CSE 3017
Computer Vision
J Component Report

A project report titled

People Counter using OpenCV

By

19BLC1065 Yashi Srivastava
19BLC1118 Manav Pruthi
19BLC1137 Saksham Mishra
19BLC1146 Kaushal Goyal

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMPUTER ENGINEERING

Submitted to

Dr. R. Menaka

April 2022

School of Electronics Engineering

DECLARATION BY THE CANDIDATE

We hereby declare that the Report entitled "**People Counter using OpenCV"** submitted by me to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. R. Menaka, Professor, SENSE, VIT Chennai.**

Signature of the Candidate

Yashi Srivastava

Manav Pruthi

Saksham Mishra

Kaushal Goyal

Chennai

25/04/2022.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Menaka,** School of Electronics Engineering for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A,** Dean of the School of Electronics Engineering (SENSE), VIT University Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our **Program Chair Dr. Thiripurasundari D (for B.Tech-ECM)** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses till date.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

BONAFIDE CERTIFICATE

Certified that this project report entitled "People Counter using OpenCV" is a bonafide work of **Yashi Srivastava (19BLC1065)** , **Manav Pruthi (19BLC1118)** , **Saksham Mishra (19BLC1137) and Kaushal Goyal (19BLC1146)** carried out the "J"-Project work under my supervision and guidance for CSE3017 Computer Vision.

Dr.R.Menaka

School of Electronics Engineering

VIT University, Chennai

Chennai – 600 127.

TABLE OF CONTENTS

ABSTRACT

People counting has a wide range of applications in the context of pervasive systems. These applications range from efficient allocation of resources in smart buildings to handling emergency situations. There exist several vision based algorithms for people counting. Each algorithm performs differently in terms of efficiency, flexibility and accuracy for different indoor scenarios

One of the biggest challenges in the retail industry is measuring how many customers enter a store throughout the day. In this pandemic situation too, maintaining track of people movements and social distancing rules in a commercial setting is crucial in the current circumstances.

Hence, evaluating these algorithms with respect to different application scenarios, environment conditions and camera orientations will provide a better choice for actual deployment. The performance of these algorithms under different scenarios demonstrates the need for more accurate and faster people counting algorithms.

CHAPTER – 1

INTRODUCTION

People counting is a spatio-temporal function of human sensing, which gives the count of people in a particular area. Counting people is a useful task, which helps in understanding the flow of people in various places. The knowledge of density of people over an area would be helpful in handling emergency situations, efficient allocation of resources in the smart buildings etc. The constant movement of people, different age groups and body types makes people counting a challenging process. In addition, the presence of obstacles in indoor spaces etc., and varying lighting conditions make the process of accurately estimating the number of people in an area at given time very difficult. Real-time crowd counting in videos becomes more and more important for public area monitoring for the purpose of safety and security. The goal of crowd counting is to estimate the number of people passing through a given line or a given area. It has many valuable real-world applications, such as controlling the number of people in the venues, estimating the people flow in the subway station, counting people entering and exiting. There are still many challenges to be solved in this task. First, in crowded scenes, the occlusion between people is serious. Second, the resolution of video in surveillance camera is relatively low, detailed information is lost. In real-word places, such as in subway stations and libraries, we find that most monitors are above the front of people's heads. This is because at this position, monitors can capture faces, dresses and other characteristics of pedestrians passing through. So in our scenario, we assume that the cameras are installed at a high place, facing the crowd flow direction.

In this project we will create a system where we use a webcam or we can give our video or images to count the number of people in a frame and generate graphs, and a crowd report summarising the detections obtained using the most efficient model. An interactive graphic user interface will facilitate this application.

CHAPTER – 2

LITERATURE SURVEY

Object Detection: There are large data related to object detection, some of the popular object detection methods uses Selective Search, sliding windows in Edge Box and CPMC. Detecting an object mainly contains two things, the first is to locate the object and the second is to classify the object in a different class. Previous algorithms are mainly focused on face detection. Afterward, more challenging and realistic face detection datasets were created.

Histogram of Oriented Gradients method (HOG): It is one of the well-known human detection methods. It is a feature descriptor made of small regions of gradient orientation called. This method performs well if there is large number of images for training is giver, therefore needs a very careful selection of different training images.

Human Detection using a combination of face, head, and shoulder detection: Human detection can be done by identifying different body parts separately and combining them into one detection. These different methods such as Haar classifier, gradient maps, golden ratio, etc., are used for detecting face, head, and shoulder. The detection results were efficient but the time of detection was high.

R-CNN: Selecting large number of regions is very difficult problem. To solve this problem, R-CNN was introduced. In this method, the author uses selective search approach, in which first it extracts just 2000 regions from the frame, called the region proposals .But this method took approximately 47 seconds to give the detection and needed more amount of training time than other algorithms.

Fast R-CNN: Author Ross Girshick, solved the drawbacks of Region Based Convolutional Neural Networks(R-CNN) algorithm to build an algorithm that is faster for object detection and it was called Fast R-CNN .Now, in this, we fed the input video or image to the convolutional Neural Networks to generate a convolutional feature map. From there they identified the region proposals and wrap them into boxes. But the results were not satisfying.

Faster R-CNN: Both RCNN and Fast R-CNN algorithms uses selective search approach, thus they were slow when real-time object detection was the concern. Therefore, yet another algorithm was proposed, called the Faster R-CNN , in which the selective search algorithm was replaced by the object detection algorithm and it let the network learn the region proposals.

CHAPTER – 3

PROPOSED SYSTEM

The solutions for human detection have restrictions. For instance, people must be moving so that the system can distinguish other things and people, the object background must be plane or simple, or the resolution of image must be high. However, real-time scenarios always have both stationary and moving object, the object background might be complicated, and almost 80% videos in a visual surveillance system have a relatively low resolution. In this pandemic, one of the biggest challenges in the retail industry is measuring how many customers enter a store throughout the day. Maintaining track of people movements and social distancing rules in a commercial setting is crucial in the current pandemic situation.

Human Detection uses computer vision to detect humans within a video. The detection of object in real-time scenarios is significant trend in industries from various cities for the surveillance. This can be used in:

- Counts the pedestrians along a path.
- Analysing shopper behaviour or dwell time.
- Home or shop security cameras detecting intruders or visitors, etc.

The project will focus on human object detection and count of people in an image or in a real time video through camera or some other source.

We’ve worked on three different methods and algorithms to detect a human object in order of their increasing accuracy.

In this we can use various predefined methods and can detect the human in any image, video and can even get various factors like accuracy, each detections counting, etc.

Some common methods are :

- **Using Haar Cascade Classifier:**
  Haar Cascade is a feature based object detection algorithm. It was proposed by Paul Viola and Michael Jones in their paper 'Rapid Object Detection using a Boosted Cascade of Simple Features', published in 2001. Most commonly it is used for face detection. It has some pre-trained models for face detection, full body detection, upper body detection and lower body detection. In this example we will work with full body detection model.

- **Using HOG(Histogram of Oriented Gradients) :**
  The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.
- **Using Tensorflow:**
  TensorFlow is an open-source API from Google, which is widely used for solving machine learning tasks that involve Deep Neural Networks. And again this method gives even better accuracy than above two methods.

Here we have implemented the application using the third method and got almost the better accuracy.

CHAPTER – 4

MODULE DESCRIPTION

1. **Graphic User Interface**

   Created an interactive frontend part in the project. For this we used Tkinter GUI and added different buttons and labels.

   Additional Libraries used:

   Tkinter – for frontend GUI window

   PIL – for adding images to the tkinter GUI window

   Messagebox from tkinter – to show any message using dialog box

   Filedialog – to select images and video using select buttons.

   When we run the code, a GUI window will open with START and EXIT button on it.

2. **Detection & Counting through Image**

   This section works with real time images. Here will allow user to select any real time image from the local system and then user can detect the humans in it. And along with that it also gives the count of humans detected.

3. **Detection & Counting through Video**

   This section works with real time videos. Here will allow user to select any real time video from the local system and then user can detect the humans in it.

   Now in case of video, since it is running, while the detection process is going on user will be able to see the detected peoples and their count for each frames per second of the video.

4. **Detection & Counting through Camera**

   This section works somehow similar to case of video. Here user will be asked to first open the webcam, and it will detect humans that will comes in that webcam during the detection process.

5. **Average Accuracy & Enumeration Plots**

This section basically deals with the graphical representation of the data[4][5] we got from the detection process. Using this graphical representation, one can do the analysis of the human count and accuracy very well.

In our application, we have basically talked about two basic plots.

- **Enumeration Plot:** This plot basically represent the plot between humans count against each time interval. For this plot, the parameter we took on X-axis is time (in seconds) and on Y-axis, we took, human count at that particular time. The highest peak in this enumeration plot, indicates the maximum no. of people detected in whole detection process.
- **Avg. Accuracy Plot:** This plot basically represent the plot between Average Accuracy against each time interval. For this plot, the parameter we took on X-axis is time (in seconds) and on Y-axis, we took, average accuracy with which humans got detected at that particular time. The highest peak in this plot, indicates the maximum avg. accuracy with which people detected in whole detection process.

