# OBJECT DETECTION SYSTEM

**A PROJECT REPORT**

***Submitted by***

| | |
|---|---|
| **Ameya Bavkar** | **(FS18IF047)** |
| **Bhavya Singh** | **(FS18IF053)** |
| **Sanjay Prajapati** | **(FS18IF054)** |

***in partial fulfillment for the award of***

**Diploma**

**IN**

**INFORMATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**GOVERNMENT POLYTECHNIC MUMBAI**

**JUNE 2021**

# GOVERNMENT POLYTECHNIC MUMBAI

**DEPARTMENT OF INFORMATION TECHNOLOGY**

## CERTIFICATE

This is to certify that the project entitled **" OBJECT DETECTION SYSTEM "** is the bonafide work of " **Ameya Bavkar (FS18IF047) , Bhavya Singh (FS18IF053) , Sanjay Prajapati (FS18IF054) "** , submitted in partial fulfillment of the requirements for the award of Diploma in Information Technology of Government Polytechnic Mumbai.

Dr. R. A. Patil
**HEAD OF THE DEPARTMENT**

Mrs. Dipali Gosavi
**PROJECT GUIDE**
**Lecturer in Information Technology**

**EXTERNAL EXAMINER**

Mrs. Swati. D. Deshpande
**PRINCIPAL**

# DECLARATION

We hereby declare that the project entitled **" OBJECT DETECTION SYSTEM "** being submitted by us towards the partial fulfillment of the requirements for the award of Diploma in Information Technology is a project work carried by us under the supervision of **Mrs. Dipali Gosavi** and have not been submitted anywhere else. We will be solely responsible if any kind of plagiarism is found.

Date :- 16/07/2021

**Team Members :-**

| | |
|---|---|
| Ameya Bavkar | (FS18IF047) |
| Bhavya Singh | (FS18IF053) |
| Sanjay Prajapati | (FS18IF054) |

## ACKNOWLEDGEMENT

We like to share our sincere gratitude to all those who help us in completion of this project. During the work we faced many challenges due to our lack of knowledge and experience but these people help us to get over from all the difficulties and in final compilation of our idea to a shaped sculpture.

We would like to thank **Mrs. Dipali Gosavi** for her governance and guidance, because of which our whole team was able to learn the minute aspects of a project work.

We would also like to show our gratitude to our Project Coordinators **Dr. R. A Patil** and **Mrs. Namrata A. Wankhade** for their continuous help and monitoring during the project work.

All of our team is thankful to all the Faculties and Staff of the Department of Information Technology , Government Polytechnic Mumbai , for their help and support towards this project and our team.

We are also thankful to our friends and most of all to our parents who have inspired us to face all the challenges and win all the hurdles in life

Thank you All.

**TEAM MEMBERS**

Ameya Bavkar (FS18IF047)
Bhavya Singh (FS18IF053)
Sanjay Prajapati (FS18IF054)

# ABSTRACT

Object detection is a huge, dynamic and challenging field of computer vision. If an object can be identified in the image, it is called image positioning, in the case where there are multiple objects in the image for object recognition. A semantic in digital images and videos. The aim is to recognize objects using the You Only Look Once (YOLO) approach. This method has several advantages over other object detection algorithms. In other algorithms like Convolutional Neural Network, Fast Convolutional Neural Network, the algorithm does not see the picture completely, but in YOLO the algorithm looks at the picture completely by predicting the bounding boxes using a convolutional network and the class probabilities for these frames and recognizes the pictures faster compared to other algorithms.

Object detection applications include object tracking, video surveillance, pedestrian detection, people counting, self-driving cars, face recognition, sports ball tracking, etc.

.

# CONTENT

.

.

# CHAPTER 1

# INTRODUCTION

.

# INTRODUCTION

Object detection is the act of identifying and detecting real-world object instances in pictures or videos, such as a vehicle, bike, TV, flowers, and persons. An object detection approach helps you to comprehend the intricacies of an image or video by recognising, localising, and detecting numerous things inside an image. It is commonly used in image retrieval, security, surveillance, and advanced driver assistance systems (ADAS). Object detection from video is a key challenge in video surveillance applications these days. The object detection approach is used to locate and cluster pixels of necessary objects in video sequences. The identification of an item in a video sequence is important in a variety of applications, particularly video surveillance applications.

Objects in a picture can be easily detected and identified by humans. The human visual system is quick and precise, and it is capable of doing complicated tasks such as recognizing several objects with minimal conscious thinking. We can now quickly train computers to recognize and categorize numerous items inside a picture with high accuracy because to the availability of enormous quantities of data, faster GPUs, and improved algorithms. Object detection in computer vision refers to the detection and identification of an object in an image or video. Feature extraction , feature processing , and object classification are the major processes in object detection. Object detection obtained outstanding performance with several classical approaches, which may be characterized in four ways: bottom feature extraction, feature coding, feature aggregation, and classification. thee feature extraction is critical in the object detection and recognition procedure.

There will be more redundant information that can be modelled to obtain greater performance than earlier point-of-interest detection. Previously, scale-invariant feature transformations (SIFT) and histograms of directed gradients were employed.

# CHAPTER 2
# LITERATURE REVIEW

.

# LITERATURE REVIEW

Neural networks are a class of algorithms that are modelled after the human brain and are used to recognize patterns. They perceive sensory data via a raw input device perception, marking, or grouping method. Face recognition, identification of people in photographs, and recognition of facial emotions are all skills that may be learned. Recognize things in photographs or videos. Clustering and grouping are two methods for identifying similarities. With categorization, deep learning might build links between pixels in a photograph and a person's name, for example. Over time, it has been demonstrated that neural networks surpass other algorithms in terms of accuracy and speed and there are various ways such as through CNN (Convolutional Neural Network), RNN (Recurrent Neural Networks ), etc.

To extract qualified deep characteristics, most researchers employ deep learning approaches. The demand for speedy and precise object detection systems is growing, thanks to the emergence of autonomous vehicles, smart video monitoring, facial recognition, real time detection, video analysis , image recognition and other people-counting applications. This makes object recognition far more difficult than its traditional counterpart in computer vision, picture recognition. Item detection is described as a classification issue in which we take fixed-size gaps from the input object at all available locations and feed these patches into an image classifier.

CNNs consist of neurons with learning weights and biases such as neural networks. Each neuron receives multiple inputs, takes over a weighted sum, passes it through an activation function, and provides an output response. YOLO is one of the fastest object detection algorithms available. Although it is no longer the most precise algorithm for object detection, it is a decent choice when you need real-time detection without sacrificing too much precision. YOLO predicts various bounding boxes and category probabilities for these boxes using a single neural network. You Only Look Once: Unified, Real-Time Object Detection, by Joseph Redmon. Their prior work is on detecting objects using a regression algorithm. To get high accuracy and good predictions they have proposed YOLO algorithm.

.

In our Project we have used the Yolo ( You Only Look Once ) as a name implies it only looks for the object for the one time and or stegonate it for just one time and pass it through the neural networks and the detection will take place .In Our Project we have implemented the real time analysis of objects , Video analysis , Image analysis which will use the same concept for detection and before we have just done with the real time detection after that we have implemented this same into the video and image before we were facing the problem with detection because we were not having the proper dataset which leads to the falls data and its accuracy got down we have taken the more dataset to make it accurate and perform at the good accuracy .

Secondly , we were having the issue with the Fps of the detection in video and real it was slow near to 4 or 5 Fps due to which it was very slow and the detection was taking place at a very low rate and this was causing some times the detection to be crashed and after that we have find the solution to run the detection and machine learning model on the GPU using CUDA which will increase the speed of detection and that helps in increasing the Speed of the Detection . For doing these we have compiled the Cuda and make it fast .

To make it easily usable we need better GUI so , we have also focused on making the GUI as we know that the web applications are good and faster then the traditional GUI they are more responsive than the traditional GUI so we have taken that into consideration and created the web based application in python using cross platforming this functionality provided by the eel framework and it provide better user experience and make it easy to access . We have created the Multi detection model at the same time in all our modules which provide more and more objects to get detected on the basis of the dataset the more and more dataset provided the better the detection will take place.

.

# CHAPTER 3
# YOLO FRAMEWORK

.

## 3.1. YOLO (YOU ONLY LOOK ONCE)

The phrase "You Only Look Once" is abbreviated as YOLO. Unified, Object Detection, by Joseph Redmon. This is an algorithm for detecting and recognizing different items in a photograph (in real-time). Object detection in YOLO is done as a regression problem, and the identified photos' class probabilities are provided.

YOLO is a real-time object identification technique that use neural networks. Because of its speed and precision, this algorithm is very popular. It has been used to identify traffic signals, pedestrians, parking metres , and animals in a variety of applications.

Convolutional neural networks (CNN) are used in the YOLO method to recognize objects in real time. To detect objects, the approach just takes a single forward propagation through a neural network, as the name suggests.

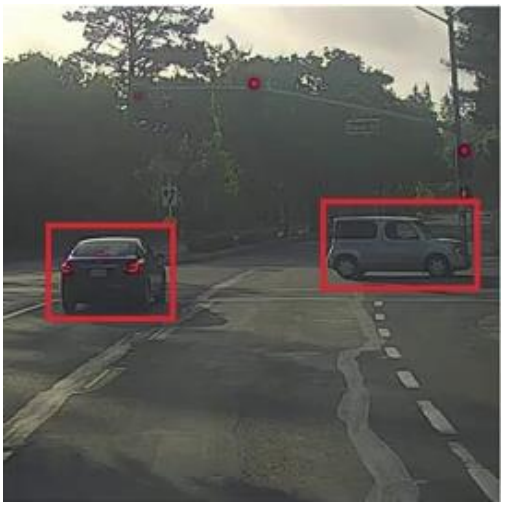
This indicates that a single algorithm run is used to forecast the entire image. The CNN is used to forecast multiple bounding boxes and class probabilities at the same time. YOLO focuses on enhancing the speed and simplicity of detection jobs in particular.

