Abstract

Object Detection has been an important topic in the advancement of computer vision systems. With the advent of Neural Network techniques, the accuracy for object detection has increased drastically.

The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the Neural Network learning based approach, which leads to slow and non-optimal performance.

In this project, we use a You Only Look Once (YOLO) learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the most challenging publicly available dataset (PASCAL VOC,MS COCO), on which a object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

ACKNOWLEDGEMENT

It is our pleasure to acknowledge the assistance of a number of people without whose help this project would not have been possible. First and foremost ,We would like to express our gratitude to **Mr. Sharad Gupta** , Assistant professor of department CEA ,our project guide for providing invaluable encouragement, guidance and assistance. We would like to thank the group member for the operation extended to us throughout the project. After doing this project we can confidently say that this experience has not only enriched us with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required being a successful professional.

Sakshi Gupta(171500283)

Shivam Singh(171500323)

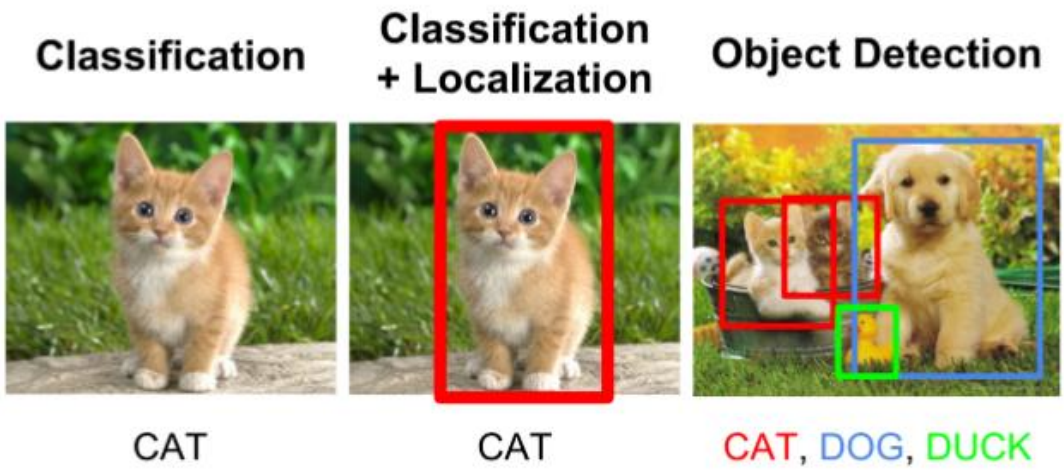
Gouri Shrivastava(171500113)

Surbhi Sharma(171500348)

1. INTRODUCTION

Object Detection is a software developed for real-time tracking object in videos and images. In computer vision were saturating on their accuracy before a decade. However, with the rise of Neural Network learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image.

A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box.



1.1 PROBLEM AND MOTIVATION

- Object detection is a technologically challenging and practically useful problem in the field of computer vision. Object detection deals with identifying the presence of various individual objects in an image.

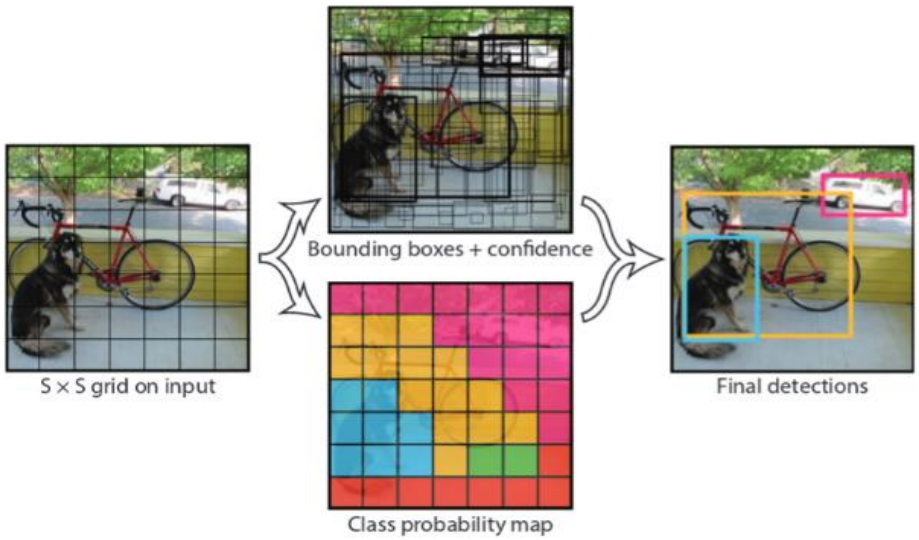
- object detection is face detection, that is used in almost all the mobile cameras. A more generalized (multi-class) application can be used in autonomous driving where a variety of objects need to be detected
- Also it has a important role to play in surveillance systems. These systems can be integrated with other tasks such as pose estimation where the first stage in the pipeline is to detect the object, and then the second stage will be to estimate pose in the detected region.