6. Crowd Reports

This section generates a pdf report in which we are mentioning max. human detected, max. accuracy and max. average accuracy.

In the final pdf report, the region is marked as crowded or not according to the crowd status in that particular real time video.

CHAPTER – 5

RESULTS AND DISCUSSION

Accuracy

Now here we have discussed about the main keypoint of all computer vision project i.e. Accuracy. During the detection process of human, we along with process also kept track of the accuracy with each human is getting detected in image, video and camera.

In our method, we have set the threshold accuracy for the detection process as 70%, so the object detected with accuracy more than the threshold accuracy, we declared it as the well detected human, and display detection indicator around that human during process. We have set this threshold in order to prevent false detection to det displayed while detection process.

Now whenever term accuracy comes, there is always a general question, "What is the maximum accuracy of the detection?" and that we have discussed in the next topic.

Maximum Accuracy

Since we got the factor of accuracy with which each human is getting detected, so we also kept the track of maximum accuracy which we are getting throughout the detection process. This factors basically tells us about the preciseness of our implemented application.

Maximum Average Accuracy

This factor comes in the case of detecting through video and camera. Because in case of image, since the image is static, there is no meaning of maximum avg. accuracy.

In case of video and camera, it basically reflects the maximum of all the average of accuracy that we get for each frames in running video and webcam.

Sample output

Further we have improved some user interfaces and added some additional features.

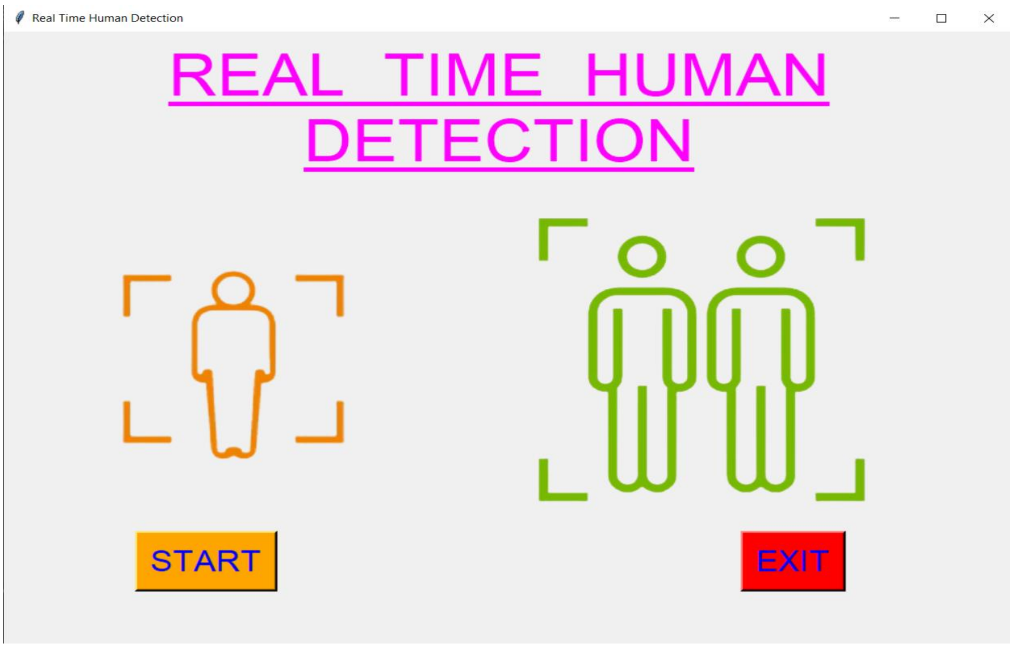
Used additional libraries like

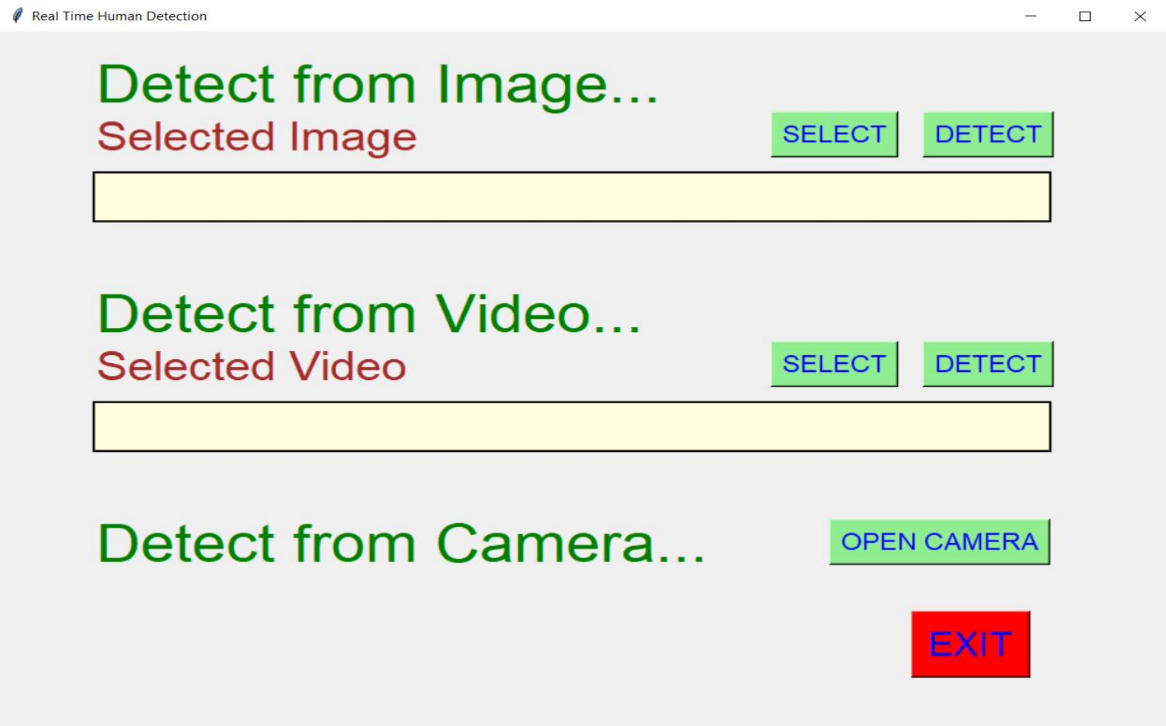
Imutils - a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.

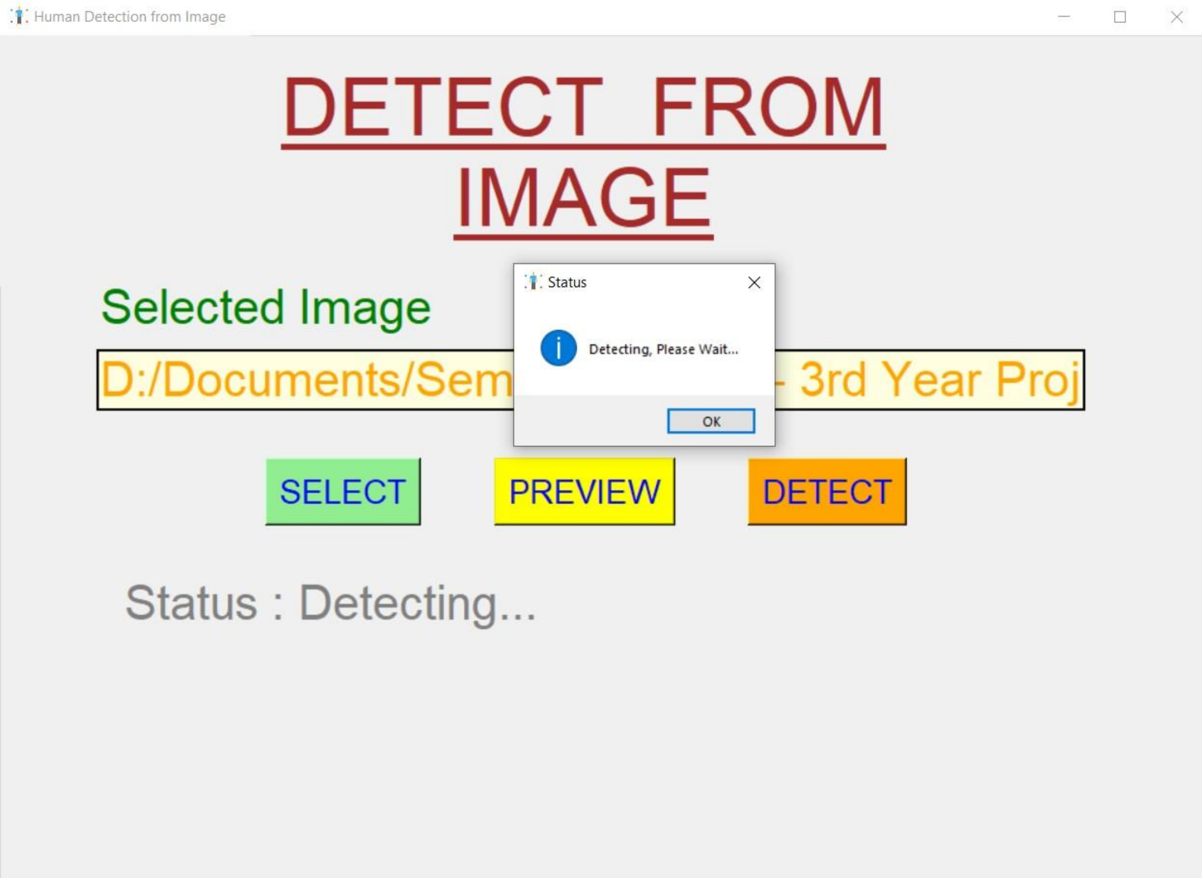
Argparse – this module makes it easy to write user-friendly command-line interfaces.