On the other hand, the YOLO framework (You Only Look Once) approaches object identification in a unique way. It predicts the bounding box coordinates and class probabilities for these boxes using the full image as a single instance. The most significant benefit of adopting YOLO is its incredible speed; it can process 45 frames per second. YOLO also recognizes generalized object representations. This is one of the best object detection algorithms, with a performance that is comparable to the R-CNN algorithm

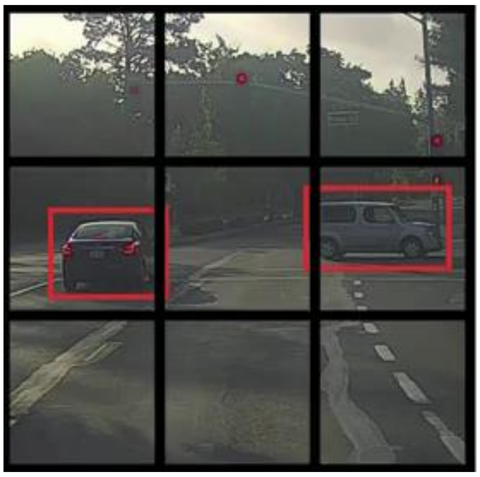
.

## 3.2. WORKING OF YOLO

First, an image is captured, and then the YOLO algorithm is performed. For example, we took a picture and split it into a grid of 3x3 matrices. Depending on the image's intricacy, we can split it into any number of grids.



**Input Image**



**The framework then divides the input image into grids (3 X 3 grid)**

.

Image classification and localization are used on each matrix. YL then predicts the bounding boxes and their corresponding class probabilities for objects (if any are discovered, of course).

Assume we've divided the image into a 3 X 3 framework, with a total of three classes into which the items should be categorized.

| | |
|---|---|
| y = | pc |
| | bx |
| | by |
| | bh |
| | bw |
| | c1 |
| | c2 |
| | c3 |

The classes should be pedestrian , car, and motorcycle , in that order. As a consequence, each framework cell's label y will be an eight-dimensional vector:

- Pc determines whether an object is present in the grid or not (it is the probability)

- If there is an object, the bounding boxes are specified by bx, by, bh, and bw, and the classes are represented by c1, c2, and c3. So, if the item is a vehicle, c2 will be 1 and c1 and c3 will be 0, and so on.

Let's say we select the first grid from the above example:

Since there is no object in this grid, pc will be zero and the y label for this grid will be:

| | |
|---|---|
| y = | 0 |
| | ? |
| | ? |
| | ? |
| | ? |
| | ? |
| | ? |
| | ? |

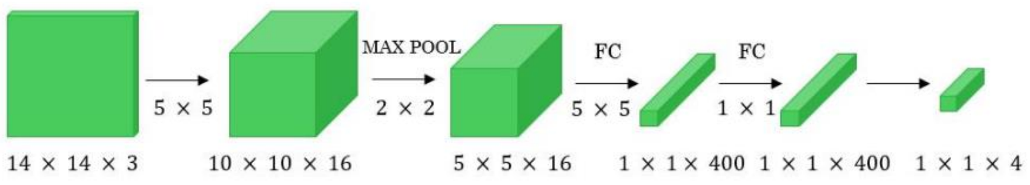
.

Here, ‘?’ means that it doesn’t matter what $b_x$, $b_y$, $b_h$, $b_w$, $c_1$, $c_2$, and $c_3$ contain as there is no object in the grid. Let’s take another grid in which we have a car ($c_2 = 1$):

We need to know how YOLO detects whether or not there is an object in the grid before we can generate the y label for this grid. Because the previous image has two items (two vehicles), YOLO will pick the mid-point of these two things and assign it to the grid that contains the midpoint of these objects. The automobile's y label in the centre left grid will be:

$$y = \begin{array}{|c|}\hline 1 \\ \hline bx \\ \hline by \\ \hline bh \\ \hline bw \\ \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline \end{array}$$

set to 1 since there is an item in this grid. bx, by, bh, and bw will be computed in relation to the grid cell in question. Because vehicle is the second class, c2 = 1 and c1 and c3 = 0. As a consequence, for each of the nine grids, we'll get an eight-dimensional output vector. The finished dimensions will be 3 X 3 X 8.



.

To train our model, we will use both forward and backward propagation. We send a picture to the model during the testing phase and conduct forward propagation until we have an output y. I've presented this using a 3 X 3 grid here to keep things easy, but in most real-world settings, we use larger grids (perhaps 19 X 19).

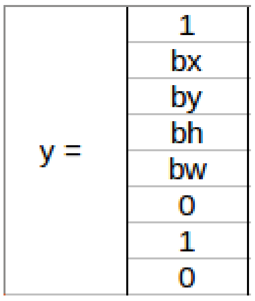
Even if an object spans many grids, it will only be allocated to the one in which its midpoint is found. By increasing the number of grids, we can limit the probability of numerous items showing in the same grid cell (19 X 19, for example).

.

## 3.3. ENCODING THE BOUNDING BOXES

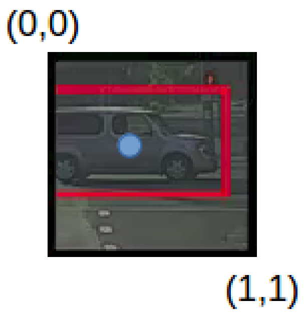
As previously stated, bx, by, bh, and bw are all calculated relative to the grid cell in question. Let's look at an example to better comprehend this notion. Consider the grid in the center-right corner, which has a car:

As a result, bx, by, bh, and bw will only be calculated in relation to this grid. For this grid, the y label will be:



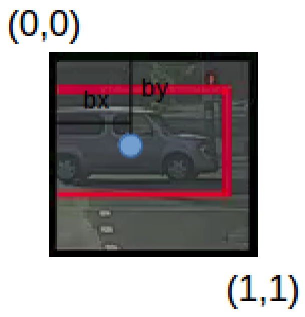
| | |
|---|---|
| | 1 |
| | bx |
| | by |
| | bh |
| y = | bw |
| | 0 |
| | 1 |
| | 0 |

pc = one c2 = 1 because there is an object in this grid and it is an automobile. Let's look at how to choose bx, by, bh, and bw. The coordinates assigned to all grids in YOLO are:



The x and y coordinates of the object's midpoint in relation to this grid are bx, by. It will be (approximately) bx = 0.4 and by = 0.3 in this case:

.



bh is the ratio of the bounding box's (red box in the example above) height to the height of the corresponding grid cell, which is roughly 0.9 in our case. As a result, bh = 0.9. The width of the bounding box divided by the width of the grid cell equals bw. As a result, bw = 0.5. (approximately). For this grid, the y label will be:

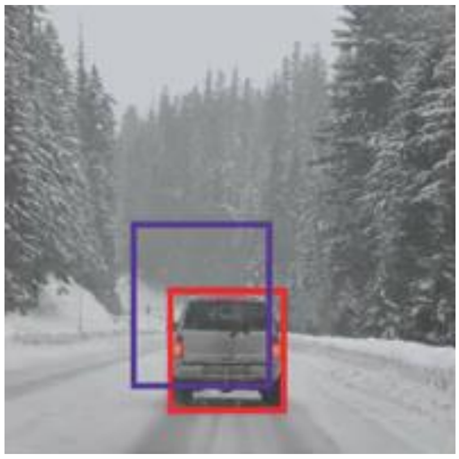
| | |
|---|---|
| y = | 1 |
| | 0.4 |
| | 0.3 |
| | 0.9 |
| | 0.5 |
| | 0 |
| | 1 |
| | 0 |

Because the midpoint will always be within the grid, bx and by will always be between 0 and 1. bh and bw, on the other hand, can be greater than one if the bounding box's dimensions exceed the grid's.

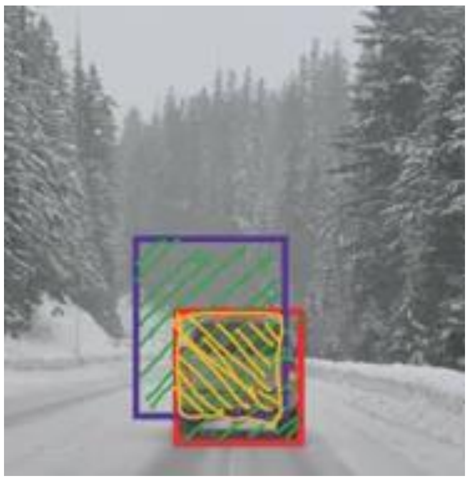
.

