recently reported for the related task of face recognition. In this project we showed that by learning representations through the use of deep convolutional neural networks (CNN), a significant increase in performance can be obtained on these tasks. To this end, we used a simple convolutional net architecture that can be used even when the amount of learning data is limited.

➢ WHAT IS COMPUTER VISION?

Computer Vision is the field of study that enables computers to see and identify digital images and videos as a human would. The challenges it faces largely follow from the limited understanding of biological vision. Computer Vision involves acquiring, processing, analyzing, and understanding digital images to extract high-dimensional data from the real world in order to generate symbolic or numerical information which can then be used to make decisions. The process often includes practices like object recognition, video tracking, motion estimation, and image restoration.

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand. Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

➢ OPENCV:

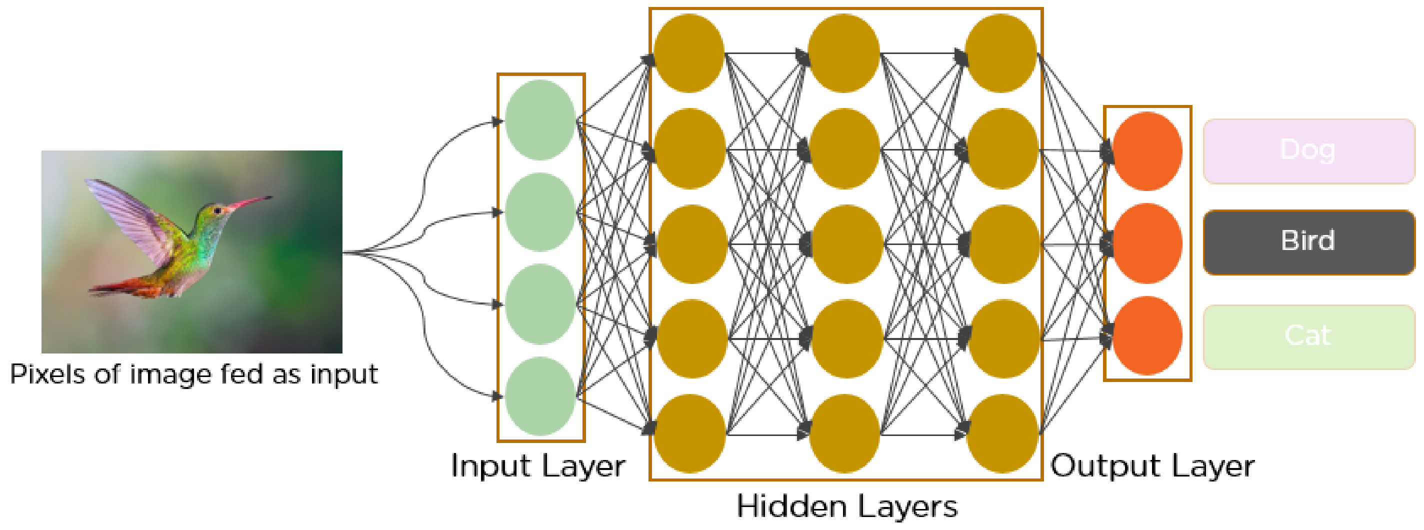
OpenCV is short for Open Source Computer Vision. Intuitively by the name, it is an open-source Computer Vision and Machine Learning library. This library is capable of processing real-time image and video while also boasting analytical capabilities. It supports the Deep Learning frameworks TensorFlow, Caffe, and PyTorch. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

➢ WHAT IS A CNN?

Convolutional neural network (CNN or convnet) is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision (CV) tasks and for applications where object recognition is vital, such as self-driving cars and facial recognition.

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.



- **CNN LAYERS:**

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer. The convolutional layer is the first layer while the FC layer is the last. From the convolutional layer to the FC layer, the complexity of the CNN increases. It is this increasing complexity that allows the CNN to successively identify larger portions and more complex features of an image until it finally identifies the object in its entirety.

1. CONVOLUTIONAL LAYER:

The majority of computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a kernel or filter inside this layer moving across the receptive fields of the image, checking if a feature is present in the image. Over multiple iterations, the kernel sweeps over the entire image. After each iteration a dot product is calculated between the input pixels and the filter. The final output from the series of dots is known as a feature map or convolved feature.