Also worked on the labelling of the human body detected by the program by the counter which tells us the count of the number of people at that particular instance in the video.

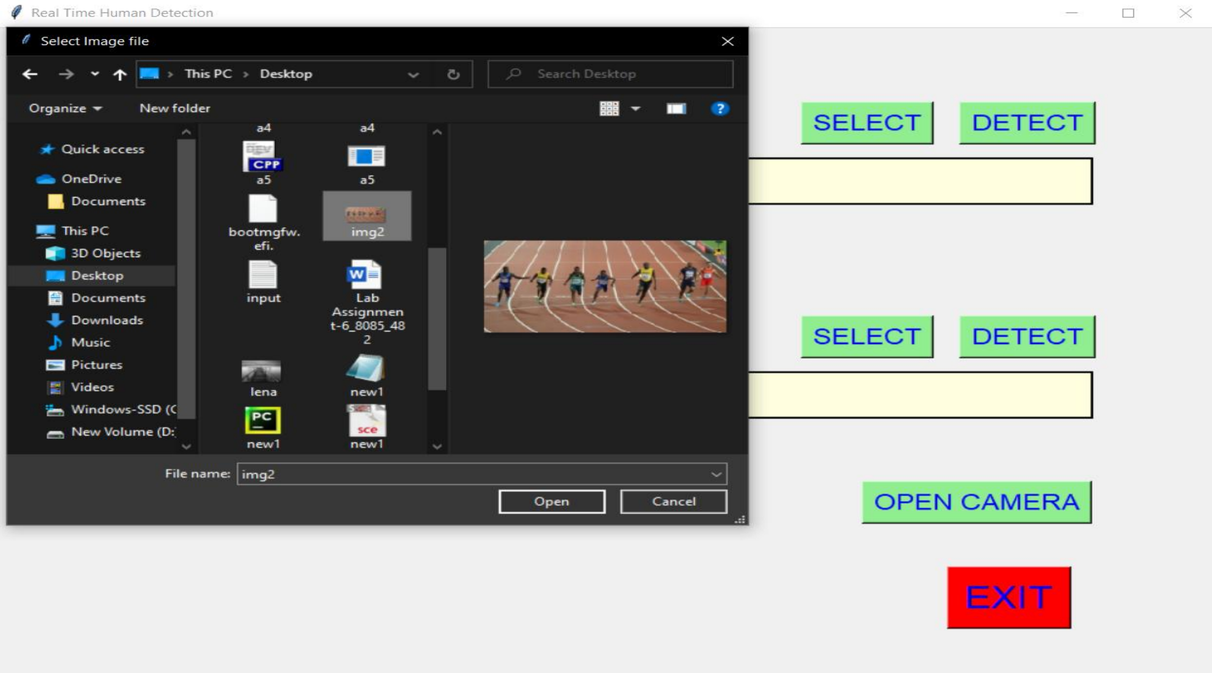
GUI Interface

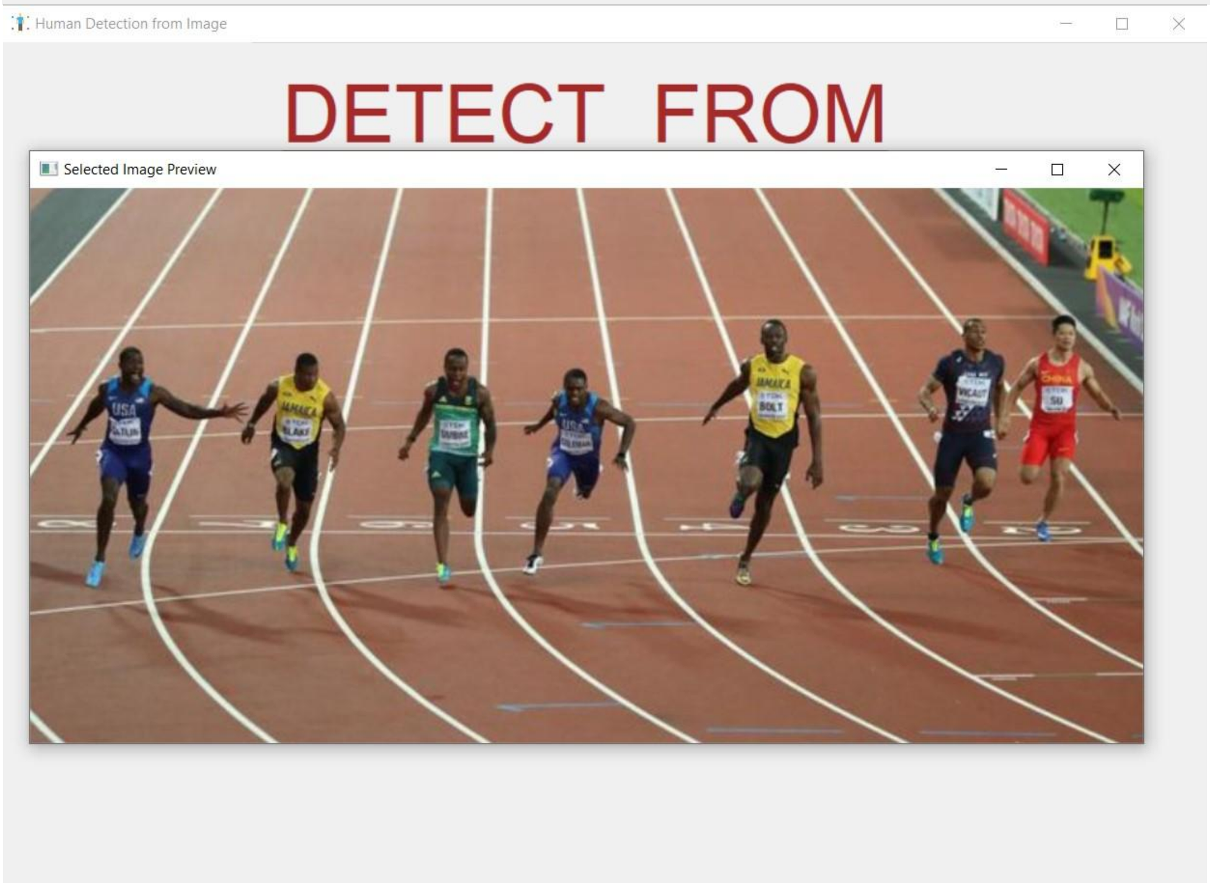


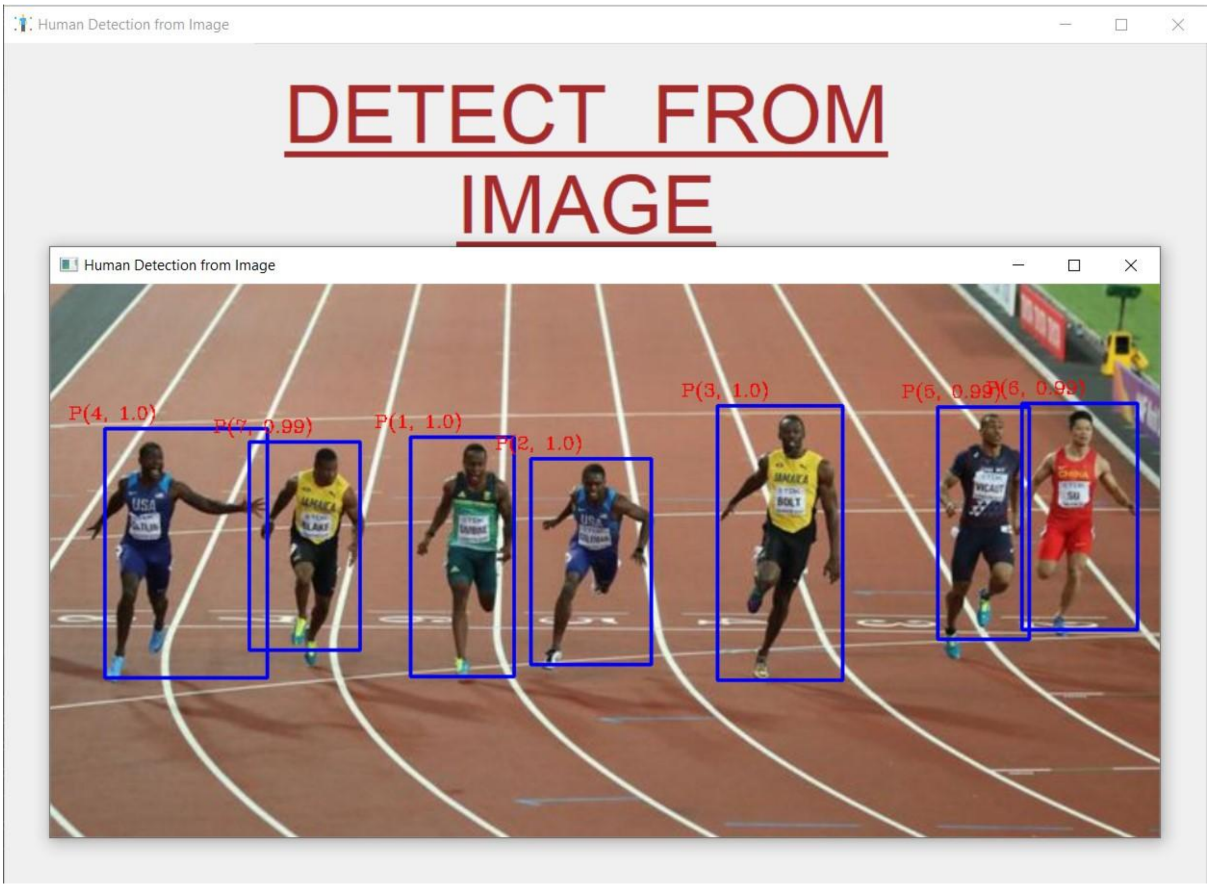




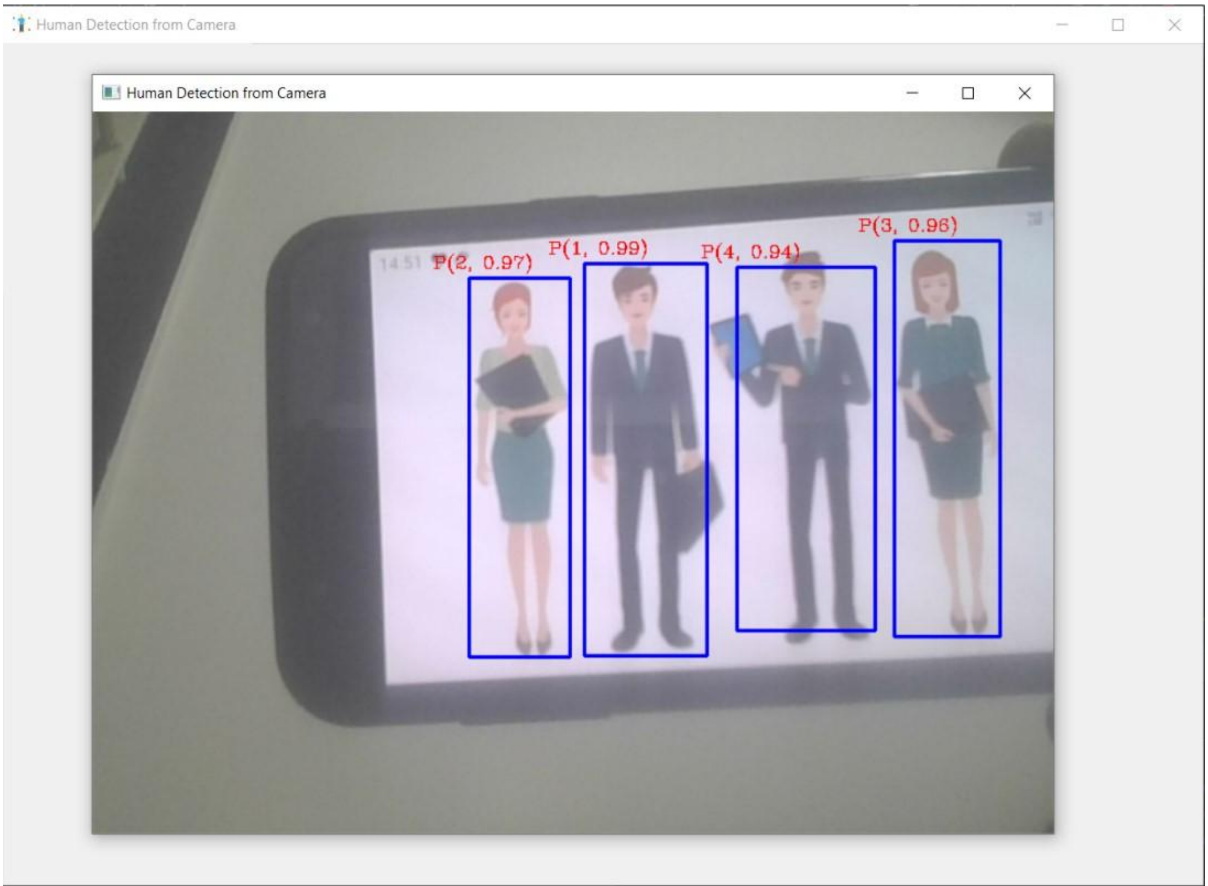
When we click on select button on image option, we get option to select image from the local system and when click on the detect button, an output image is shown which detected human and their count.


Real Time Human Detection
Select Image file
This PC > Desktop
Search Desktop
Organize
New folder
Quick access
OneDrive
Documents
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows-SSD (C
New Volume (D:
a4
a4
a5
a5
bootmgfw.efi.
img2
input
Lab Assignment-6_8085_482
lena
new1
new1
new1
File name:
img2
Open
Cancel
SELECT
DETECT
SELECT