## 3.4. INTERSECTION OVER UNION AND NON-MAX SUPPRESSION

Here's some food for thought: how can we tell if the expected bounding box is producing a good (or bad) result? This is when the concept of Intersection over Union comes into play. It computes the intersection of the actual bounding box and the predicted bounding box over their union. Consider the following actual and expected bounding boxes for a car:



The red box represents the actual bounding box, whereas the blue box represents the predicted bounding box. How can we tell if it's a good or bad prediction? The area of the intersection over union of these two boxes will be calculated using IoU (Intersection over Union). This will be the subject of:



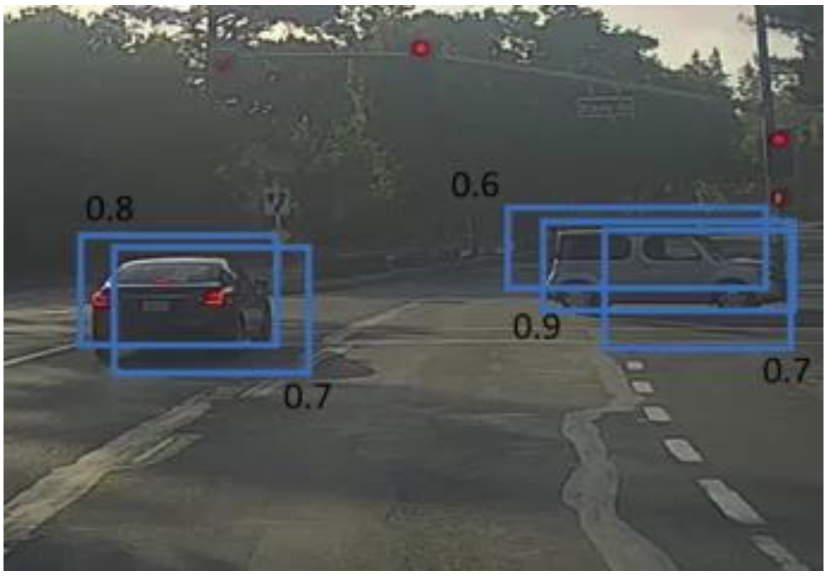
.

IoU = Area of the union / Area of the intersection, i.e.

IoU = Yellow box area / Green box area

We can state that the forecast is good enough if IoU is bigger than 0.5. We chose 0.5 as an arbitrary threshold, but it can be modified to fit your individual problem. Intuitively, the higher the threshold, the more accurate the forecasts become.Non-Max Suppression is another approach that can dramatically boost YOLO's output.

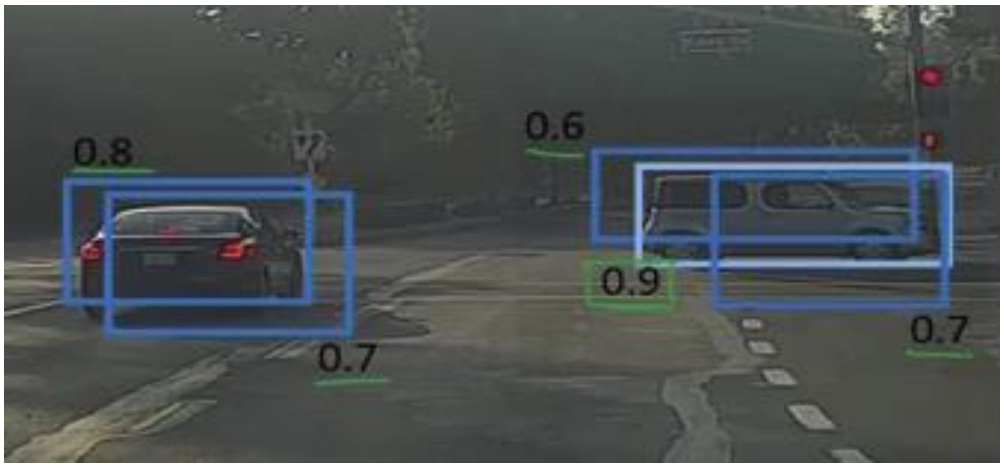
One of the most prevalent issues with object detection algorithms is that they may detect an object several times rather than simply once. Consider the following illustration:



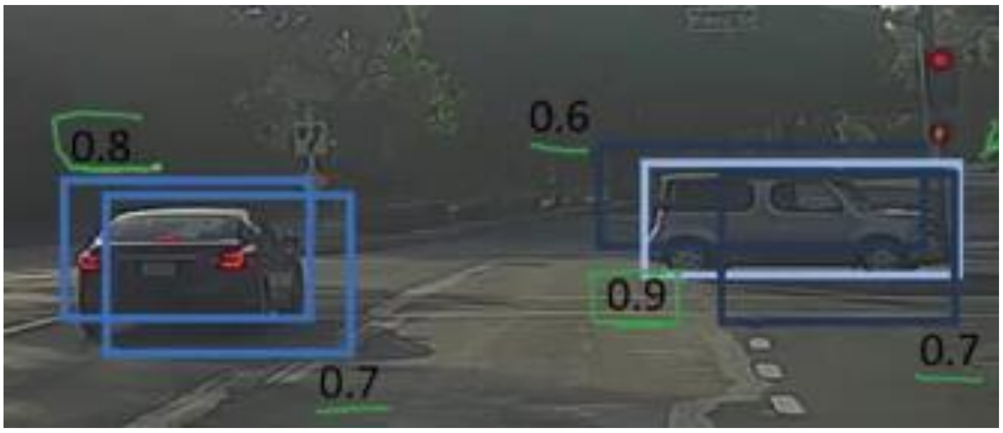
The automobiles are identified multiple times in this scene. This is cleaned up using the Non-Max Suppression approach, which results in only one detection per item. Let's have a look at how this strategy works.

1. It looks at the probabilities associated with each detection and chooses the one with the highest probability. Because 0.9 is the highest probability in the above image, the box with 0.9 probability will be chosen first:

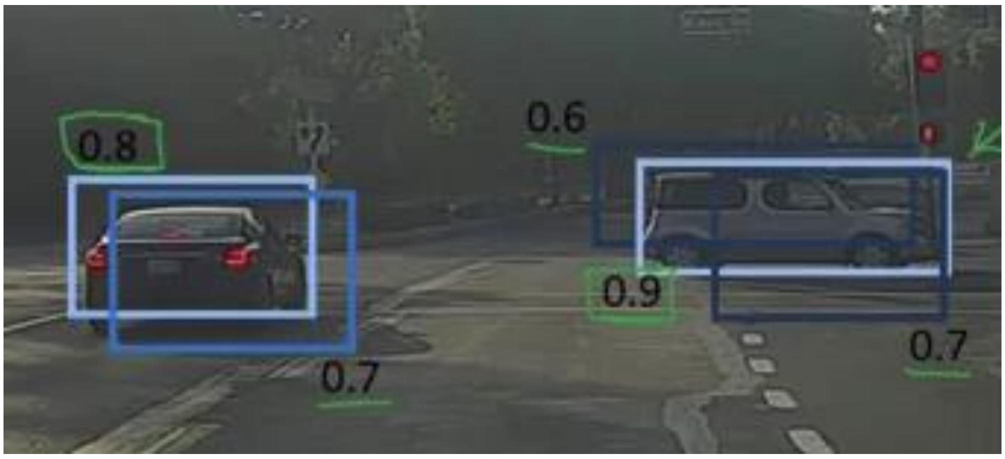
.



2. Now, it appears at all of the different packing containers withinside the image. The packing containers that have excessive IOU with the modern-day field are suppressed. So, the packing containers with 0.6 and 0.7 possibilities could be suppressed in our example:

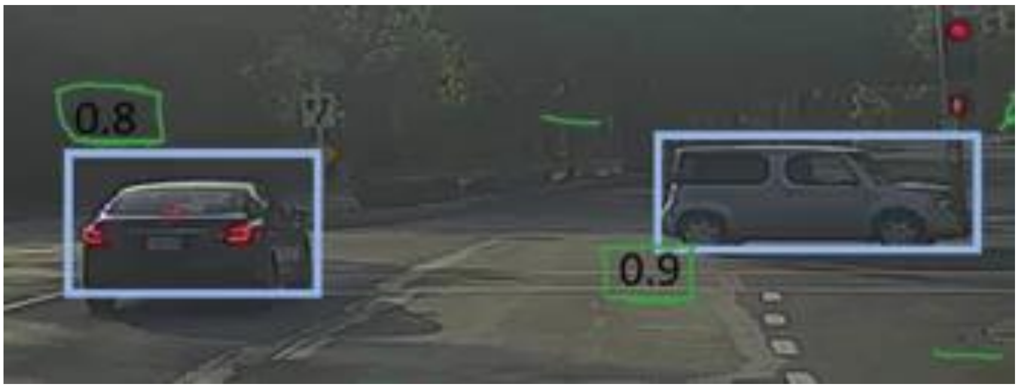


3. After the containers were suppressed, it selects the subsequent container from all of the containers with the very best probability, that's 0.eight in our case:



.

4. We repeat those steps till all of the packing containers have both been decided on or compressed and we get the very last bounding packing containers:



This is the essence of Non-Max Suppression. We take the boxes with the highest likelihood and eliminate the boxes with lower probabilities.

Steps that occur in the Non-Max suppression method in this section:

STEP 1:- All boxes with probability less than or equal to a predetermined threshold are discarded (say, 0.5)
STEP 2 :- For the remaining boxes, use the following formula:
1.Choose the box with the highest probability as your output forecast.
2.Any other box with an IoU larger than the threshold should be discarded together with the output box from the previous phase.

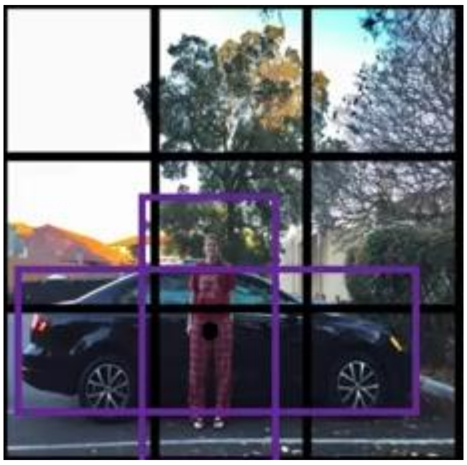
STEP 3 :- It Should be repeated until all of the boxes have been either selected as the output prediction or rejected.

