A Project-1 Report on

OBJECT DETECTION USING YOLO

Submitted to

The Department of Computer Science and Engineering

In partial fulfillment of the academic requirements of

Jawaharlal Nehru Technological University

For

The award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

(2017 – 2021)

By

A. Pratyusha	17311A05D1
M.A Javeed Ather	17311A05F2
T. Bhargavi	17311A05F5

Under the Guidance of

Dr. Preethi Jeevan

(Assistant Professor)

Sreenidhi Institute of Science and Technology
(An Autonomous Institution)
Yamnampet, Ghatkesar, R.R. District, Hyderabad – 501301

Department of Computer Science and Engineering

Sreenidhi Institute of Science and Technology

CERTIFICATE

This is to certify that this Group Project report on “ **OBJECT DETECTION USING YOLO**”, submitted by A.Pratyusha (17311A05D1), M.A Javeed Ather (17311A05F2), T.Bhargavi (17311A05F5), in the year 2021 in partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as part of their **Project-1 during Fourth Year First Semester**, under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

Internal guide	**Project Co-ordinator**	**Head of Department**
Mr. Nallapu Venkata Subba Reddy	Dr. Preethi Jeevan	Dr. Aruna Varanasi
Associate Professor	Assistant Professor	Professor & HOD
Department of CSE	Department of CSE	Department of CSE

External Examiner

Date: -

DECLARATION

We, **A. Pratyusha (17311A05D1), M. A Javeed Ather (17311A05F2), T. Bhargavi (17311A05F5),** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR,** studying IVrd year First Semester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Group project work, titled **"OBJECT DETECTION USING YOLO"** is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **Dr. T. Ch. Siva Reddy,** who most ably runs the institution and has had the major hand in enabling me to do my project.

I profoundly thank Dr. **ARUNA VARANASI**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank my internal guide **Mr. Venkata Subba Reddy** for his/her technical guidance, constant encouragement, and support in carrying out my project at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

A. PRATYUSHA	**17311A05D1**
M.A JAVEED ATHER	**17311A05F2**
T. BHARGAVI	**17311A05F5**

INDEX	**Page Number**
List of Figures	i
Abstract	ii
1. INTRODUCTION	2
1.1 Motivation	2
1.2 Problem Definition	2
1.3 Objective of Project	3
2. LITERATURE SURVEY	4
2.1 Related Work	4
2.1 Existing System	4
2.2 Proposed System	5
3. SYSTEM ANALYSIS	7
3.1 Functional Requirement Specifications	7
3.2 Software Requirements	7
3.3 Hardware Requirements	8
3.4 Performance Requirements	8
4. SYSTEM DESIGN	9
4.1 Architecture of Proposed System	9
4.2 Modules	12
4.2 UML Diagrams	12

4.2.1 Use Case Diagram	13
4.2.2 Class Diagram	14
4.2.3 Sequence Diagram	15
4.2.4 Activity Diagram	15
5. IMPLEMENTATION AND RESULTS	17
5.1 Methodology	17
5.2 Implementation	22
5.3 Results	34
6. TESTING	39
6.1 Types of Testing	39
6.2 List of Test Cases	39
7. CONCLUSION AND FUTURE SCOPE	40
BIBLIOGRAPHY	41

Appendix-A: Python Modules Appendix-B: Unified Modeling Language	42

LIST OF FIGURES

S.No	Fig No	Title of Figure	Page No
1	2.2	Existing System Image of R-CNN	4
2	4.1	Architecture Diagram	9
3	4.1.1	A Yolo Implementation	11
4	4.1.2	Algorithm dealing with object detection	11
5	4.3.1	Use Case Diagram	13
6	4.3.2	Class Diagram	14
7	4.3.3	Sequence Diagram	15
8	4.3.4	Activity Diagram	16
9	5.1.1	Dataset	17
10	5.1.1.1	Dataset Image 1	18
11	5.1.1.2	Dataset Image 2	18
12	5.2.1	Identifying and importing the required libraries	23
13	5.2.2	Code Snippet for Cloning the dataset	23
14	5.2.3	Darknet for Python	24
15	5.2.4	Yolov4 on test image	24
16	5.2.5	Labels to the objects and persons in image	25
17	5.2.6	Helper Functions	25
18	5.2.7	Helper Functions(2)	26
19	5.2.8	Yolov4 on webcam images code snippet	27
20	5.2.9	Yolov4 on webcam images	27
21	5.2.10	Yolov4 on webcam videos	28
22	5.2.11	Yolo Algorithm workflow	29
23	5.2.12	Example image with 3*3 grids	30
24	5.2.13	Image with the bounding probabilities	31
25	5.2.14	Image with the bounding probabilities2	31
26	5.2.15	Image with non max suppression before and after	32
27	5.2.16	An example image for anchor box	32
28	5.2.17	Anchor box prediction values	33
29	5.3.1	Accuracy of the models implemented	35