DETECT
OPEN CAMERA
EXIT


Human Detection from Image
DETECT FROM
Selected Image Preview



Human Detection from Image
DETECT FROM IMAGE
Human Detection from Image
P(4, 1.0)
P(1, 1.0)
P(2, 1.0)
P(3, 1.0)



Human Detection from Camera
Human Detection from Camera
P(2, 0.97)
P(1, 0.99)
P(4, 0.94)
P(3, 0.96)

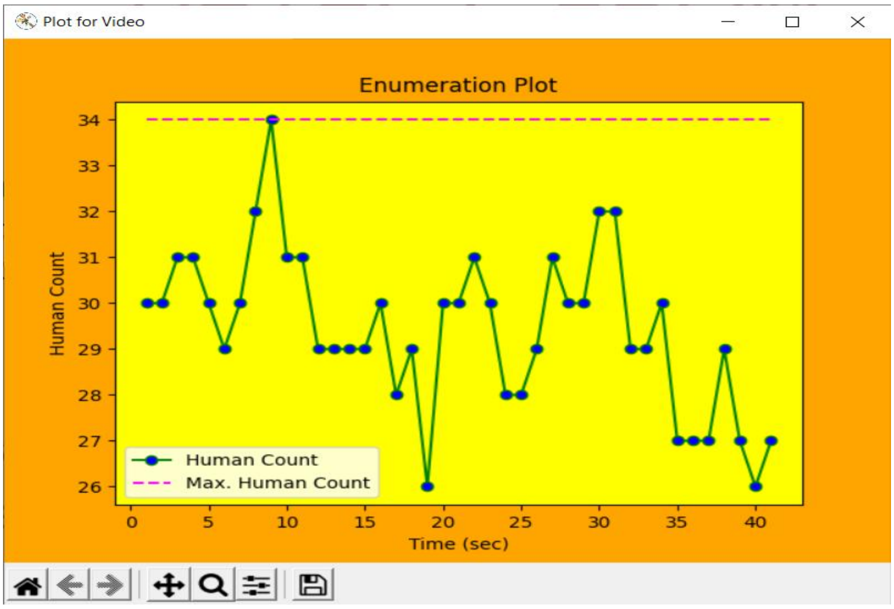
We implemented the tensorflow method of detecting the human using neural networks.
And we got better accuracy as compared to the 1st method that was Haar Cascade Classifier and the 2nd method that was using HOG descriptor.

Both the earlier method, gave almost the same accuracy and only few humans are only getting detected but with tensorflow method, accuracy increased very much and almost all the humans got detected and along with detecting and counting number of humans, we also got the accuracy of each human with which it is getting detected.