2. **POOLING LAYER:**

Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image. But unlike the convolutional layer, the pooling layer reduces the number of parameters in the input and also results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN.

3. FULLY CONNECTED LAYER:

The FC layer is where image classification happens in the CNN based on the features extracted in the previous layers. Here, fully connected means that all the inputs or nodes from one layer are connected to every activation unit or node of the next layer.

All the layers in the CNN are not fully connected because it would result in an unnecessarily dense network. It also would increase losses and affect the output quality, and it would be computationally expensive.

- **HOW DO CONVOLUTIONAL NEURAL NETWORKS WORK?**

A CNN can have multiple layers, each of which learns to detect the different features of an input image. A filter or kernel is applied to each image to produce an output that gets progressively better and more detailed after each layer. In the lower layers, the filters can start as simple features.

At each successive layer, the filters increase in complexity to check and identify features that uniquely represent the input object. Thus, the output of each convolved image -- the partially recognized image after each layer -- becomes the input for the next layer. In the last layer, which is an FC layer, the CNN recognizes the image or the object it represents.

With convolution, the input image goes through a set of these filters. As each filter activates certain features from the image, it does its work and passes on its output to the filter in the next layer. Each layer learns to identify different features and the operations end up being repeated for dozens, hundreds or even thousands of layers. Finally, all the image data progressing through the CNN's multiple layers allow the CNN to identify the entire object.

➢ STEPS TO IMPLEMENT GENDER AND AGE DETECTION:

1. 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. 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.

2. For face, age, and gender, initialize protocol buffer and model.

3. Initialize the mean values for the model and the lists of age ranges and genders to classify from.

4. Now, use the readNet() method to load the networks. The first parameter holds trained weights and the second carries network configuration.

5. Let’s capture video stream in case you’d like to classify on a webcam’s stream. Set padding to 20.

6. Now until any key is pressed, we read the stream and store the content into the names hasFrame and frame. If it isn’t a video, it must wait, and so we call up waitKey() from cv2, then break.

7. Let’s make a call to the highlightFace() function with the faceNet and frame parameters, and what this returns, we will store in the names resultImg and faceBoxes. And if we got 0 faceBoxes, it means there was no face to detect. Here, net is faceNet- this model is the DNN Face Detector and holds only about 2.7MB on disk.

8. Create a shallow copy of frame and get its height and width.

   Create a blob from the shallow copy.

   Set the input and make a forward pass to the network. faceBoxes is an empty list now. for each value in 0 to 127, define the confidence (between 0 and Wherever we find the confidence greater than the confidence threshold, which is 0.7, we get the x1, y1, x2, and y2 coordinates and append a list of those to faceBoxes. Then, we put up rectangles on the image for each such list of coordinates and return two things: the shallow copy and the list of faceBoxes.

9. But if there are indeed faceBoxes, for each of these, we define the face, create a 4-dimensional blob from the image. In doing this, we scale it, resize it, and pass in the mean values.

10. We feed the input and give the network a forward pass to get the confidence of the two class. Whichever is higher, that is the gender of the person in the picture.