30	5.3.2	Input with expected output table	36
31	5.3.3	Test cases for the input provided	36
32	5.3.4	Output of the image showing the labels for person and phone with accuracy	37
33	5.3.5	Output of the image showing the labels for cup with accuracy	37
34	5.3.6	Output of the image showing the labels for spoon with accuracy	38
35	5.3.7	Output of the image showing the labels for person and sports ball with accuracy	38

Abstract

In the field of image processing, the required tool kit should support the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems. Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Object detection does the work of combining these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term “object recognition“, they often mean “object detection“.

1.INTRODUCTION

1.1 Motivation

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc.

As modern technical solutions turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems are formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers.

Object detection algorithms can be divided into the traditional methods which used the technique of sliding window where the window of specific size moves through the entire image and the deep learning methods that includes YOLO algorithm. You Only Look Once: Unified, Real-Time object detection, detecting objects using a regression algorithm. To get higher accuracy and good predictions they have proposed the YOLO algorithm in this project.

Understanding Object Detection Based on CNN Family and YOLO, they generally explained about the object detection families like CNN, R-CNN and compared their efficiency and introduced the YOLO algorithm to increase the efficiency. Learning to Localize Objects with Structured Output Regression, this project is about Object Localization. In this, they had used the Bounding box method for localization of the objects to overcome the drawbacks of the sliding window method.

1.2 Problem Definition

The primary motive for the project is to to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Object detection does the work of combining these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term “object recognition “, they often mean “object detection “. It may be challenging for beginners to distinguish between different related computer vision tasks.

1.3 Objective

The YOLO algorithm is faster as compared to other classification algorithms. The YOLO algorithm makes localization errors, but it predicts less false positives in the background. These algorithms are not tested with degraded images, i.e. they are trained with academic data sets, including COCO. but this are not well tested with randomly captured data sets. The main issues of images captured in the real scene are:

1) Due to the instability of the camera, the captured images may be blurred.

2) The images can also not be clear enough because the object can be obstructed.

3) The images may have poor quality as a result of bad weather, overexposure, or low resolution.

2. LITERATURE REVIEW

2.1 Related Work

Firstly, the dataset for our project was taken from YOLOv3 in particular, YOLO trained on the COCO dataset. The COCO dataset consists of 80 labels, including, but not limited to. Further, we have referred to technical papers available to study the existing system so that we can build a project which is better and more accurate than the existing system. This would be discussed in depth in the following section.

2.2 Existing System

Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition.