- The motive of object detection is to recognize and locate all known objects in a scene. Preferably in 3D space, recovering pose of objects in 3D is very important for robotic control systems. The information from the object detector can be used for obstacle avoidance and other interactions with the environment.

1.2 OVERVIEW

Our Object Detection basically has one main module You Only Look Once(YOLO) it base on Convolutional Neural Network. YOLO is an important framework that has led to improved speed and accuracy for object-detection tasks.

- First we divide the image into a grid of cells. Say it is 7x7. Now ground truth objects which have their object centers in their respective grid cells turns out to be +ve cells.



Yolo grid cell

- Each cell predicts 2 bounding boxes and each bounding box consists of 5 predictions :x,y,w,h, and confidence. The (x,y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image.

- Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. So for VOC with 20 classes the output would be ((4+1)x2+20) = 30. If the grid size is 7x7, we will have 7x7x30 outputs.

1.3 OBJECTIVE

- The project objective was to produce a working system for tracking objects in 3 dimensional space.

- The objective of the project was to begin from this spec, and design a solution to the problem. After a satisfactory solution was designed, the task came to implement the solution

- Also it has a important role to play in surveillance systems

- Objects detection is very important for robotic control systems.

Throughout the project, many problems arose. So objective of this project was creating Object detection software Results of this project have been applied to a range of products such as video surveillance, robotic vision and autonomous flight. In robotics, tracking is frequently used to provide a means for localization and mapping of an unknown environment .

1.4 CHALLENGES

The major challenge in this problem is that of the variable dimension of the output which is caused due to the variable number of objects that can be present in any

given input image. Any general machine learning task requires a fixed dimension of input and output for the model to be trained.

Another important obstacle for widespread adoption of object detection systems is the requirement of real-time (30fps) while being accurate in detection. The more complex the model is, the more time it requires for inference; and the less complex the model is, the less is the accuracy. This trade-off between accuracy and performance needs to be chosen as per the application. The problem involves classification as well as regression .

1.5 REQUIREMENT ANALYSIS

1.5.1 Hardware Requirement

1) Computer system with minimum 8GB RAM
2) 4 GB of available disk space minimum
3) Camera
4) 1280x800 minimum screen resolution

1.5.2 Software Requirement

1) 1)Window Jupyter Notebook
2) Tensorflow
3) Tensorboard
4) pip installment

i) Operating System: Windows 10

2. TOOLS AND TECHNOLOGY :

- **PYTHON :** Python has features like high-level built-in data structure , dynamic typing , and binding, which makes it attractive for rapid application development. It is an open-source software and easy for debugging .
- **USES OF PYTHON :**
  1) Python can be used to develop different applications like web applications, graphic user interface based application, software development applications, scientific and numeric applications, network programming, Games and 3D applications and other business application. It makes an interactive interface and easy developmentof applications.
  2) There are other applications for which python is used that are Robotics, web scraping, scripting, artificial intelligence , data analysis , machine learning , face detection ,video-based apllications and applications for images etc.

2.1 LIBRARIES USED :

- **NUMPY :** NumPy stands for 'Numerical Python' or 'Numeric Python'. It is an open source module of Python which provides fast mathematical computation on arrays and matrices. Since, arrays and matrices are an essential part of the Machine Learning ecosystem. NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation. Numpy can be imported into the notebook using.
  - >>> import numpy as np

- **TENSORFLOW : TensorFlow** is an open-source software library. **TensorFlow** was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well!
  - >>>import tensorflow as tf

- **OPENCV :** 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.
    - It supports a couple of programming languages namely: python, java, c and C++. On the other hand, it supports Windows, Linux, Mac Os and even the Android operating systems.