.

## 3.5. ANCHOR BOXES

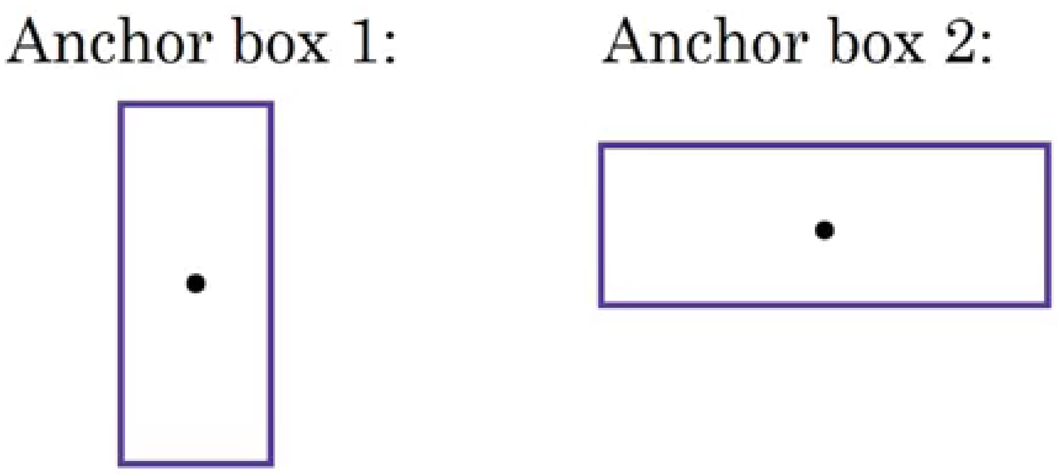
As we've seen, each grid can only recognise one item. What if a single grid has a large number of objects? In reality, this is often the case. This leads us to the concept of anchor boxes. Consider the image below, which has been split into three-by-three grids:

Remember how we put an item on a grid? We took the object's middle and allocated it to the relevant grid depending on its position. In the above example, the midpoint of both objects is located on the same grid. The actual bounding boxes for the items will be as follows:

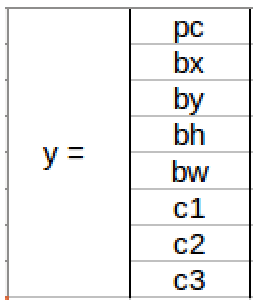


.

Only one of the crates may be introduced to us, both for the automobile or for the person. However, we is probably capable of output each containers if we use anchor containers! What's the first-rate manner to head approximately it? First, opportunity shapes referred to as anchor containers or anchor container shapes are pre-defined. Instead of getting one output for every grid, we are able to now have . The wide variety of anchor containers can continually be increased.



This is how the y label for YOLO without anchor boxes looks like:



| | |
|---|---|
| y = | pc |
| | bx |
| | by |
| | bh |
| | bw |
| | c1 |
| | c2 |
| | c3 |

.

If we have 2 anchor boxes then the y will be :

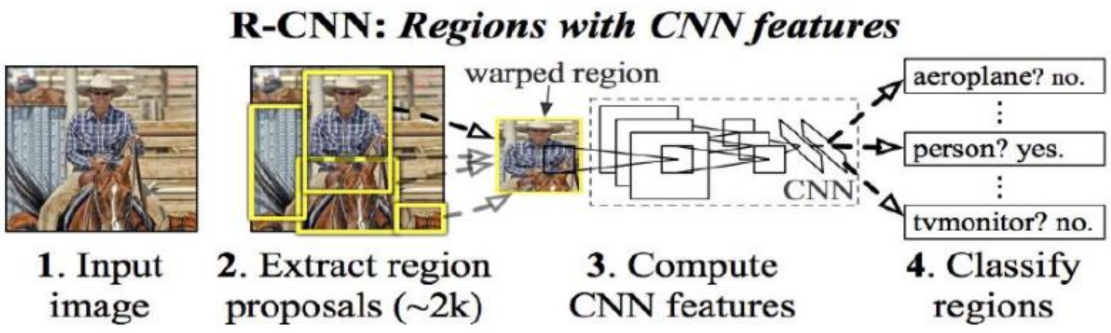
| | |
|---|---|
| y = | pc |
| | bx |
| | by |
| | bh |
| | bw |
| | c1 |
| | c2 |
| | c3 |
| | pc |
| | bx |
| | by |
| | bh |
| | bw |
| | c1 |
| | c2 |
| | c3 |

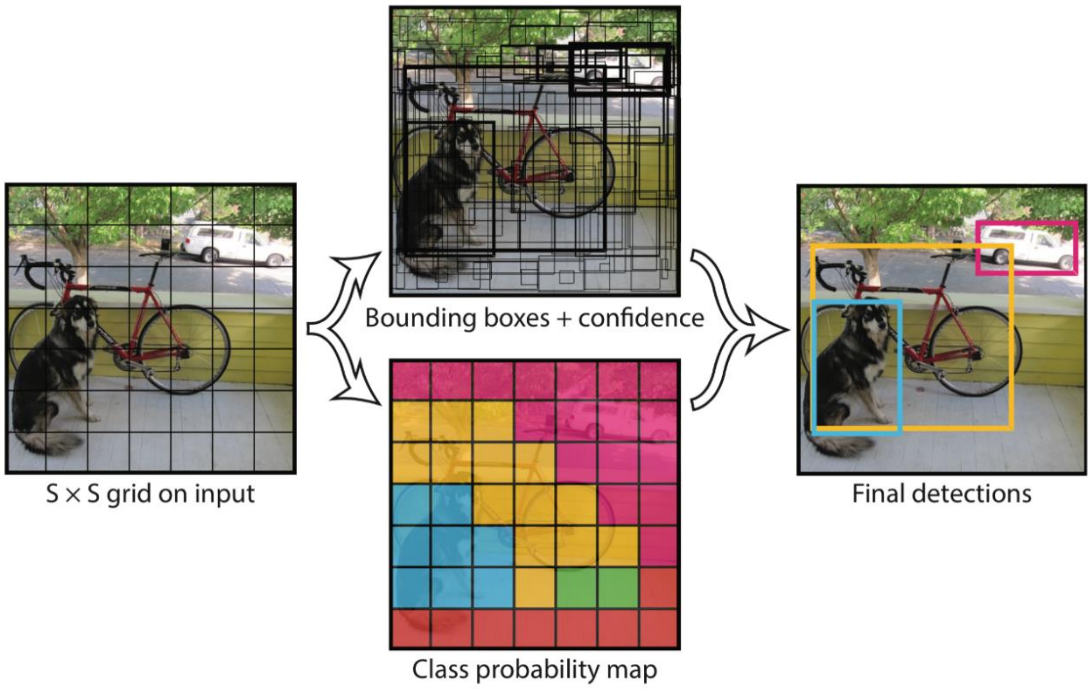
Anchor box 1 owns the first eight rows, while anchor box 2 owns the last eight. The anchor boxes are assigned to the objects depending on the shape of the bounding boxes and the shape of the anchor box. Because the geometry of anchor box 1 resembles that of a person's bounding box, the latter will be assigned to it, while the car will be assigned to anchor box 2. Instead of 3 X 3 X 8 (with a 3 X 3 grid and 3 classes), the output will be 3 X 3 X 16. (since we are using 2 anchors).

As a result, based on the number of anchors, we can detect two or more objects for each grid. Let's put all of the ideas we've discussed so far together and incorporate them into the YOLO framework.

.

**Combination of anchor boxes, non-max suppression and intersection over union which forms the YOLO framework :-**





.

## 3.6. APPLICATION OF OBJECT DETECTION

Object detection is making inroads into a wide range of businesses, with applications ranging from personal security to workplace efficiency. Object detection is used in a wide range of image processing applications, including image retrieval, security, surveillance, automated vehicle systems, and machine research. In the subject of object detection, there are still significant challenges.

1. **Automated cctv surveillance :-**

Surveillance is an important part of security and monitoring. Continuous advancements in computer vision innovation are required to better various programmed surveillance systems. However, their viability is governed by a variety of conditions, and they are not completely dependable. In both discovery and follow-up activities, this investigation looked into the ability of an automated surveillance system to reduce the CCTV administrator's outstanding task.

2. **Person Detection :-**

Person recognition is a required and crucial task in any intelligent video surveillance system, as it provides crucial information for semantic understanding of video recordings. Because of the potential for increasing security frameworks, it is a noticeable addition to automotive applications. Person detection is the task of using computer vision frameworks to locate and track individuals. Person detection is the challenge of locating all examples of individuals in a photograph, and it has traditionally been accomplished by scanning all sections of the photograph at all possible scales and contrasting a small region at each scale with known layouts or examples of individuals.

3. **Vehicle Detection :-**

One of the most crucial aspects of our daily lives is vehicle detection. Vehicle detection is becoming increasingly vital as the globe moves quicker and the number of cars on the road continues to rise. We can detect the number plate of a speeding car or a car

.

that has been involved in an accident utilising the Vehicle Detection technology. This also contributes to society's security and the reduction of crime.

4. **People Counting :-**

Object detection can also be used to count the number of people. It's used to analyse business operations or group measurements at festivals. As people move out of the picture faster, these will become increasingly problematic .

5. **Self Driving Cars :-**

Self-driving automobiles are another unique application of object detecting technology. A self-driving car can only safely travel a roadway if it can detect all items on the road, such as pedestrians, other cars, and road signs, and decide what action to take.