11. Then, we do the same thing for age.

12. We'll add the gender and age texts to the resulting image and display it with imshow().

- **IMPLEMENTATION:**

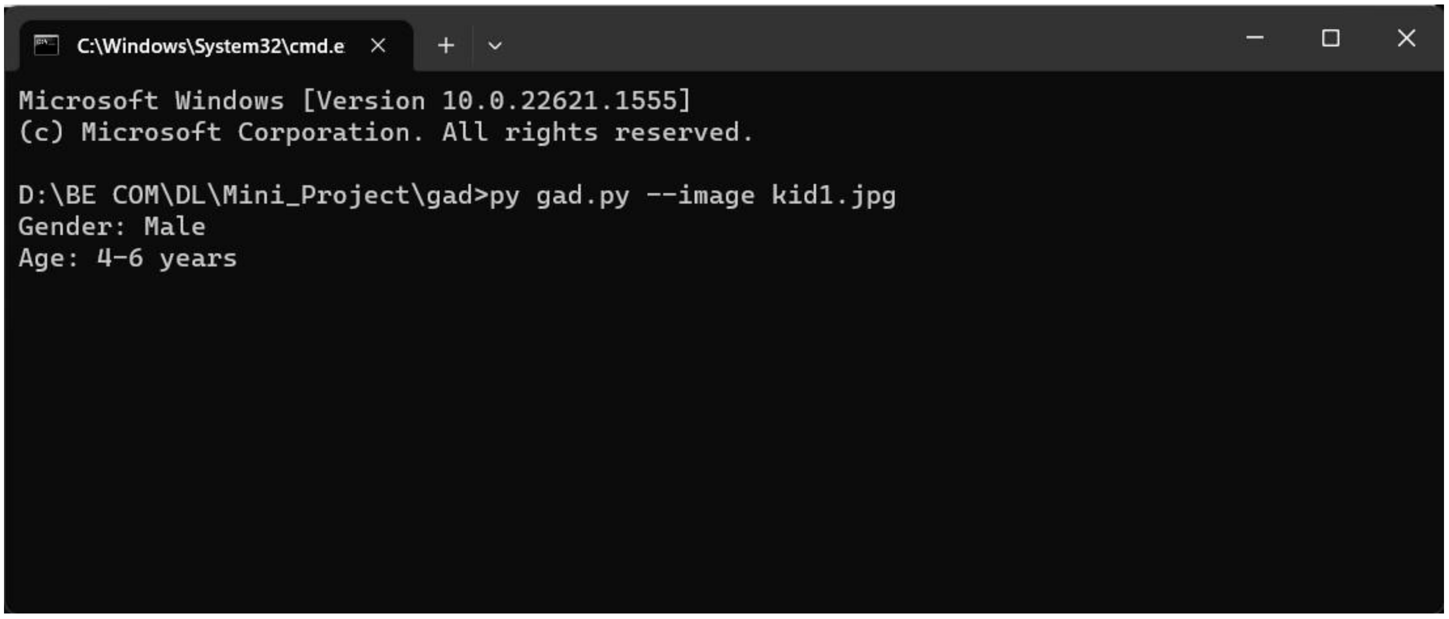
- **CODE:**

```
import cv2
import math
import argparse


def highlightFace(net, frame, conf_threshold=0.7):
        frameOpencvDnn=frame.copy()
        frameHeight=frameOpencvDnn.shape[0]
        frameWidth=frameOpencvDnn.shape[1]
        blob=cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True,
False)
        net.setInput(blob)
        detections=net.forward()
        faceBoxes=[]
        for i in range(detections.shape[2]):
                confidence=detections[0,0,i,2]
```