2.2 ALGORITHMS :

- **NEURAL NETWORK :** Neural networks are one of the learning algorithms used within machine learning. They consist of different layers for analyzing and learning data. Every hidden layer tries to detect patterns on the picture. When a pattern is detected the next hidden layer is activated and so on. The first layer detects edges. Then the following layers combine other edges found in the data, ultimately a specified layer attempts to detect a wheel pattern or a window pattern. Depending on the amount of layers, it will be or not be able to define what is on the picture, in this case a car.The more layers in a neural network, the more is learned and the more accurate the pattern detection is.

- **BOUNDING BOX :** In object detection, we usually use a bounding box to describe the target location. The bounding box is a rectangular box that can be determined by the xx and yy axis coordinates in the upper-left corner and the xx and yy axis coordinates in the lower-right corner of the rectangle. We will define the bounding boxes of the dog and the cat in the image based on the coordinate information in the above image. The origin of the coordinates in the above image is the upper left corner of the image, and to the right and down are the positive directions of the xx axis and the yy axis, respectivelyPA

- **YOLO :** YOLO is an extremely fast real time multi object detection algorithm. YOLO stands for **"You Only Look Once".**

  The algorithm applies a neural network to an entire image. The network divides the image into an S x S grid and comes up with bounding boxes, which are boxes drawn around images and predicted probabilities for each of these regions.The method used to

come up with these probabilities is logistic regression. The bounding boxes are weighted by the associated probabilities. For class prediction, independent logistic classifiers are used.

2.3 DATASET :

COCO:

- COCO is a large image dataset designed for object detection, segmentation, person keypoints detection, stuff segmentation, and caption generation. This package provides Matlab, Python, and Lua APIs that assists in loading, parsing, and visualizing the annotations in COCO.
- The Common Objects in Context (COCO) dataset has 200,000 images with more than 500,000 object annotations in 80 categories. It is the most extensive publicly available object detection database. The following image has the list of objects present in the dataset:

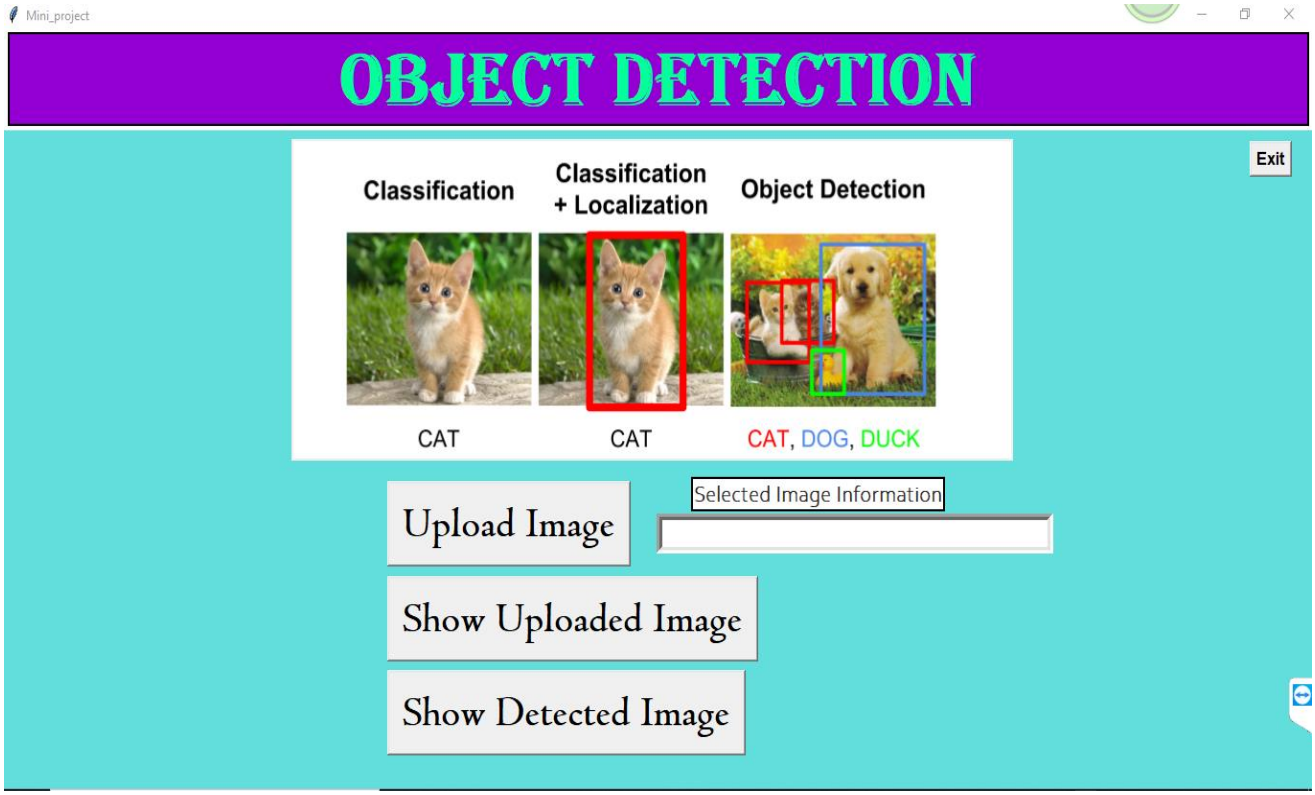
- The average number of objects is 7.2 per image. These are the famous datasets for the object detection challenge. Next, we will learn how to evaluate the algorithms against these datasets.