6. **Object extraction from an image or video :-**

Object Extraction is a problem that is strongly related to the segmentation process. Image segmentation is the division of an image into subparts depending on attributes such as colour, intensity, and so on. Object extraction's fundamental purpose is to transform an image's representation into something more meaningful. We must first segment the entire image in order to extract an object from it.

7. **Activity Recognition :-**

The goal of activity recognition is to recognise one or more agents' actions and goals based on a sequence of observations of the agents' behaviours and the surrounding environment. Due to its strength in providing individualised support for a variety of applications and its connections to a variety of fields, this research subject has piqued the interest of various computer science communities.

Thus there are numerous application of object detection system where it can be used . Object detection also can be used in Military applications and for medical applications for detecting the disease .

.

## 3.7. IMPLEMENTING YOLO IN PYTHON FOR OBJECT DETECTION

First we need to implement all the libraries needed for object detection .

```
import cv2

import numpy as np
```

Next we need to import our dataset which is in weights form and the yolo configuration file .

```
net = cv2.dnn.readNet(' configuration file', 'weights file')
```

After that if we have to run the detection on GPU and if our GPU is CUDA enabled than we will add

```
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)

net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
```

Now for we need the the names file which will contain the names of the objects that are trained and convolve into the weights file and after that we have to read that file for making the program know that the specific name given to the objects

```
with open("coco.names", "r") as f:

    classes = f.read().splitlines()
```

The Below line will now start detection depending upon the used inputs if we pass the ‘0’ in the function then it will start detection from our camera and if we want to detect the images/videos we need to give the full path with name and extension of file .

.

```
cap = cv2.VideoCapture('User Input')
```

Now we have to set the colors for labelling the image so for that we have to extend the cv2.FONT_HERSHEY_PLAIN for giving the colors to labels for the object detected and we have to give the random colors to each object so the code will be :-

```
font = cv2.FONT_HERSHEY_PLAIN
 colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

Now the main thing will happen we have to read the image / frame from the given input and to find the height , width and center point of the image and to ouput the bounding boxes random colors for each object and the confidence of detection and to avoid the duplication of the bounding boxes we will use the non max suppression and we will create the anchor box to get detected multiple objects.

```
while True:
        _, img = cap.read()
        height, width, _ = img.shape
  blob = cv2.dnn.blobFromImage(img, 1 / 255, (416, 416), (0, 0, 0), swapRB=True,
crop=False)
        net.setInput(blob)
        output_layers_names = net.getUnconnectedOutLayersNames()
        layerOutputs = net.forward(output_layers_names)

        boxes = []
        confidences = []
        class_ids = []
        for output in layerOutputs:
            for detection in output:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > 0.5:
                    center_x = int(detection[0] * width)
                    center_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)
                    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)
```

.

```
# it will remove the duplicate detections in our detection
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.2, 0.4)

        if len(indexes) > 0:
            for i in indexes.flatten():
                x, y, w, h = boxes[i]
                label = str(classes[class_ids[i]])
                confidence = str(round(confidences[i], 2))
                color = colors[i]
cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
cv2.putText(img, label + " " + confidence, (x, y + 20), font, 2, (255, 0, 0), 2)
```

Now we have to output the window using the opencv

```
cv2.imshow('window_name', img)
```

.

# CHAPTER 4

# CREATE AND TRAIN CUSTOM MODEL

.

## 4.1 CREATING DATASET

**i):** To begin, open a command prompt or terminal and use the following command to clone the AIGuysCode/OIDv4 ToolKit github repo into your own system in your desired location.

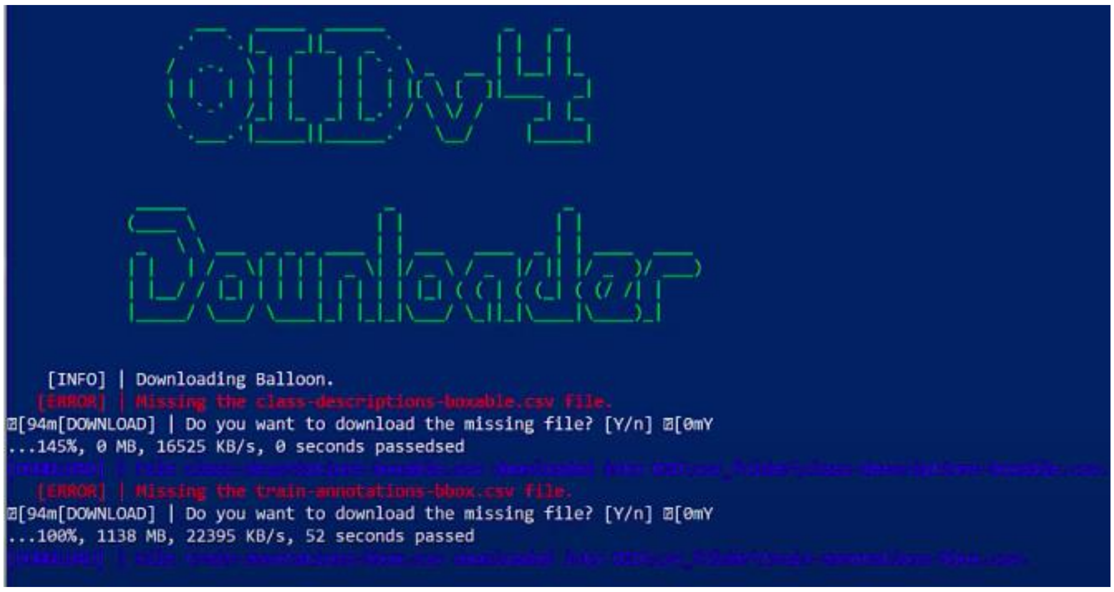
**ii):** From your terminal, go inside the folder 'OIDv4_ToolKit' and type in

pip install -r requirement.txt

**iii):** Now that you've completed all of the prerequisites, go to the Open Images Dataset website and select the image classes that interest you. Assume you're interested in three classes: Balloon, Person, and Dog. Return to your terminal and, within the same 'OIDv4 ToolKit' folder, enter in: to obtain pictures from these three classes.

python main.py downloader  --classes Balloon Person Dog --type_csv train --limit 200

**iv):**  When the download begins, you will be asked twice if you want to download some missing files. Both times, press y.

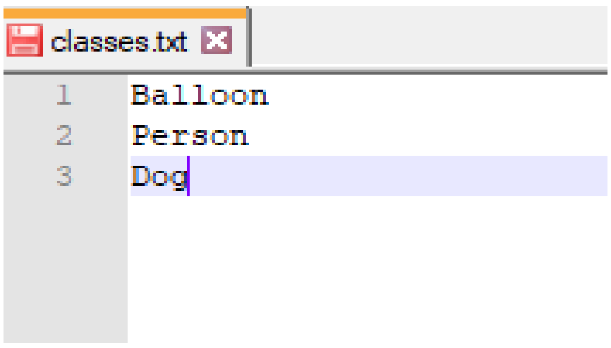


```
    [INFO] | Downloading Balloon.
   [ERROR] | Missing the class-descriptions-boxable.csv file.
⍰[94m[DOWNLOAD] | Do you want to download the missing file? [Y/n] ⍰[0mY
...145%, 0 MB, 16525 KB/s, 0 seconds passedsed
[illegible]
   [ERROR] | Missing the train-annotations-bbox.csv file.
⍰[94m[DOWNLOAD] | Do you want to download the missing file? [Y/n] ⍰[0mY
...100%, 1138 MB, 22395 KB/s, 52 seconds passed
[illegible]
```

**v):** Repeat Step 3 once again but this time change 'train' to 'validation' in the command, and also reduce the amount of images to 40 here (20% of train data). This is done to evaluate your custom object detector after training.

**vi):** Within the root OIDv4_ToolKit folder, open the file 'classes.txt' and edit it to have the classes you just downloaded, one per line.

.


```
Balloon
Person
Dog
```

**vii):** Now run this command to convert the labels generated by OIDv4_Toolkit into YOLOv4 labels:

```
python convert_annotations.py
```

This should convert both train and validation dataset labels which can now be used by darknet to properly train our custom object detector.

**viii):** Remove the old 'Label' folder in the train and validation folders which contains the non-YOLOv4 formatted labels.

.

## 4.2. TRAIN A YOLO MODEL

**Step 1: Enabling GPU within your notebook**

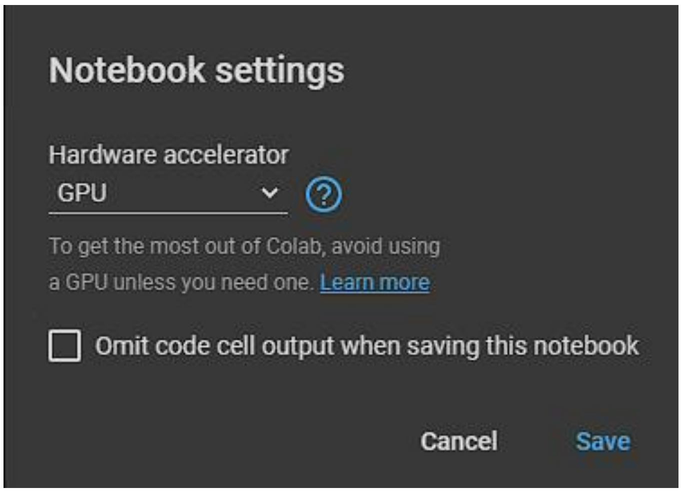
You will want to enable GPU acceleration within your Colab notebook so that your YOLOv4 system will be able to process detections over 100 times faster than CPU.

Steps:

i) Click Edit at top left of your notebook

ii) Click **Notebook Settings** within dropdown

iii) Under 'Hardware Accelerator' select GPU and then hit Save



Notebook settings

Hardware accelerator: GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

☐ Omit code cell output when saving this notebook

**Step 2: Cloning and Building Darknet**

The following cells will clone darknet from AlexeyAB's famous repository, adjust the Makefile to enable OPENCV and GPU for darknet and then build darknet.