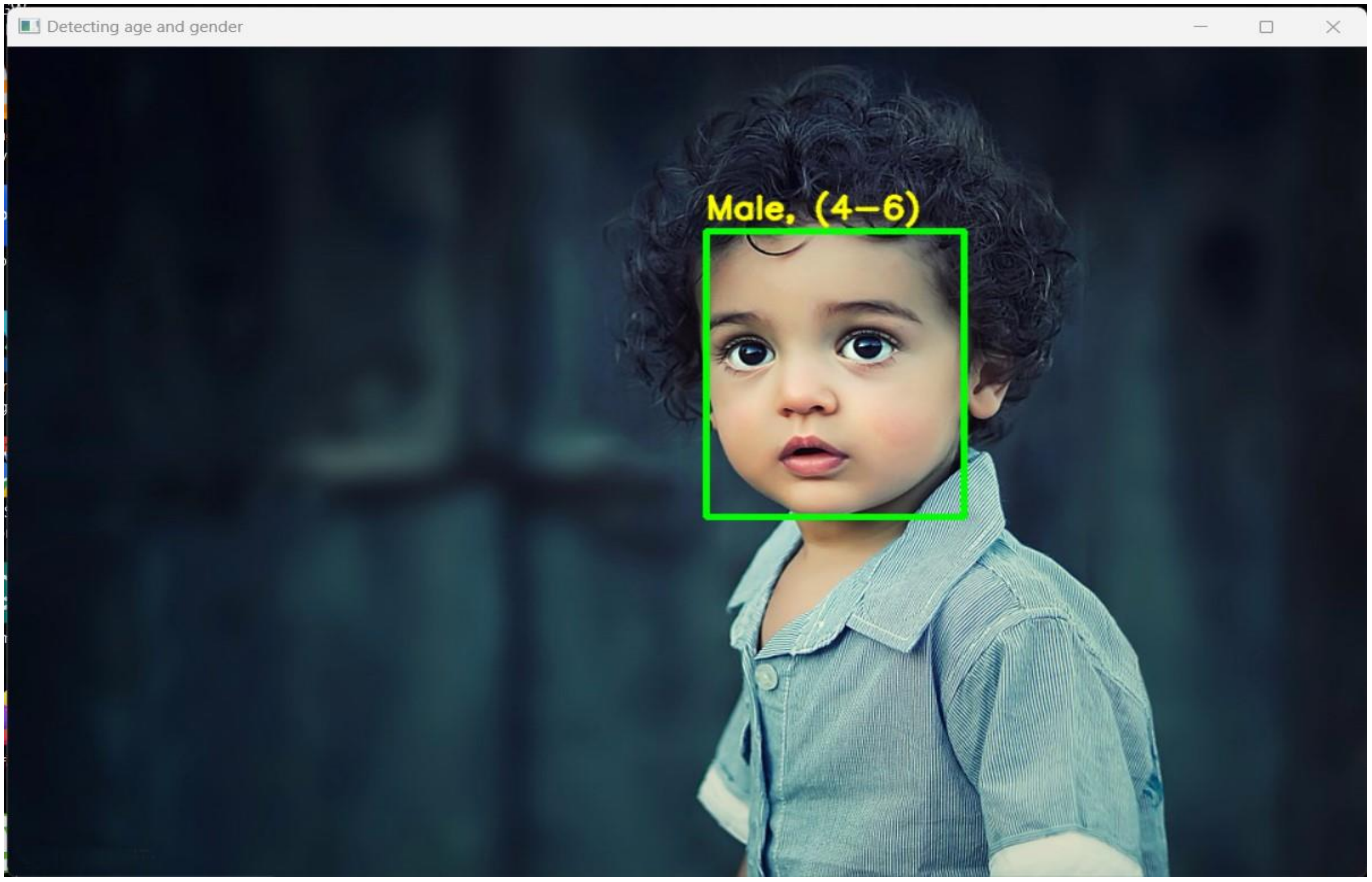
```
            if confidence>conf_threshold:
                x1=int(detections[0,0,i,3]*frameWidth)
                y1=int(detections[0,0,i,4]*frameHeight)
                x2=int(detections[0,0,i,5]*frameWidth)
                y2=int(detections[0,0,i,6]*frameHeight)
                faceBoxes.append([x1,y1,x2,y2])
                cv2.rectangle(frameOpencvDnn, (x1,y1), (x2,y2), (0,255,0),
int(round(frameHeight/150)), 8)
    return frameOpencvDnn,faceBoxes


parser=argparse.ArgumentParser()
parser.add_argument('--image')

args=parser.parse_args()

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)
ageList=['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
genderList=['Male','Female']

faceNet=cv2.dnn.readNet(faceModel,faceProto)
ageNet=cv2.dnn.readNet(ageModel,ageProto)
genderNet=cv2.dnn.readNet(genderModel,genderProto)
```