3) IMPLEMENTATION

3.1) FRONT- END DESIGN

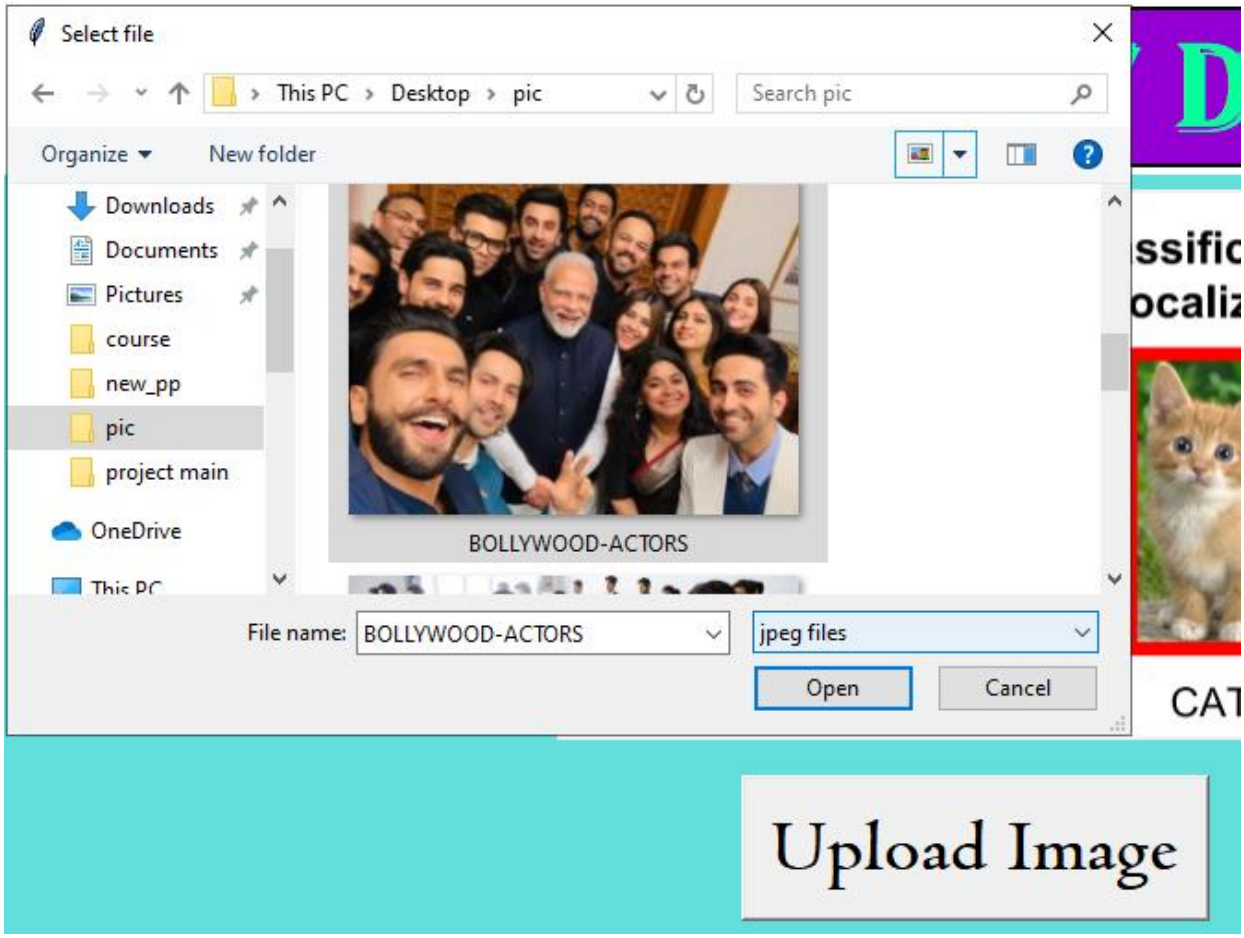
3.1.1 Introduction

- This is the first home screen titled by OBJECT DETECTION and home screen consists of three bottons. In order to go to the desired screen ; the users just have to click the related button.
- By clicking the Upload Image , Show Uploaded Image,Show Detected Image choose ,Exit buttons, their respective screens will appear.
- In home screen there is also a Entry Box where user selected image file path appear.



3.1.2Upload Image

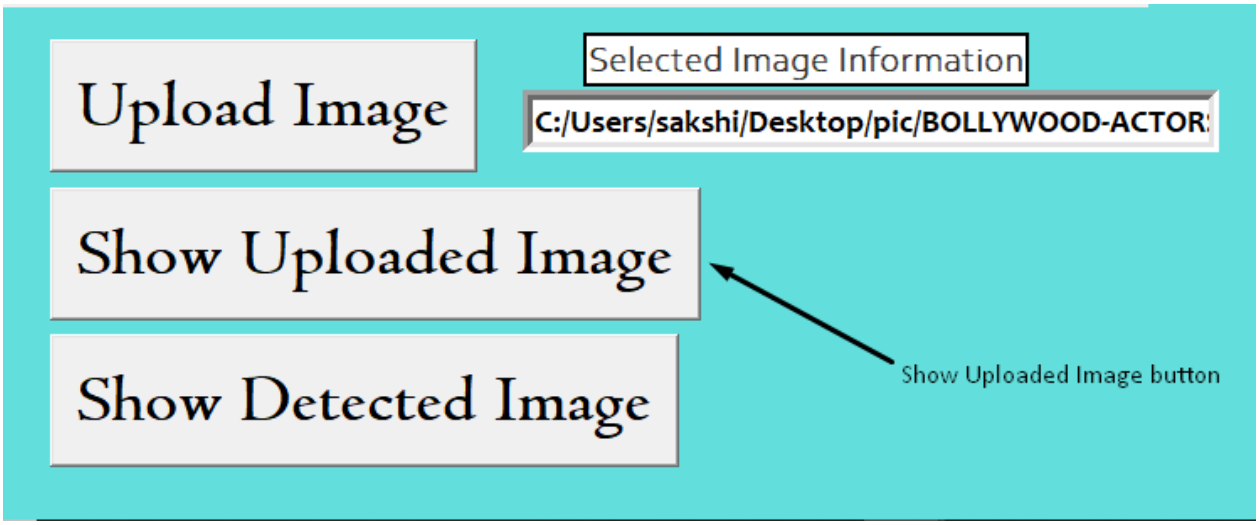
- Before using another function frist of all upload image by using Upload Image button
- This Upload Image button provides a function to browse any image which are present in our hole system.
- After clicking the upload button browse window popup , which are help to choose any image type file .
- It specially shows image file in our system
- And after the image has been successfully uploaded show Image Successfully Uploaded message and file path in Entry Box .
- If user not selected any image show Image not uploaded message in Entry Box .