```
!git clone https://github.com/AlexeyAB/darknet
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!make
```

.

**Step 3: Moving Your Custom Datasets Into Your Cloud VM**

So now that you have your datasets properly formatted to be used for training and validation, we need to move them into this cloud VM so that when it comes the time we can actually train and validate our model.

```
!cp /mydrive/yolov4/obj.zip ../
!cp /mydrive/yolov4/test.zip ../
!unzip ../obj.zip -d data/
!unzip ../test.zip -d data/
```

**Step 4: Configuring Files for Training**

This step involves properly configuring your custom .cfg, obj.data, obj.names, train.txt and test.txt files. It is important to configure all these files with extreme caution as typos or small errors can cause major problems with your custom training.

```
!cp cfg/yolov4-custom.cfg /mydrive/yolov4/yolov4-obj.cfg
```

Now you need to edit the .cfg to fit your needs based on your object detector. Open it up in a code or text editor to do so. If you downloaded cfg to google drive you can use the built in Text Editor by going to your google drive and double clicking on yolov4-obj.cfg and then clicking on the Open with drop down and selectin Text Editor.

I recommend having batch = 64 and subdivisions = 16 for ultimate results. If you run into any issues then up subdivisions to 32.

Make the rest of the changes to the cfg based on how many classes you are training your detector on.

How to Configure Your Variables:

width = 416

height = 416

max_batches = (# of classes) * 2000 (but no less than 6000 so if you are training for 1, 2, or 3 classes it will be 6000, however detector for 5 classes would have max_batches=10000)

steps = (80% of max_batches), (90% of max_batches) (so if your max_batches = 10000, then steps = 8000, 9000)

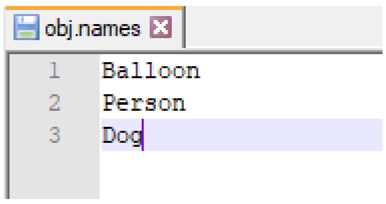
filters = (# of classes + 5) * 3 (so if you are training for one class then your filters = 18, but if you are training for 4 classes then your filters = 27)

```
!cp /mydrive/yolov4/yolov4-obj.cfg ./cfg
```

.

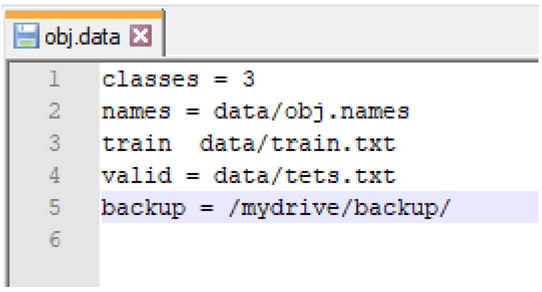
### i) obj.names and obj.data

Create a new file within a code or text editor called obj.names where you will have one class name per line in the same order as your classes.txt from the dataset generation step.



```
Balloon
Person
Dog
```

You will also create a obj.data file and fill it in like this (change your number of classes accordingly, as well as your backup location)



```
classes = 3
names = data/obj.names
train  data/train.txt
valid = data/tets.txt
backup = /mydrive/backup/
```

### ii) Generating train.txt and test.txt

The last configuration files needed before we can begin to train our custom detector are the train.txt and test.txt files which hold the relative paths to all our training images and validation images.

```
!cp /mydrive/yolov4/generate_train.py ./
!cp /mydrive/yolov4/generate_test.py ./
!python generate_train.py
!python generate_test.py
```

**Step 5: Download pre-trained weights for the convolutional layers.**

This step downloads the weights for the convolutional layers of the YOLOv4 network. By using these weights it helps your custom object detector to be way more accurate and not have to train as long

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```

.

**Step 6: Train Your Custom Object Detector**

The time has finally come! You have made it to the moment of truth! You are now ready to train your custom YOLOv4 object detector on whatever crazy classes you have decided on. So run the following command. (-dont_show flag stops chart from popping up since Colab Notebook can't open images on the spot, -map flag overlays mean average precision on chart to see how accuracy of your model is, only add map flag if you have a validation dataset)

```
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137
```

.

# CHAPTER 5

# COMPILING CUDA

**(Compute Unified Device Architecture)**

.

## 5.1. CUDA(Compute Unified Device Architecture)

Nvidia's CUDA (Compute Unified Device Architecture) platform and application programming interface (API) paradigm is a parallel computing platform and API model. It enables software developers and engineers to use a CUDA-enabled graphics processing unit (GPU) for general-purpose processing (a technique known as GPGPU) (general-purpose computing on graphics processing units).
The CUDA platform is a software layer that allows compute kernels to have direct access to the GPU's virtual instruction set and parallel computational elements.
We did the cuda compilation to run our object detection with better speed and provide better accuracy .

## 5.2. PREREQUISITES OF CUDA COMPILATION

Step 1:-Download and install CUDA_v10.1.243_426.00 - https://developer.nvidia.com/cuda-toolkit-archive and cuDNN_v7.6.5.32 - https://developer.nvidia.com/rdp/cudnn-archive .
Step 2 :- Download and install CmakeGUI - https://cmake.org/download/
Step 3 :- Download and install Visual Studio Community Edition from https://visualstudio.microsoft.com/downloads/ . Install with **Desktop Development for C++** option.
Step 4 :- Download OpenCV source from https://opencv.org/releases/ .
Step 5 :- Download OpenCV contrib from https://github.com/opencv/opencv_contrib/ . Make sure the version matches with OpenCV.
Step 6 :- Extract OpenCV and OpenCV contrib zip files.
Step 7 :- Make an empty build folder.

.

## 5.3. BUILDING OPENCV WITH CMAKE GUI

Step 1:- Open CMake GUI and browse for OpenCV source folder.

Step 2 :- Browse for make folder that we created above.

Step 3 :- Click on Configure and select X64 platform and hit Finish.

Step 4 :- New options will appear in CMake in red color. Tick these checkboxes there: WITH_CUDA, OPENCV_DNN_CUDA, ENABLE_FAST_MATH.

Step 5 :- On the same window, go to OPENCV_EXTRA_MODULES_PATH and browse for OpenCV contrib directory and point to the modules subfolder.

Step 6:- Hit Configure again. You will see new options in red color. Tick CUDA_FAST_MATH checkbox. From CUDA_ARCH_BIN property, remove any compute architecture that your model of Nvidia GPU does not support. You can find a list of compatible compute architectures for your model of GPU :- https://en.wikipedia.org/wiki/CUDA .

Step 7 :- Hit Configure and then Generate**.**

## 5.4. MAKING OPENCV WITH VISUAL STUDIO

1. Go to **build** folder and open **OpenCV.sln** file with Visual Studio.
2. Once opened, change **Debug** to **Release** from the top.
3. On the panel at the right-hand side, expand **Cmake Targets**.
4. Right-click on **ALL_BUILD** and click on **build**.
5. Once done, right-click on **Install** and click on **build**.

Thus we done with our cuda compilation now we can run the object detection on GPU which will speed our execution .

.

# CHAPTER 6

# LIBRARIES USED TO DEVELOP THE OBJECT DETECTION SYSTEM

.

## 6.1. EEL FRAMEWORK

Eel is a Python library that allows you to create simple Electron-like offline HTML/JS GUI programmes with full access to Python's capabilities and libraries. Eel was created to make building short and simple GUI apps easier. If you're comfortable with Python and web programming, you'll probably want to start with this example, which generates random file names from a folder (something that is impossible from a browser).

There are various ways to create GUI apps in Python, but if you want to use HTML/JS (for example, to use jQueryUI or Bootstrap), you'll have to write a lot of boilerplate code to communicate from the Client (Javascript) to the Server (Python) side.Cefpython is the Python equivalent to Electron that I'm aware of. It's a little heavy for what I intended.

There are many options for building GUI applications in Python, but if you want to use HTML/JS, you usually have to write a lot of boilerplate code to read from the client to communicate with the waiter. Page (Python). The closest Python to Electron is cefpython. What I want is too difficult.

Eel is not as versatile as Electron or cefpython, and may not be suitable for building complete applications like Atom, but it is great for building equivalent GUIs for small utility scripts that you use on your computer. ), but many of the best display libraries use Javascript .

Thus Eel is good for small applications and easier to implement than other frameworks like electron for which we have to write lots of boilerplate code for just starting our application . And thus eel is better in implementing the applications for frontend .

.

**Installing Eel :-**

```
pip install eel[jinja2]
```

**Directory structure of eel :**

```
.
├── main.py
└── web
    └── canvas.html
    └── css
        └── styles.css
    └── img
        └── logo.png
```

**Starting the app :**

Suppose you place all the frontend files in an exceedingly directory known as web, as well as your begin page main.html, then the app is started like this :-