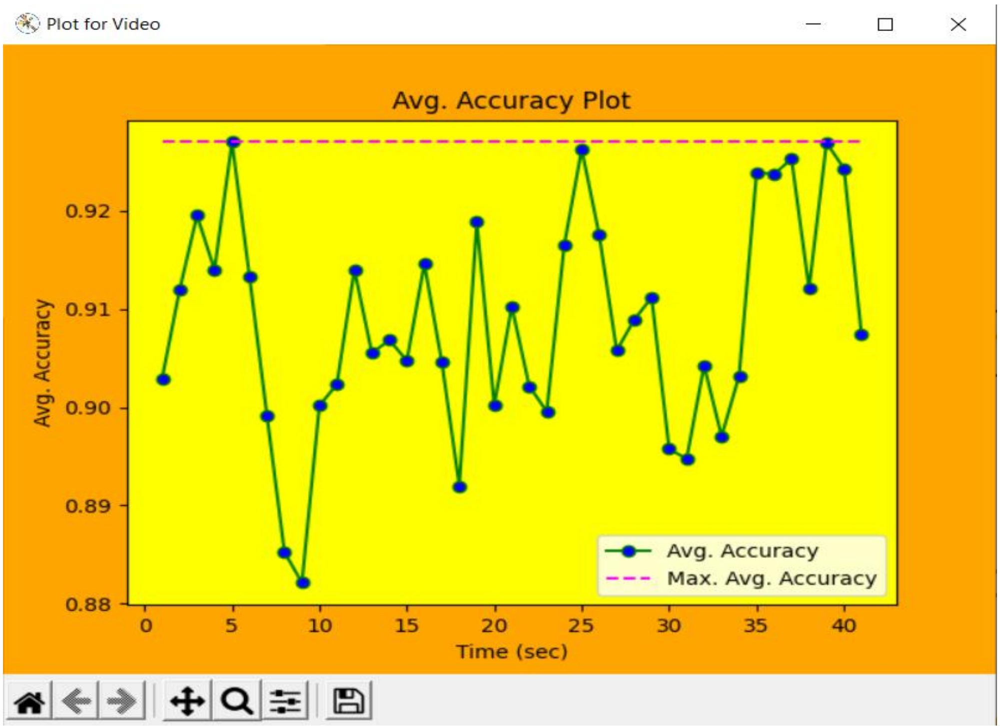
Graph Plotting

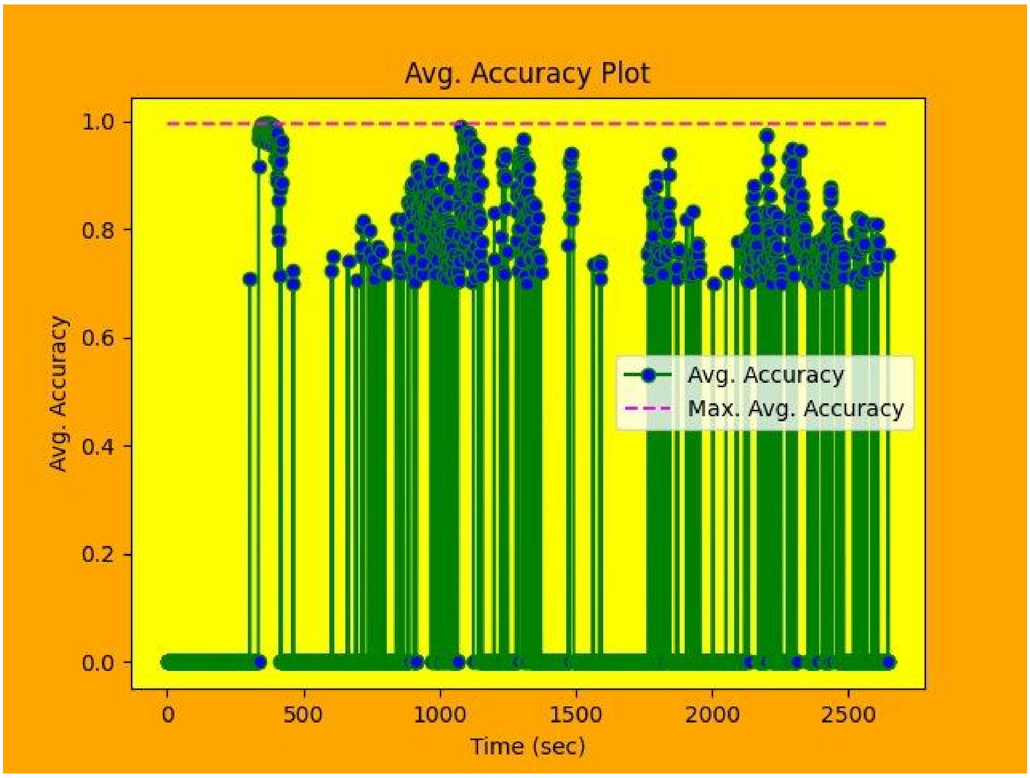
After detection and counting process, we plotted the two following plot using the data collected.

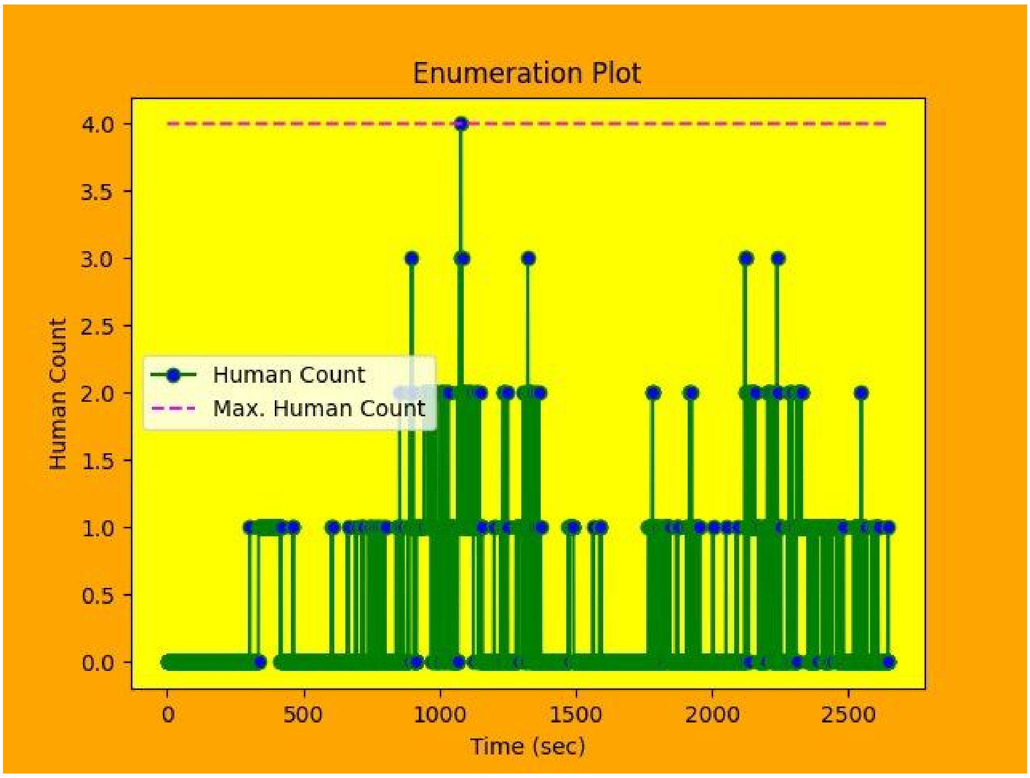
1st is Enumeration Plot :



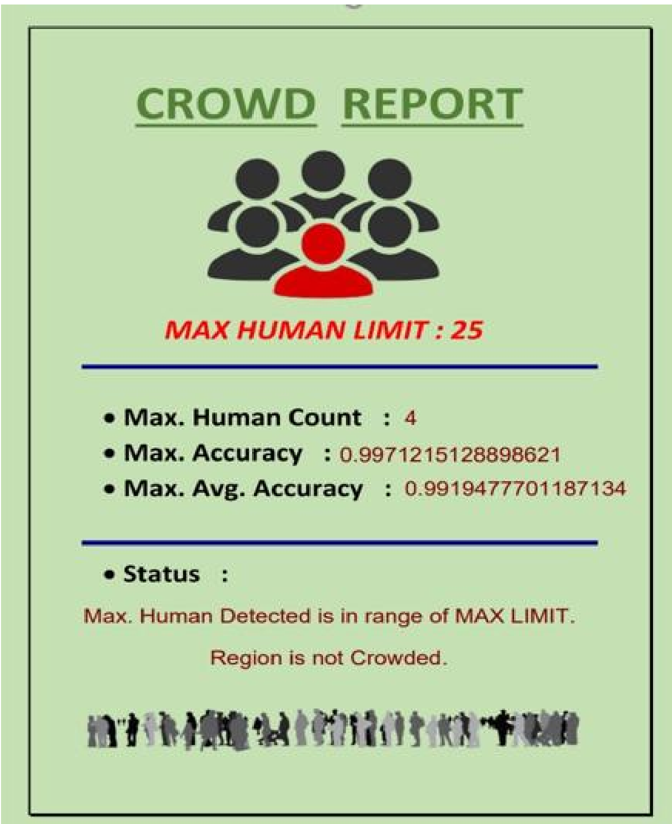
2nd is Average accuracy plot:

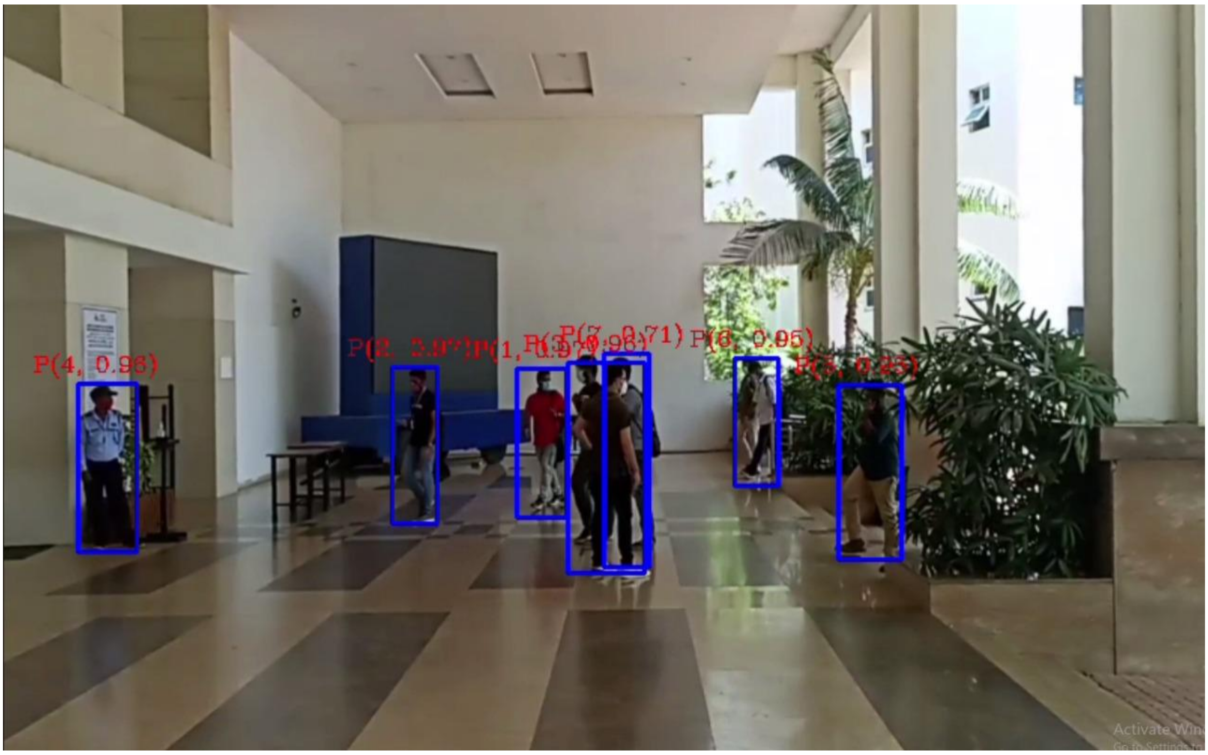



Avg. Accuracy Plot
1.0
0.8
0.6
0.4
0.2
0.0
Avg. Accuracy
0
500
1000
1500
2000
2500
Time (sec)
Avg. Accuracy
Max. Avg. Accuracy

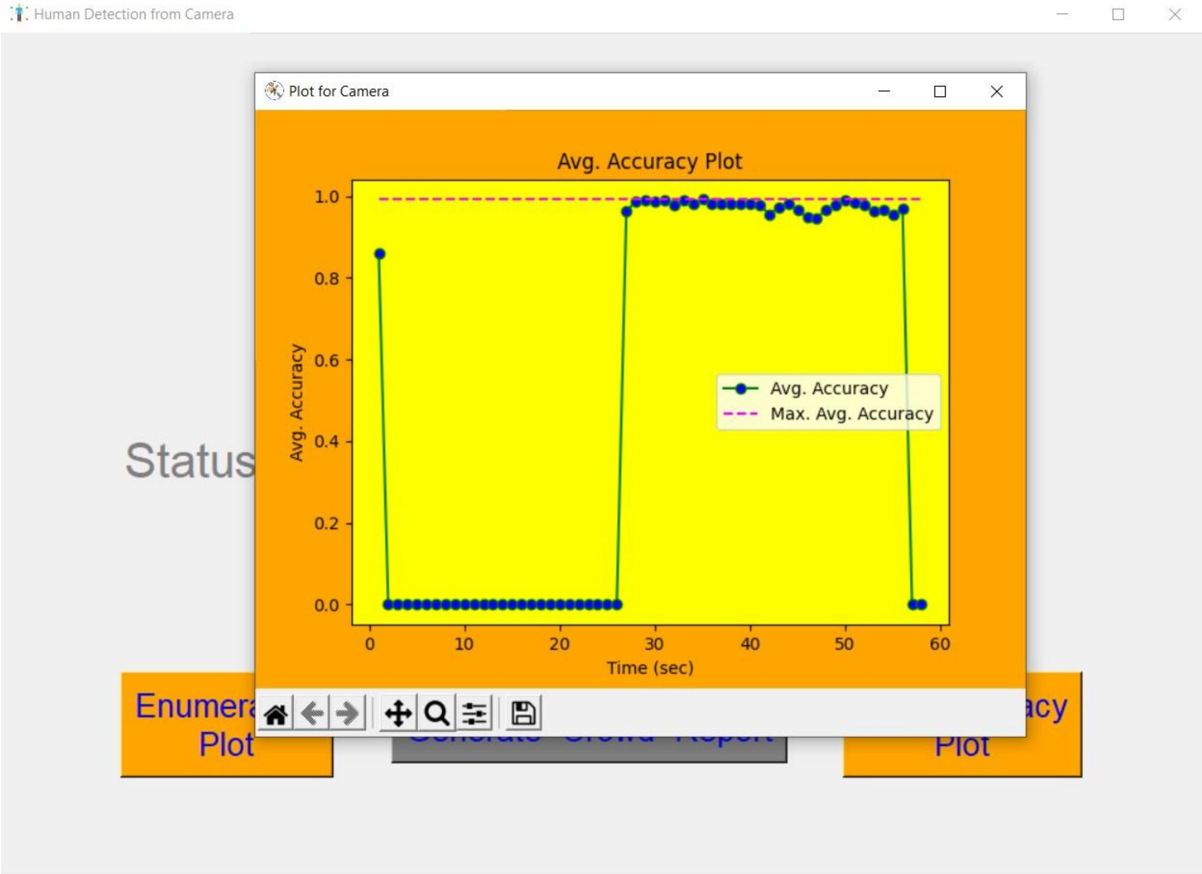

Enumeration Plot
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Human Count
0
500
1000
1500
2000
2500
Time (sec)
Human Count
Max. Human Count