3.1.3 Show Uploaded Image

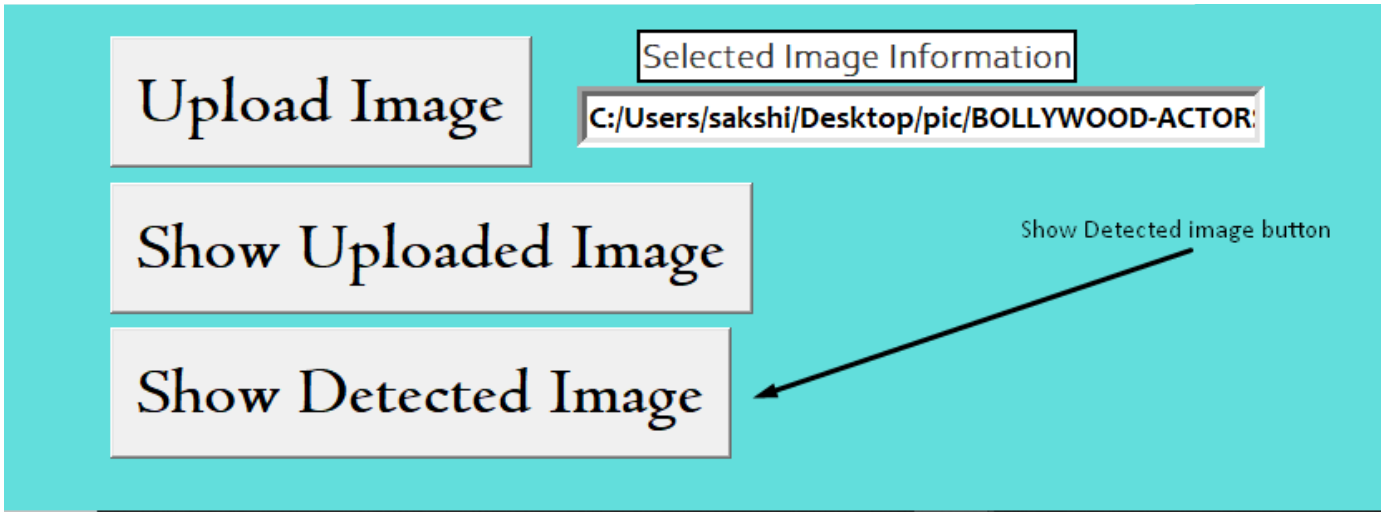
- It is provided functionality to show upload image by using this button .

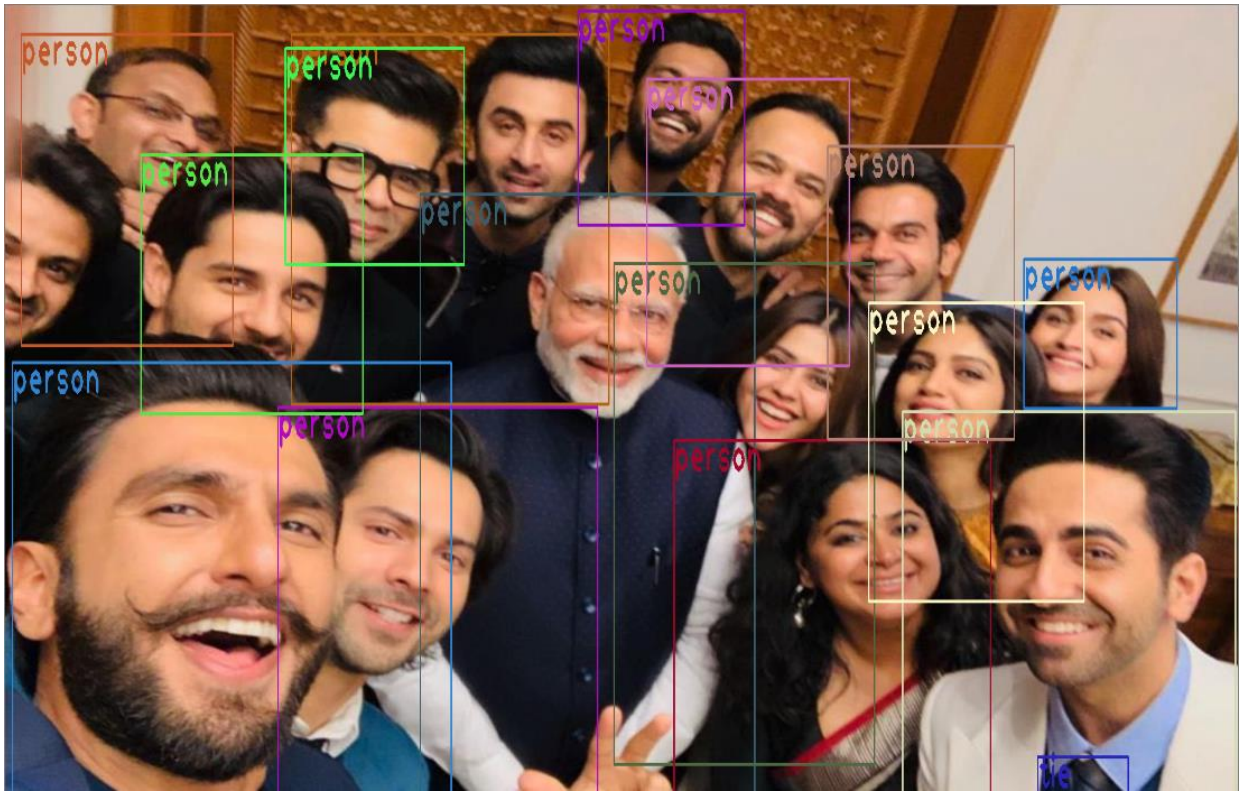
- When we click this Show Upload Image button show the uploaded image on a separate screen .