```
import eel

eel.init('web')

eel.start('main.html')
```

Thus we have created our frontend using eel framework and due to support of html and css templating support of the eel we have successfully developed aur frontend

.

## 6.2. OPENCV

OpenCV is the large open-supply library for the pc vision, device learning, and photo processing and now it performs a main function in real-time operation which may be very vital in today’s systems. By the use of it, one could system snapshots and motion pictures to discover objects, faces, or maybe handwriting of a human. When incorporated with diverse libraries, consisting of Numpy, python is able to process the OpenCV array shape for analysis. To Identify photo samples and its diverse capabilities we use vector areas and carry out mathematical operations on those capabilities. The first OpenCV model turned into 1.0. OpenCV is launched below a BSD license and therefore it’s unfastened for each educational and business use. It has C++, C, Python and Java interfaces and helps Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the primary cognizance turned into real-time programs for computational efficiency. All matters are written in optimized C/C++ to take advantage of multi-middle processing.

OpenCV-Python is a library of Python bindings designed to clear up pc imaginative and prescient problems. Python is a standard reason programming language commenced via the means of Guido van Rossum that have become very famous very quickly, specifically due to its simplicity and code readability. It permits the programmer to specific thoughts in fewer traces of code without decreasing readability.. OpenCV-Python is a Python wrapper for the unique OpenCV C++ implementation. Application of opencv library : Street view image stitching, Automated inspection and surveillance, Robot and driver-less car navigation and control, Medical image analysis ,Video/image search and retrieval.

**Installation of opencv :**

```
pip install cv2
```

.

## 6.3. NUMPY

NumPy is the essential bundle for medical computing in Python. It is a Python library that offers a multidimensional array object, diverse derived objects (consisting of masked arrays and matrices), and an collection of exercises for instant operations on arrays, consisting of mathematical, logical, form manipulation, sorting, selecting, I/O, discrete Fourier transforms, fundamental linear algebra, fundamental statistical operations, random simulation and lots more.

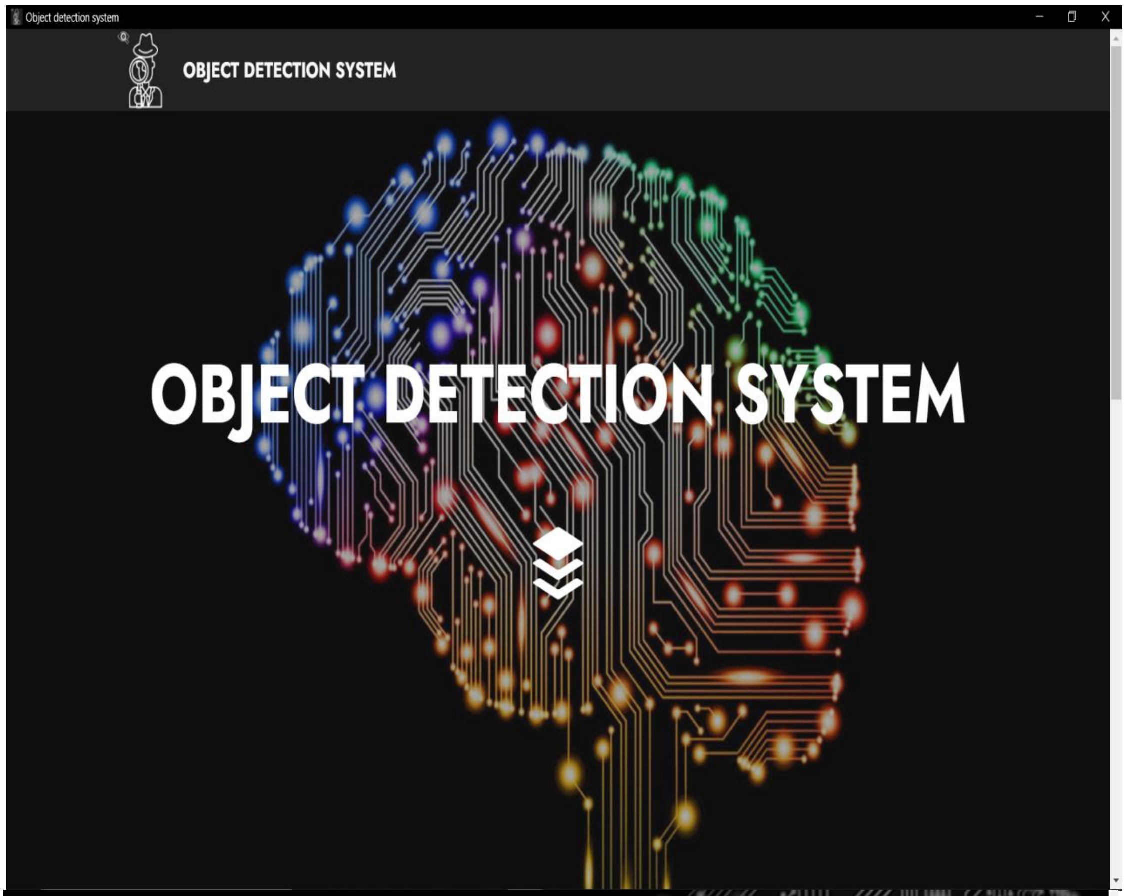
It affords an effective N-dimensional array object ,sophisticated (broadcasting) functions,gear for integrating C/C++ and Fortran code ,beneficial linear algebra, Fourier transform, and random quantity capabilities. Besides its apparent clinical uses, NumPy also can be used as a green multi-dimensional field of common data. Arbitrary data-sorts may be defined. This permits NumPy to seamlessly and in a timely fashion combine with an extensive type of database.

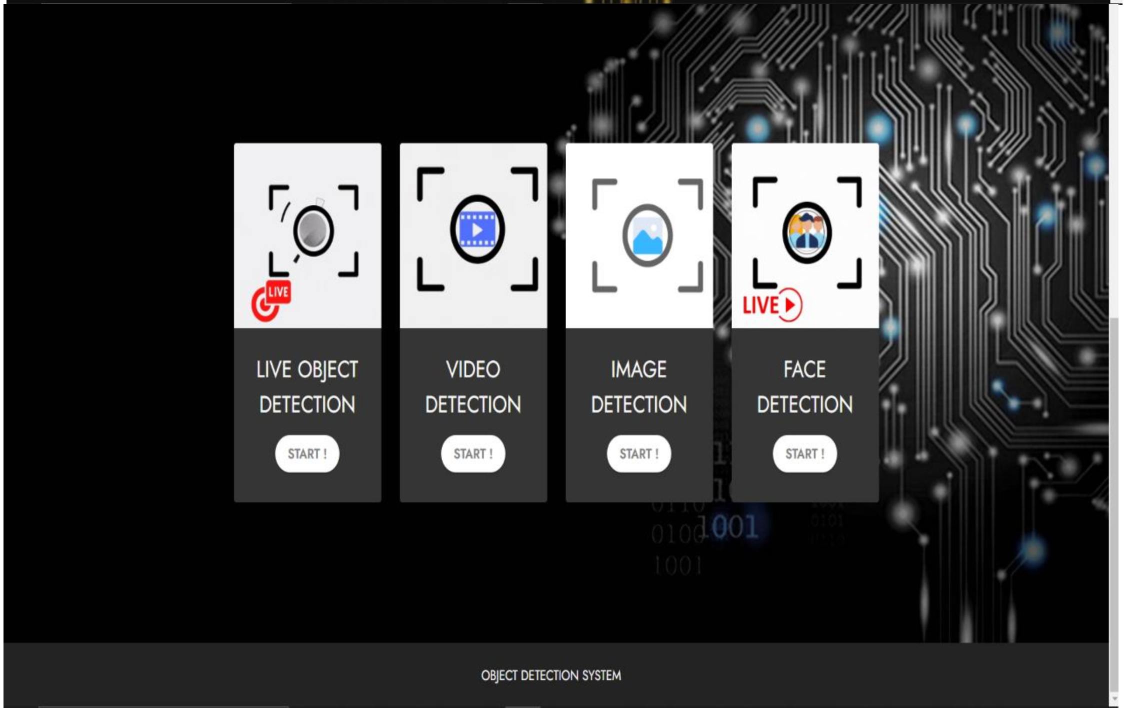
In Python we've lists that serve the reason for arrays, however they're sluggish to process. NumPy aims to offer an array item that is as much as 50x quicker than conventional Python lists. The array item in NumPy is known as ndarray, it presents numerous helping features that make running with ndarray very easy. Arrays are very regularly utilized in facts science, wherein velocity and sources are very important.

**Install numpy :**