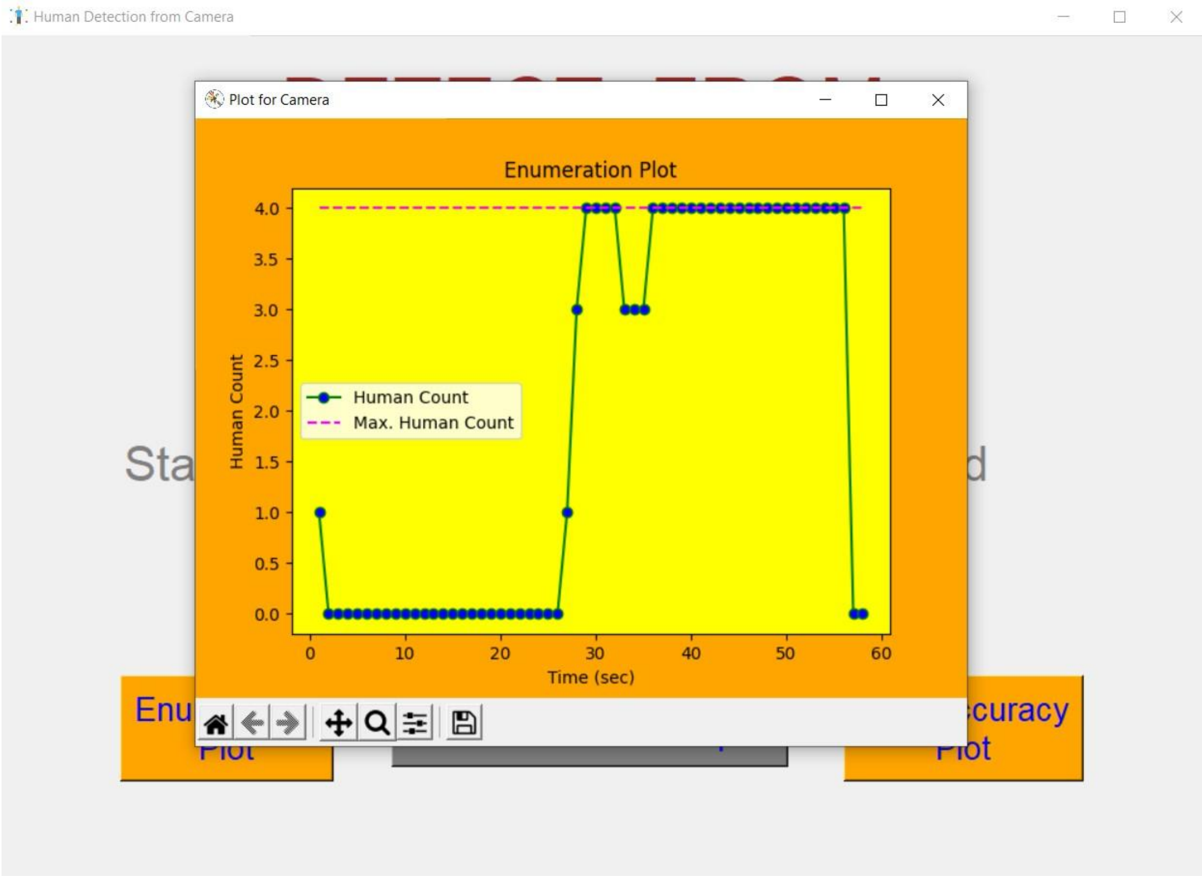
Lastly we prepared a crowd report in which we are putting max. human detected, max. accuracy and max. average accuracy.





A sample from AB1 Portico


Human Detection from Camera
Plot for Camera
Avg. Accuracy Plot
Avg. Accuracy
Time (sec)
Avg. Accuracy
Max. Avg. Accuracy
Status
Enumera
Plot
acy
Plot


Human Detection from Camera
Plot for Camera
Enumeration Plot
Human Count
Time (sec)
Human Count
Max. Human Count
Sta
Enu
curacy
Plot


CROWD REPORT
MAX HUMAN LIMIT : 25
• Max. Human Count : 34
• Max. Accuracy : 0.99791020154953
• Max. Avg. Accuracy : 0.9270146608352661
• Status :
Max. Human Detected is greater than MAX LIMIT.
Region is Crowded.

CHAPTER – 6

CODE

```python
# options ----------------------------
lbl1 = tk.Label(text="OPTIONS", font=("Arial", 50, "underline"),fg="brown")  # same way bg
lbl1.place(x=340, y=20)

# image on the main window
pathi = "Images/image1.jpg"
imgi = ImageTk.PhotoImage(Image.open(pathi))
paneli = tk.Label(window1, image = imgi)
paneli.place(x = 90, y = 110)

# image on the main window
pathv = "Images/image2.png"
imgv = ImageTk.PhotoImage(Image.open(pathv))
panelv = tk.Label(window1, image = imgv)
panelv.place(x = 700, y = 260)# 720, 260

# image on the main window
pathc = "Images/image3.jpg"
imgc = ImageTk.PhotoImage(Image.open(pathc))
panelc = tk.Label(window1, image = imgc)
panelc.place(x = 90, y = 415)

# created button for all three option
Button(window1, text="DETECT  FROM   IMAGE ->",command=image_option, cursor="hand2", font=("Arial",30), bg = "light green", fg = "blue").place(x = 350, y = 150)
Button(window1, text="DETECT  FROM  VIDEO ->",command=video_option, cursor="hand2", font=("Arial", 30), bg = "light blue", fg = "blue").place(x = 110, y = 300) #90, 300
Button(window1, text="DETECT FROM CAMERA ->",command=camera_option, cursor="hand2", font=("Arial", 30), bg = "light green", fg = "blue").place(x = 350, y = 450)
```

```python
# funcion defined to preview the selected video
def prev_vid():
    global filename2
    cap = cv2.VideoCapture(filename2)
    while (cap.isOpened()):
        ret, frame = cap.read()
        if ret == True:
            img = cv2.resize(frame, (800, 500))
            cv2.imshow('Selected Video Preview', img)
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break
        else:
            break
    cap.release()
    cv2.destroyAllWindows()


lbl1 = tk.Label(windowv, text="DETECT  FROM\nVIDEO", font=("Arial", 50, "underline"), fg="brown")
lbl1.place(x=230, y=20)
lbl2 = tk.Label(windowv, text="Selected Video", font=("Arial", 30), fg="green")
lbl2.place(x=80, y=200)
path_text2 = tk.Text(windowv, height=1, width=37, font=("Arial", 30), bg="light yellow", fg="orange", borderwidth=2,relief="solid")
path_text2.place(x=80, y=260)

Button(windowv, text="SELECT", command=open_vid, cursor="hand2", font=("Arial", 20), bg="light green", fg="blue").place(x=220, y=350)
Button(windowv, text="PREVIEW", command=prev_vid, cursor="hand2", font=("Arial", 20), bg="yellow", fg="blue").place(x=410, y=350)
Button(windowv, text="DETECT", command=det_vid, cursor="hand2", font=("Arial", 20), bg="orange", fg="blue").place(x=620, y=350)

info1 = tk.Label(windowv, font=("Arial", 30), fg="gray")  # same way bg
info1.place(x=100, y=440)
# info2 = tk.Label(windowv, font=("Arial", 30), fg="gray")  # same way bg
# info2.place(x=100, y=500)
```

```python
# the main process of detection in video takes place here
def detectByPathVideo(path, writer):
    global filename2, max_count2, framex2, county2, max2, avg_acc2_list, max_avg_acc2_list, max_acc2, max_avg_acc2
    max_count2 = 0
    framex2 = []
    county2 = []
    max2 = []
    avg_acc2_list = []
    max_avg_acc2_list = []
    max_acc2 = 0
    max_avg_acc2 = 0

    # function defined to plot the people detected in video
    def vid_enumeration_plot():
        plt.figure(facecolor='orange', )
        ax = plt.axes()
        ax.set_facecolor("yellow")
        plt.plot(framex2, county2, label = "Human Count", color = "green", marker='o', markerfacecolor='blue')
        plt.plot(framex2, max2, label="Max. Human Count", linestyle='dashed', color='fuchsia')
        plt.xlabel('Time (sec)')
        plt.ylabel('Human Count')
        plt.title('Enumeration Plot')
        plt.legend()
        plt.get_current_fig_manager().canvas.set_window_title("Plot for Video")
        plt.show()

    def vid_accuracy_plot():
        plt.figure(facecolor='orange', )
        ax = plt.axes()
        ax.set_facecolor("yellow")
        plt.plot(framex2, avg_acc2_list, label="Avg. Accuracy", color="green", marker='o', markerfacecolor='blue')
        plt.plot(framex2, max_avg_acc2_list, label="Max. Avg. Accuracy", linestyle='dashed', color='fuchsia')
        plt.xlabel('Time (sec)')
        plt.ylabel('Avg. Accuracy')
        plt.title('Avg. Accuracy Plot')
        plt.legend()
        plt.get_current_fig_manager().canvas.set_window_title("Plot for Video")
        plt.show()
```

```python
# function defined to detect inside the video
def det_vid():
    global filename2, max_count2, framex2, county2, max2, avg_acc2_list, max_avg_acc2_list, max_acc2, max_avg_acc2
    max_count2 = 0
    framex2 = []
    county2 = []
    max2 = []
    avg_acc2_list = []
    max_avg_acc2_list = []
    max_acc2 = 0
    max_avg_acc2 = 0

    video_path = filename2
    if (video_path == ""):
        mbox.showerror("Error", "No Video File Selected!", parent = windowv)
        return
    info1.config(text="Status : Detecting...")
    # info2.config(text="                                                  ")
    mbox.showinfo("Status", "Detecting, Please Wait...", parent=windowv)
    # time.sleep(1)

    args = argsParser()
    writer = None
    if args['output'] is not None:
        writer = cv2.VideoWriter(args['output'], cv2.VideoWriter_fourcc(*'MJPG'), 10, (600, 600))

    detectByPathVideo(video_path, writer)
```

```python
# ---------------------------- video section ------------------------------------------------------------
def video_option():
    # new windowv created for video section
    windowv = tk.Tk()
    windowv.title("Human Detection from Video")
    windowv.iconbitmap('Images/icon.ico')
    windowv.geometry('1000x700')

    max_count2 = 0
    framex2 = []
    county2 = []
    max2 = []
    avg_acc2_list = []
    max_avg_acc2_list = []
    max_acc2 = 0
    max_avg_acc2 = 0

    # function defined to open the video
    def open_vid():
        global filename2, max_count2, framex2, county2, max2, avg_acc2_list, max_avg_acc2_list, max_acc2, max_avg_acc2
        max_count2 = 0
        framex2 = []
        county2 = []
        max2=[]
        avg_acc2_list = []
        max_avg_acc2_list = []
        max_acc2 = 0
        max_avg_acc2 = 0

        filename2 = filedialog.askopenfilename(title="Select Video file", parent=windowv)
        path_text2.delete("1.0", "end")
        path_text2.insert(END, filename2)
```