3.1.4 Show Detected Image

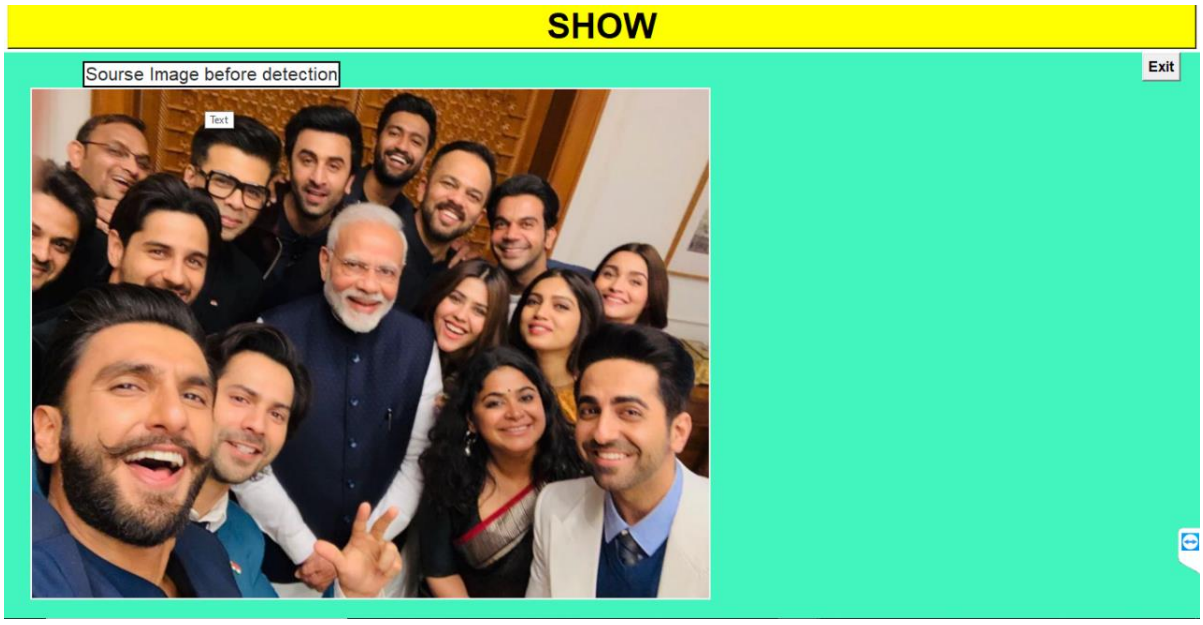
- After using both upper button functionality and ensure that image is uploaded successfully use this Show Detected Image . .
- After clicking this Show Detected Image button , the final Object Detected Image appears in new popup window.





3.1.5 New Window(SHOW)

- It is provided functionality to show upload image by using Show Uploaded Image button .
- When we click this Show Upload Image button show the uploaded image on a separate screen .



3.2) BACK- END DESIGN

3.2.1 Home Screen

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method

Python with tkinter outputs the fastest and easiest way to create the GUI applications.

Creating a GUI using tkinter is an easy task.