```
pip install numpy
```

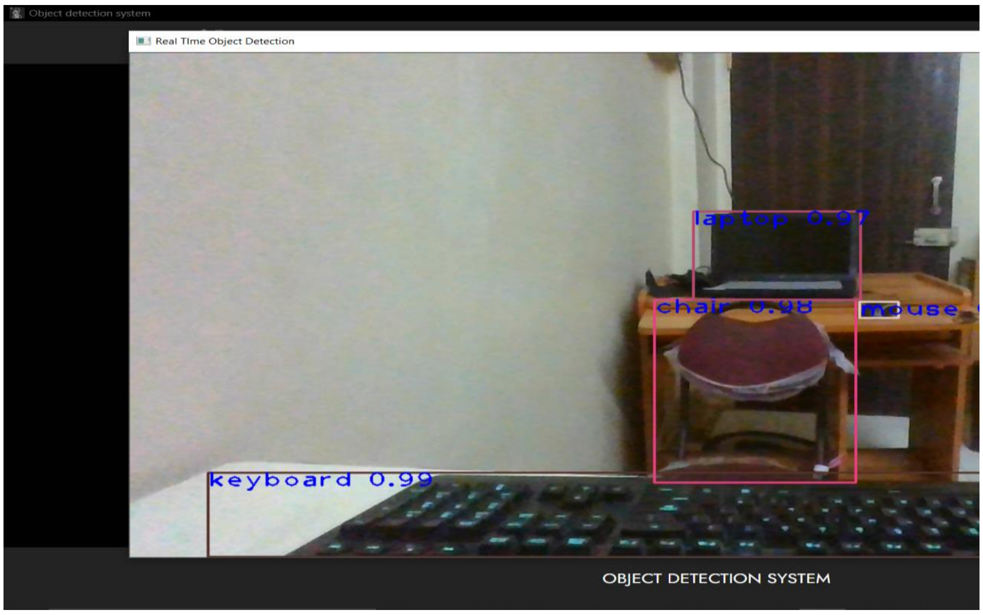
## RESULTS


Object detection system
OBJECT DETECTION SYSTEM
OBJECT DETECTION SYSTEM

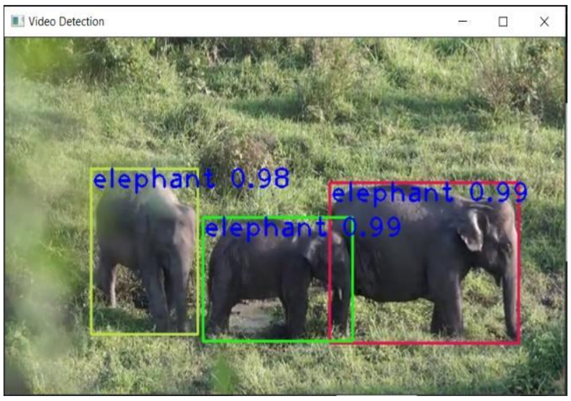

LIVE
LIVE OBJECT DETECTION
START !
VIDEO DETECTION
START !
IMAGE DETECTION
START !
LIVE
FACE DETECTION
START !
OBJECT DETECTION SYSTEM

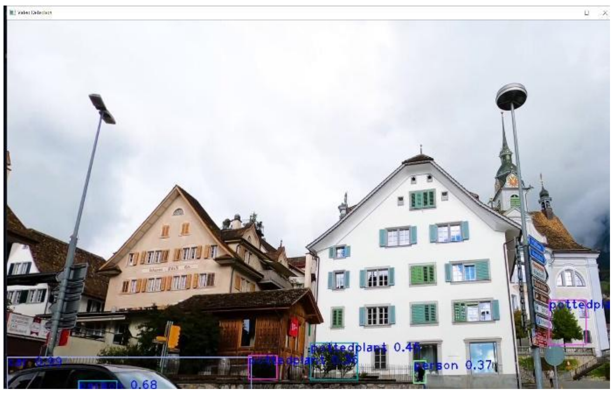
.

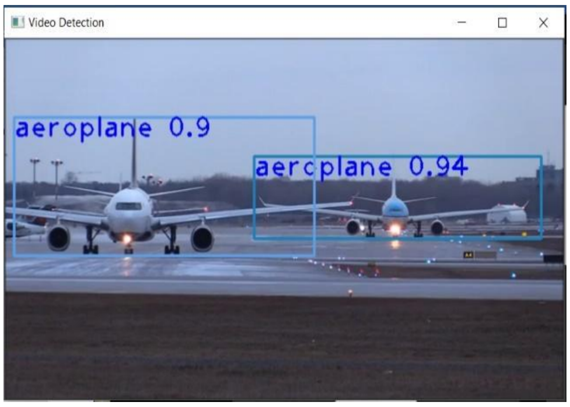
## LIVE OBJECT DETECTION :-

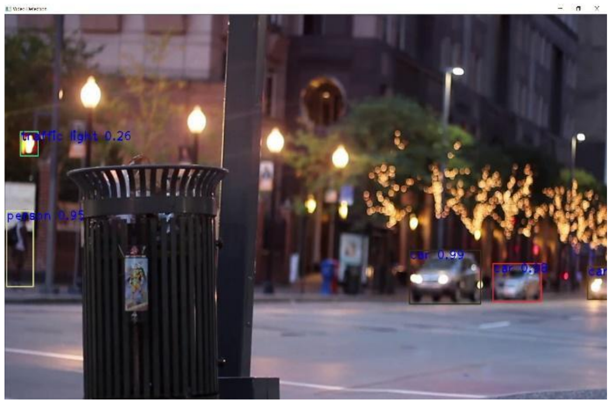


## VIDEO DETECTION :-



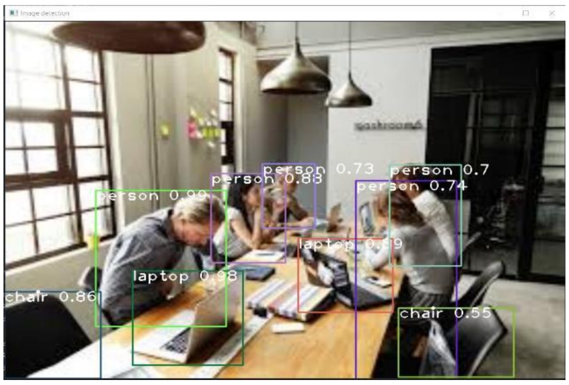


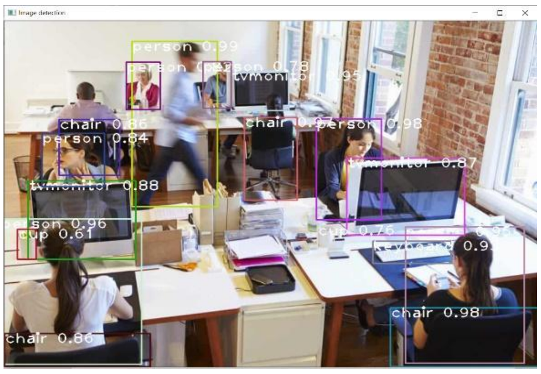




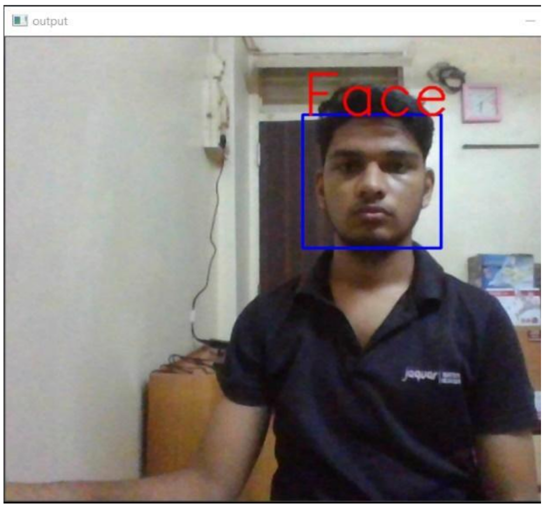
.

## IMAGE DETECTTION :-





## FACE DETECION :-



.

## CONCLUSION

Object detection based on deep learning has developed into a hotspot in recent years, which the authors achieved by combining 2 things: deep learning object recognition and OpenCV and efficient video material with OpenCV.We created a project to make object detection with deep learning and OpenCV more efficient. The goal was to create an algorithm that could be used for 3 common tasks, namely object detection and face detection.Object detection algorithms act as a mixture of image classification and object location. It takes the specified image as input and generates the bounding boxes according to the number of objects in the image, with the category tag attached to the top of each bounding box. Projects the bounding box stage in terms of position, height, and width.

## FUTURE SCOPE

Object detection is a computer vision technique that enables us to identify and locate objects in an image or video. With this type of identification and location, objects in a scene can be counted and their exact location determined and tracked, all accurate object detection is widely used in computer vision tasks such as image annotation, activity detection, face detection, face detection, and co-segmentation of video objects . It has the potential to free people from basic tasks that machines can do more efficiently and effectively. Object recognition can be used in many image processing areas, including image retrieval, security, observation, vehicle computing systems, and machine research. Critical difficulties remain in the area of object recognition. Track the ball during a soccer game, track the movement of a cricket bat, or track a person on video.

.

# REFERENCES

- Andrew NG , Convolutional Neural Networks : Deeplearning.ai , Coursera.org , 2017
- Joseph Redmon , Ali Farhadi , Yolo , prejeddie.com
- Adrians , Computer vision , Deep Learning and Opencv , pyimagesearch.com
- M . Haroon Shakeel , Building Opencv with Cuda , https://haroonshakeel.medium.com/build-opencv-4-4-0-with-cuda-gpu-support-on-windows-10-without-tears-aa85d470bcd0, 2020
- D Pavan ,REAL TIME OBJECT DETECTION USING DEEP LEARNING. http://ece.anits.edu.in/2019-20%20BE%20Project%20REPORTS/CHPS%20 (2019).
- Sainagesh Veeravali , Object Detection and identification , https://www.researchgate.net/publication/337464355_OBJECT_DETECTION_AND_IDENTIFICATION_A_Project_Report
- Grace Karimi , Introduction to yolo algorithm and object detection ,https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/#:~:text=YOLO%20is%20an%20abbreviation%20for,probabilities%20of%20the%20detected%20images. , 2021
- A.I. Guy , YOLOv4_Training_Tutorial , https://colab.research.google.com/drive/1_GdoqCJWXsChrOiY8sZMr_zbr_fH-0Fg?usp=sharing (2019, December 31).
- Pulkit sharma , https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/ , 2018
- Chriss Knot , https://pypi.org/project/Eel/ , 2020
- Gary Bradsky , https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html#:~:text=OpenCV%20was%20started%20at%20Intel,the%202005%20DARPA%20Grand%20Challenge , 2000

.

- Ahmad, Tanvir & ma, Yinglong & Yahya, Muhammad . Object Detection through Modified YOLO Neural Network. Scientific Programming. 10.1155/2020/8403262. (2020)