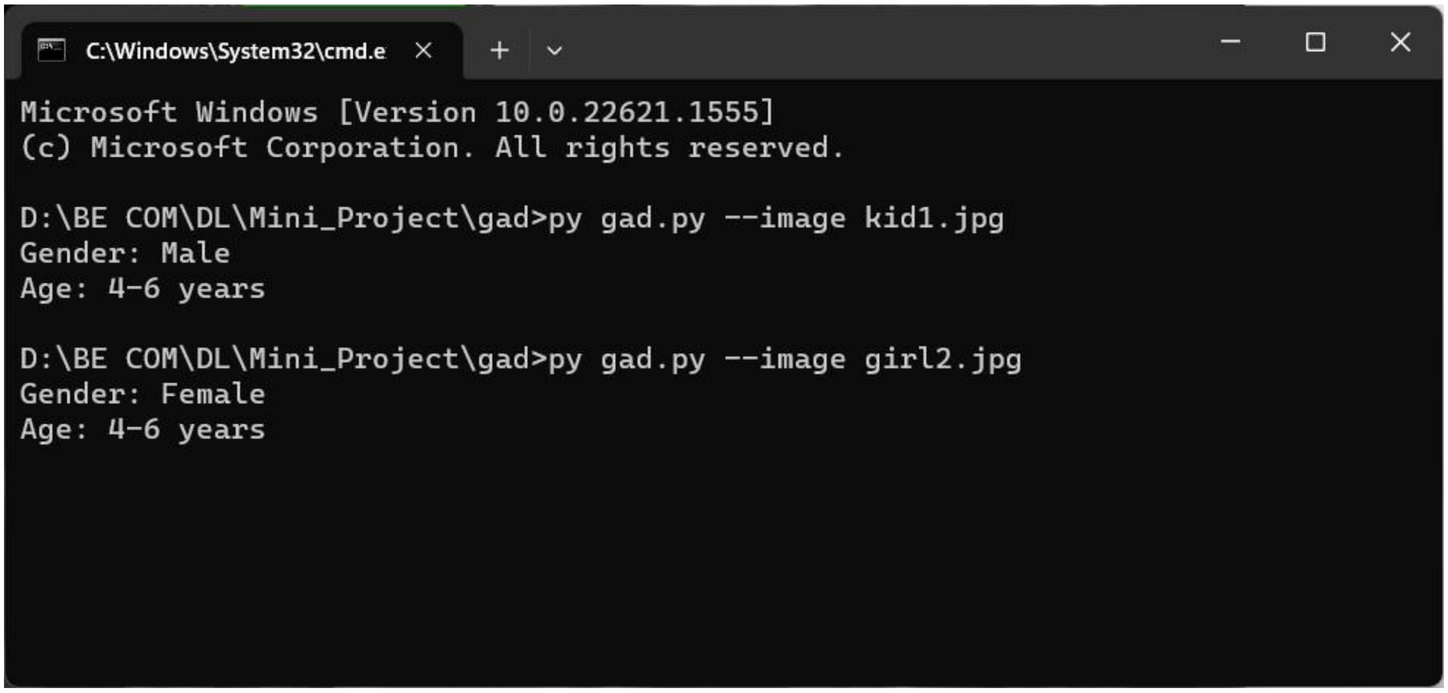
```
video=cv2.VideoCapture(args.image if args.image else 0)
padding=20
while cv2.waitKey(1)<0:
        hasFrame,frame=video.read()
        if not hasFrame:
                cv2.waitKey()
                break

        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)]

                blob=cv2.dnn.blobFromImage(face, 1.0, (227,227), MODEL_MEAN_VALUES,
swapRB=False)
                genderNet.setInput(blob)
                genderPreds=genderNet.forward()
                gender=genderList[genderPreds[0].argmax()]
                print(f'Gender: {gender}')

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

                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)
```