- Importing the module – tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets.

```
panel = Tk()
contant=''
panel.geometry("1000x700+120+120")
panel.title("Mini_project")
#--------make fram--------
top=Frame(panel,height=150,bg='white')
top.pack(fill=X)

bottom=Frame(panel,height=800,bg='#62dedc')
bottom.pack(fill=X)

 #----------top fram design----------
label1 = Label(top,text="OBJECT DETECTION",fg='#05ff9b',bg='dark violet',relief="solid",font=("Algerian",55,"bold"))
label1.pack(fill=BOTH,pady=4,padx=4)
label2 = Label(bottom,text="Selected Image Information",fg='gray17',bg="white",relief="solid",font=("Corbel Light",16,"bold")).place(x=717,y=321)
b1 = Button(bottom,text = "Upload Image",activebackground='#eaff05',relief=RAISED,font=("Centaur",30,'bold'),command=lambda :src())
b1.place(x=400,y=325)
#--------entry_box---------
entry_box=Entry(font='Candara 14 bold', width=40,bd=6)
entry_box.insert(0,'')
entry_box.place(x=681,y=450)
#-------------------main_screen_image------------
imge=Image.open('C:\\Users\\sakshi\\Desktop\\project main\\img.png')
sizes=(1500,580)
imge.thumbnail(sizes)
photo=ImageTk.PhotoImage(imge)
label7=Label(bottom,image=photo)
label7.place(x=300,y=8)

b2 = Button(bottom,text="Exit",relief=RAISED,font=("arial",12,'bold'),highlightcolor='#eaff05',command = panel.destroy)
b2.place(x=1300,y=10)
b1 = Button(bottom,text = "Show Uploaded Image",activebackground='#fff305',relief=RAISED,font=("Centaur",30,'bold'),command=secscreen)
b1.place(x=400,y=413)
b1 = Button(bottom,text = "Show Detected Image",activebackground='#ffa305',relief=RAISED,font=("Centaur",30,'bold'),command=run)
b1.place(x=400,y=500)
```

3.2.2 Function Button

Upload Image

```
def src():
    global imgpath
    panel.filename =  filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (("jpeg files","*.jpg"),("all files","*.*")))
    #root.filename.place(x=100,y=80)
    #print(panel.filename)

    if (panel.filename)=='':
        entry_box.insert(0,'No Image Selected')
        return (panel.filename)

    else:
        tkinter.messagebox.showinfo('Image','Successfully Uploaded !!')
        a=str(panel.filename)
        entry_box.insert(0,a)
    return (panel.filename)
```

Show Uploaded Image

```
def secscreen():
    panel2 = Toplevel()
    panel2.title("Mini_Project01")
    panel2.geometry("1000x700+120+120")
    top=Frame(panel2,height=150,bg='white')
    top.pack(fill=X)

    bottom=Frame(panel2,height=1000,bg='#42f5bf')
    bottom.pack(fill=X)

    label3=Label(top,text='SHOW',fg='black',bg='yellow',relief='raised',font=("arial",30,"bold"))
    label3.pack(fill=BOTH,pady=4,padx=4)
    b2 = Button(bottom,text="Exit",relief=RAISED,font=("arial",12,'bold'),command = panel2.destroy)
    b2.place(x=1300,y=0)
    label5 = Label(bottom,text="Sourse Image before detection",fg='gray17',bg="white",relief="solid",font=("arial",16,""))
    label5.place(x=90,y=10)
    #-----------image--------
    global contant
    contant=img()
    if contant =='No Image Selected':
        imge=Image.open('C:\\Users\\sakshi\\Desktop\\project main\\sorry.jpg')
        sizes=(1500,580)
        imge.thumbnail(sizes)
        photo=ImageTk.PhotoImage(imge)
        label6=Label(bottom,image=photo)
        label6.place(x=30,y=40)
    else :
        imge=Image.open(str(contant))
        sizes=(1500,580)
        imge.thumbnail(sizes)
        photo=ImageTk.PhotoImage(imge)
        label6=Label(bottom,image=photo)
        label6.place(x=30,y=40)
    Toplevel.mainloop()
```

Show Detected Image

```
def run():
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    classes = []
    with open("coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
        layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    c=imgpt()
    if c=='No Image Selected':
        tkinter.messagebox.showinfo('Image','Frist Upload Image !!')
        c=clear()
        exit
    else:
        img = cv2.imread(str(c))
        print(img.shape)
        img = cv2.resize(img, None, fx=2, fy=0.9)
        height, width, channels = img.shape
```

Add weight file

```
    class_ids = []
    confidences = []
    boxes = []

    for out in outs:

        for detection in out:

            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

            if confidence > 0.5:


            # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
```

Make grid for object

```
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
font = cv2.FONT_HERSHEY_PLAIN
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[i]
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label, (x, y + 30), font, 3, color, 3)
#cv2.imshow("Image", img)
height=(1100,770)
output=cv2.resize(img,height)
cv2.imshow("Image", output)
c=clear()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

show image using OpenCv

5. Conclusion

- An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision .
- Custom dataset was created using labeling and the evaluation was consistent. This can be used in real-time applications which require object detection for pre-processing .

- An important scope would be to train the system on a video sequence for usage in tracking applications. Addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.

GITHUB REPOSITORY LINK

https://github.com/shivamsingh3238/Object_detection