Labelling images:

```
# for images ----------------------
lbl1 = tk.Label(windowi,text="DETECT  FROM\nIMAGE", font=("Arial", 50, "underline"),fg="brown")
lbl1.place(x=230, y=20)
lbl2 = tk.Label(windowi,text="Selected Image", font=("Arial", 30),fg="green")
lbl2.place(x=80, y=200)
path_text1 = tk.Text(windowi, height=1, width=37, font=("Arial", 30), bg="light yellow", fg="orange",borderwidth=2, relief="solid")
path_text1.place(x=80, y = 260)

Button(windowi, text="SELECT", command=open_img, cursor="hand2", font=("Arial", 20), bg="light green", fg="blue").place(x=220, y=350)
Button(windowi, text="PREVIEW",command=prev_img, cursor="hand2", font=("Arial", 20), bg = "yellow", fg = "blue").place(x = 410, y = 350)
Button(windowi, text="DETECT",command=det_img, cursor="hand2", font=("Arial", 20), bg = "orange", fg = "blue").place(x = 620, y = 350)

info1 = tk.Label(windowi,font=( "Arial", 30),fg="gray")
info1.place(x=100, y=445)
# info2 = tk.Label(windowi,font=("Arial", 30), fg="gray")
# info2.place(x=100, y=500)

def exit_wini():
    if mbox.askokcancel("Exit", "Do you want to exit?", parent = windowi):
        windowi.destroy()
windowi.protocol("WM_DELETE_WINDOW", exit_wini)
```

```
# function defined to plot the enumeration fo people detected
def img_enumeration_plot():
    plt.figure(facecolor='orange', )
    ax = plt.axes()
    ax.set_facecolor("yellow")
    plt.plot(framex1, county1, label="Human Count", color="green", marker='o', markerfacecolor='blue')
    plt.plot(framex1, max1, label="Max. Human Count", linestyle='dashed', color='fuchsia')
    plt.xlabel('Time (sec)')
    plt.ylabel('Human Count')
    plt.legend()
    plt.title("Enumeration Plot")
    plt.get_current_fig_manager().canvas.set_window_title("Plot for Image")
    plt.show()

def img_accuracy_plot():
    plt.figure(facecolor='orange', )
    ax = plt.axes()
    ax.set_facecolor("yellow")
    plt.plot(framex1, avg_acc1_list, label="Avg. Accuracy", color="green", marker='o', markerfacecolor='blue')
    plt.plot(framex1, max_avg_acc1_list, label="Max. Avg. Accuracy", linestyle='dashed', color='fuchsia')
    plt.xlabel('Time (sec)')
    plt.ylabel('Avg. Accuracy')
    plt.title('Avg. Accuracy Plot')
    plt.legend()
    plt.get_current_fig_manager().canvas.set_window_title("Plot for Image")
    plt.show()
```

```
# main detection process process here
def detectByPathImage(path):
    global filename1, max_count1, framex1, county1, max1, avg_acc1_list, max_avg_acc1_list, max_acc1, max_avg_acc1
    max_count1 = 0
    framex1 = []
    county1 = []
    max1 = []
    avg_acc1_list = []
    max_avg_acc1_list = []
    max_acc1 = 0
    max_avg_acc1 = 0
```

```python
# function defined to detect the image
def det_img():
    global filename1, max_count1, framex1, county1, max1, avg_acc1_list, max_avg_acc1_list, max_acc1, max_avg_acc1
    max_count1 = 0
    framex1 = []
    county1 = []
    max1 = []
    avg_acc1_list = []
    max_avg_acc1_list = []
    max_acc1 = 0
    max_avg_acc1 = 0

    image_path = filename1
    if(image_path==""):
        mbox.showerror("Error", "No Image File Selected!", parent = windowi)
        return
    info1.config(text="Status : Detecting...")
    # info2.config(text="                                                  ")
    mbox.showinfo("Status", "Detecting, Please Wait...", parent = windowi)
    # time.sleep(1)
    detectByPathImage(image_path)
```

```python
def argsParser():
    arg_parse = argparse.ArgumentParser()
    arg_parse.add_argument("-v", "--video", default=None, help="path to Video File ")
    arg_parse.add_argument("-i", "--image", default=None, help="path to Image File ")
    arg_parse.add_argument("-c", "--camera", default=False, help="Set true if you want to use the camera.")
    arg_parse.add_argument("-o", "--output", type=str, help="path to optional output video file")
    args = vars(arg_parse.parse_args())
    return args
```

Person detection code:

```python
import numpy as np
import tensorflow as tf
# import cv2
import time
import os
# import tensorflow.compat.v1 as tf
import tensorflow._api.v2.compat.v1 as tf
tf.disable_v2_behavior()


class DetectorAPI:
    def _init_(self):
        path = os.path.dirname(os.path.realpath(_file_))
        self.path_to_ckpt = f'frozen_inference_graph.pb'
        self.detection_graph = tf.Graph()

        with self.detection_graph.as_default():
            od_graph_def = tf.GraphDef()
            with tf.gfile.GFile(self.path_to_ckpt, 'rb') as fid:
                serialized_graph = fid.read()
                od_graph_def.ParseFromString(serialized_graph)
```

```
            tf.import_graph_def(od_graph_def, name='')

        self.default_graph = self.detection_graph.as_default()
        self.sess = tf.Session(graph=self.detection_graph)

        # Definite input and output Tensors for detection_graph
        self.image_tensor = self.detection_graph.get_tensor_by_name('image_tensor:0')
        # Each box represents a part of the image where a particular object was detected.
        self.detection_boxes = self.detection_graph.get_tensor_by_name('detection_boxes:0')
        # Each score represent how level of confidence for each of the objects.
        # Score is shown on the result image, together with the class label.
        self.detection_scores = self.detection_graph.get_tensor_by_name('detection_scores:0')
        self.detection_classes = self.detection_graph.get_tensor_by_name('detection_classes:0')
        self.num_detections = self.detection_graph.get_tensor_by_name('num_detections:0')

    def processFrame(self, image):
        # Expand dimensions since the trained_model expects images to have shape: [1, None,
None, 3]
        image_np_expanded = np.expand_dims(image, axis=0)
        # Actual detection.
        start_time = time.time()
        (boxes, scores, classes, num) = self.sess.run(
            [self.detection_boxes, self.detection_scores,
                self.detection_classes, self.num_detections],
            feed_dict={self.image_tensor: image_np_expanded})
        end_time = time.time()

        # print("Elapsed Time:", end_time-start_time)
        # print(self.image_tensor, image_np_expanded)
        im_height, im_width, _ = image.shape
        boxes_list = [None for i in range(boxes.shape[1])]

        for i in range(boxes.shape[1]):
            boxes_list[i] = (int(boxes[0, i, 0] * im_height),int(boxes[0, i,
1]*im_width),int(boxes[0, i, 2] * im_height),int(boxes[0, i, 3]*im_width))

        return boxes_list, scores[0].tolist(), [int(x) for x in classes[0].tolist()], int(num[0])

    def close(self):
        self.sess.close()
        self.default_graph.close()
```

CHAPTER – 7

CONCLUSION

In the last section of the project, we generate Crowd Report, which will give some message on the basis of the results we got from the detection process. For this we took some threshold human count and we gave different message for different results of human count we got form detection process.

Now coming to the future scope of this project or application, since in this we are taking any image, video or with camera we are detecting humans and getting count of it, along with accuracy. So some of the future scope can be :

- This can be used in various malls and other areas, to analyse the maximum people count, and then providing some restrictions on number of people to have at a time at that place.
- This can replace various mental jobs, and this can be done more efficiently with machines.
- This will ultimately leads to some kind of crowd-ness control in some places or areas when implemented in that area.

REFERENCES

[1] Programming Computer Vision with Python, 1st Edition, Jan Eric Solem, 2012, O' Reily

[2] Learning OpenCv, Adrian Kaehler and Gary Rost Bradski, 2008, O' Reily

[3] Deep Learning with Tensorflow, Giancarlo Zaccone, Md. Rezaul Karim, Ahmed Menshawy, 2017

[4] Python GUI Programming with Tkinter, Alan D. Moore, 2018

[5] Python Standard Library, Fredrik Lundh, 2001, O' Reily