- **OUTPUT:**

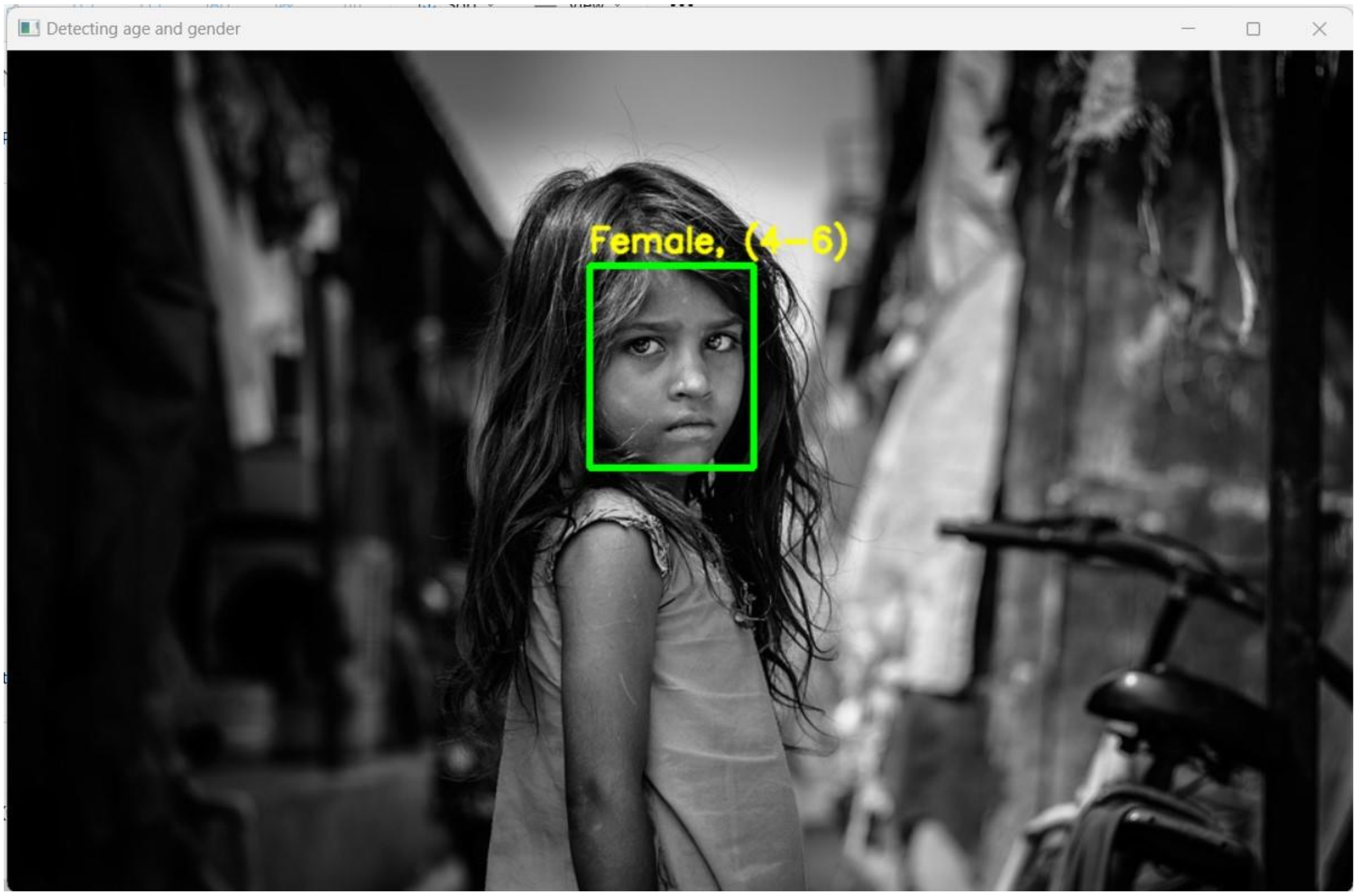


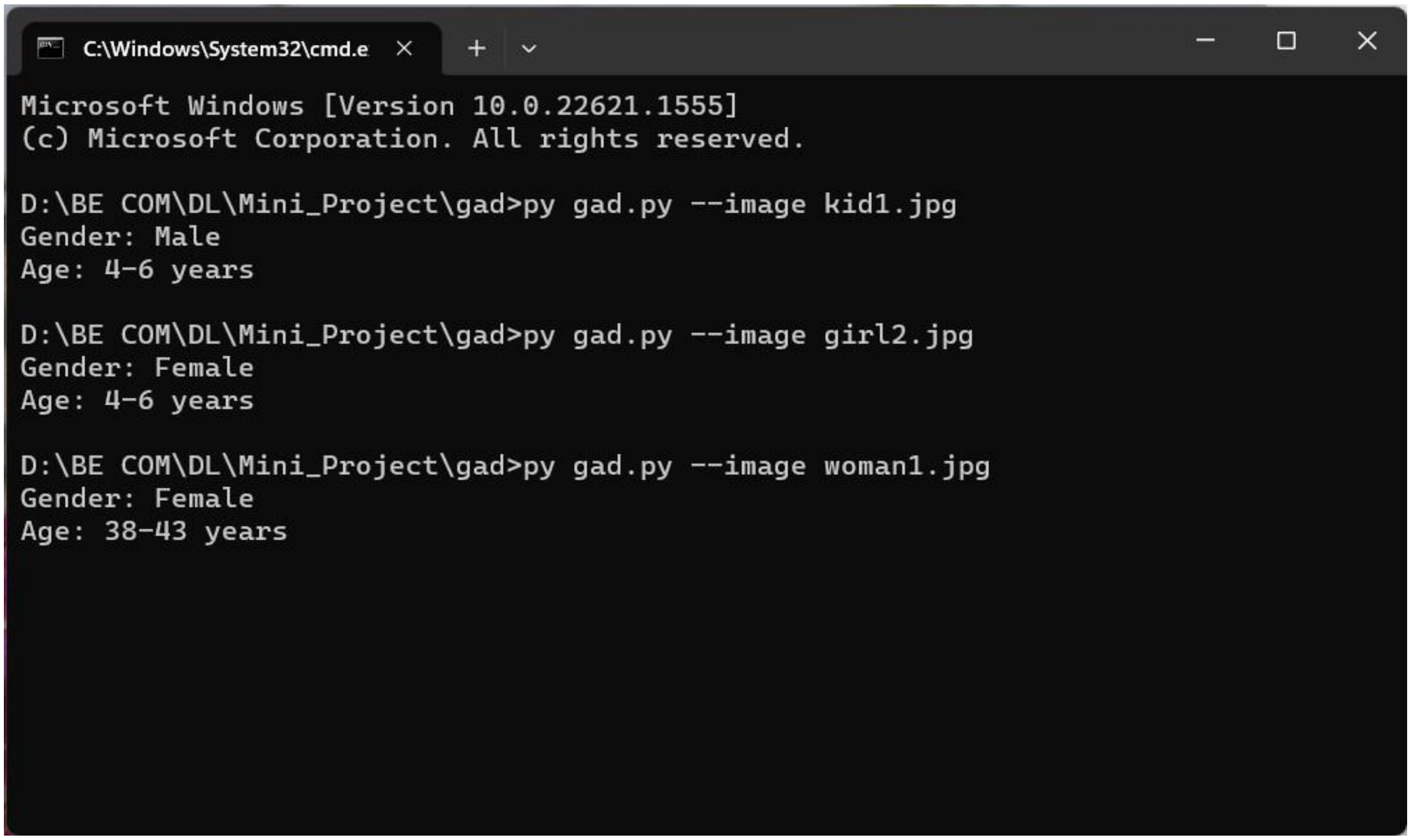
```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

D:\BE COM\DL\Mini_Project\gad>py gad.py --image kid1.jpg
Gender: Male
Age: 4-6 years
```



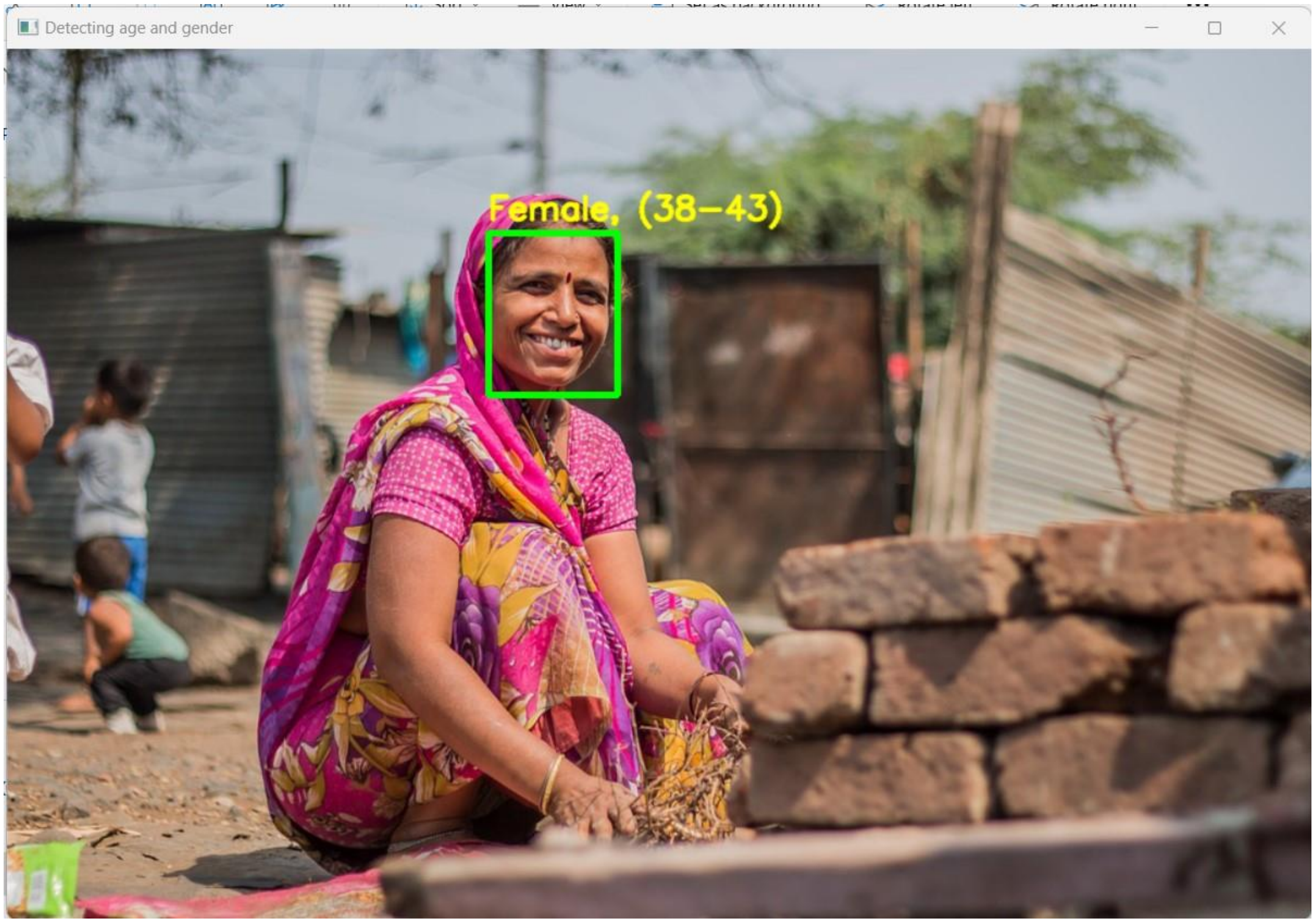


```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

D:\BE COM\DL\Mini_Project\gad>py gad.py --image kid1.jpg
Gender: Male
Age: 4-6 years

D:\BE COM\DL\Mini_Project\gad>py gad.py --image girl2.jpg
Gender: Female
Age: 4-6 years
```





```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

D:\BE COM\DL\Mini_Project\gad>py gad.py --image kid1.jpg
Gender: Male
Age: 4-6 years

D:\BE COM\DL\Mini_Project\gad>py gad.py --image girl2.jpg
Gender: Female
Age: 4-6 years

D:\BE COM\DL\Mini_Project\gad>py gad.py --image woman1.jpg
Gender: Female
Age: 38-43 years
```



➢ CONCLUSION:

To conclude this project, we have successfully implemented a CNN to detect gender and age from a single picture of a face. Age and gender, two of the key facial attributes, play a very foundational role in social interactions, making age and gender estimation from a single face image an important task in intelligent applications, such as access control, human-computer interaction, law enforcement, marketing intelligence and visual surveillance, etc.

For a wide range of applications, age and gender are critical factors. The scientific community has been more interested in estimating age and gender through facial photographs. This project offers a age and gender detection from a facial image. In this context, the system presents a system which automatically captures the face and classifies the age and gender of an individual without any physical communication. Based on the classification results, age and gender is given to the users. Experiments reveal that the proposed system's age and gender recognition approaches exceed existing methods on the basis of accuracy and computational efficiency. In future, it is planned to develop group recommendation system for a group of users in public places.

➢ REFERENCES:

1. X. Wang, L. Liang, Z. Wang, and S. Hu, "Age estimation by facial images: a survey," China Journal of Image and Graphics, vol. 17, no. 6, pp. 603–622, 2012.

2. G. W. Cottrell and J. Metcalfe, "Face emotion and gender recognition using holons," Proceedings of Conference on Advances in Neural Information Processing Systems, vol. 3, pp. 564–571, 1990.

3. W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition," ACM Computing Surveys, vol. 35, no. 4, pp. 399–458, 2003.

4. S. Baluja and H. Rowley, "Boosting sex identification performance," International Journal of Computer Vision, vol. 71, no. 1, pp. 111–119, 2007.