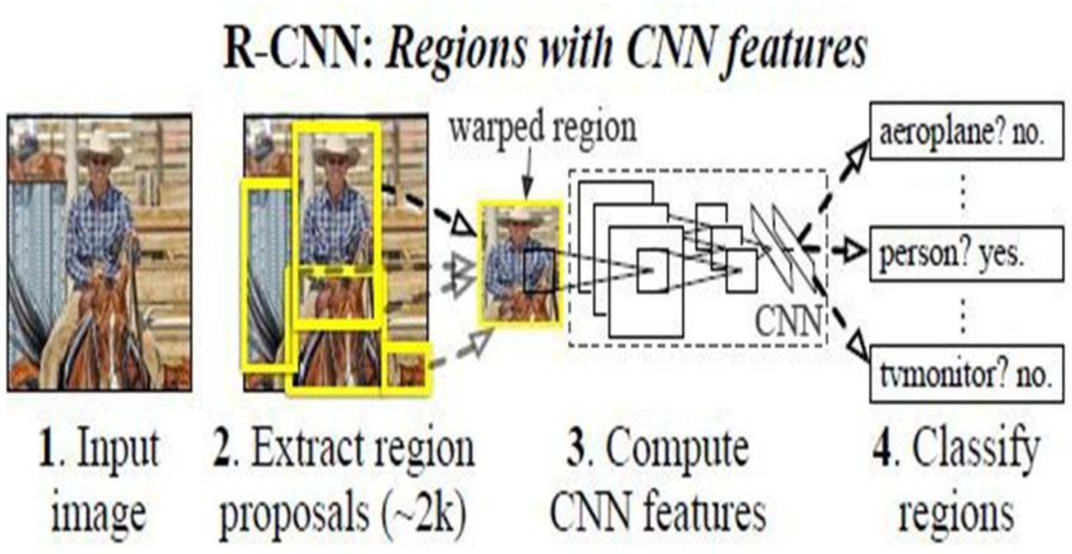


Fig 2.2 Existing System Image of R-CNN

We can think of an object detector as a combination of a **object locator** and an **object recognizer**.

In traditional computer vision approaches, a sliding window was used to look for objects at different locations and scales. Because this was such an expensive operation, the aspect ratio of the object was usually assumed to be fixed.

Early Deep Learning based object detection algorithms like the R-CNN and Fast R-CNN used a method called Selective Search to narrow down the number of bounding boxes that the algorithm had to test.

Another approach called Overfeat involved scanning the image at multiple scales using sliding windows-like mechanisms done convolutionally.

This was followed by Faster R-CNN that used a Region Proposal Network (RPN) for identifying bounding boxes that needed to be tested. By clever design, the features extracted for recognizing objects were also used by the RPN for proposing potential bounding boxes thus saving a lot of computation.

Selective Search:

1. Generate the initial sub-segmentation, we generate many candidate regions

2. Use the greedy algorithm to recursively combine similar regions into larger ones

3. Use generated regions to produce the final candidate region proposals

The existing system consists of accuracy rates which are not satisfactory. The use would lead to struggles to detect small objects, decrease the grid value in the bounding box in the absence of the anchor box and comparatively low recall and more localization error compared to Faster R-CNN.

2.3 Proposed System

All the previous object detection algorithms have used regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region-based algorithms which are seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

First, it divides the image into a 13×13 grid of cells. The size of these 169 cells varies depending on the size of the input. For a 416×416 input size that we used in our experiments, the cell size was 32×32. Each cell is then responsible for predicting a number of boxes in the image.

For each bounding box, the network also predicts the confidence that the bounding box actually encloses an object, and the probability of the enclosed object being a particular class. Most of these bounding boxes are eliminated because their confidence is low or because they are enclosing the same object as another bounding box with very high confidence score. This technique is called non-maximum suppression.

3. SYSTEM ANALYSIS

3.1 Functional Requirements Specifications

These are the requirements that are to be offered by the system to the end-user. All these functionalities need to be undoubtedly integrated into the system as a part of the contract. These are represented or stated in the form of input to the system, the operations are performed, and we get expected results. They are the basic requirements stated by the user which are reflected directly in the final product, far from the non-functional requirements.

- Data Collection
- Handling Missing Values
- Remove Duplicate rows
- Data Cleaning
- Data Labelling
- Data Scaling
- Finding Feature Importance
- Data Visualization
- Model Creating
- Model Training
- Hyper parameter tuning
- Model boosting and bagging.
- Model Evaluation
- Model Deployment

3.2 Software Requirements

● Programming Language	: Python
● IDE	: VS Code
● UML Design	: StarUML
● Tools	: PIP
● Cross-Platform	: OpenCv(YOLO Algorithm)
● Platform	: TensorFlow
● Database	: COCO

3.3 Hardware Requirements

● Processor	: Intel i3 and above
● RAM	: 4GB and Higher
● Hard Disk	: 20GB: Minimum

3.4 Performance requirements

Performance is measured in terms of the output provided by the application which is mostly based on accuracy. Requirement specification plays an important role in analysing the system. Only when the requirement specifications are properly given, it is possible to design a system. The requirement specification for any system can be stated as follows:

1. The system should be precise.
2. The system should be able to interface with the existing system.
3. The system should be better than the previous system.

4. SYSTEM DESIGN

4.1 Architecture of Proposed System

The architecture of the proposed system involves the input data, the trained model and the output. The user provides the feature values as input and these are processed by the trained model. Further the output is shown to the user via the interface.The architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

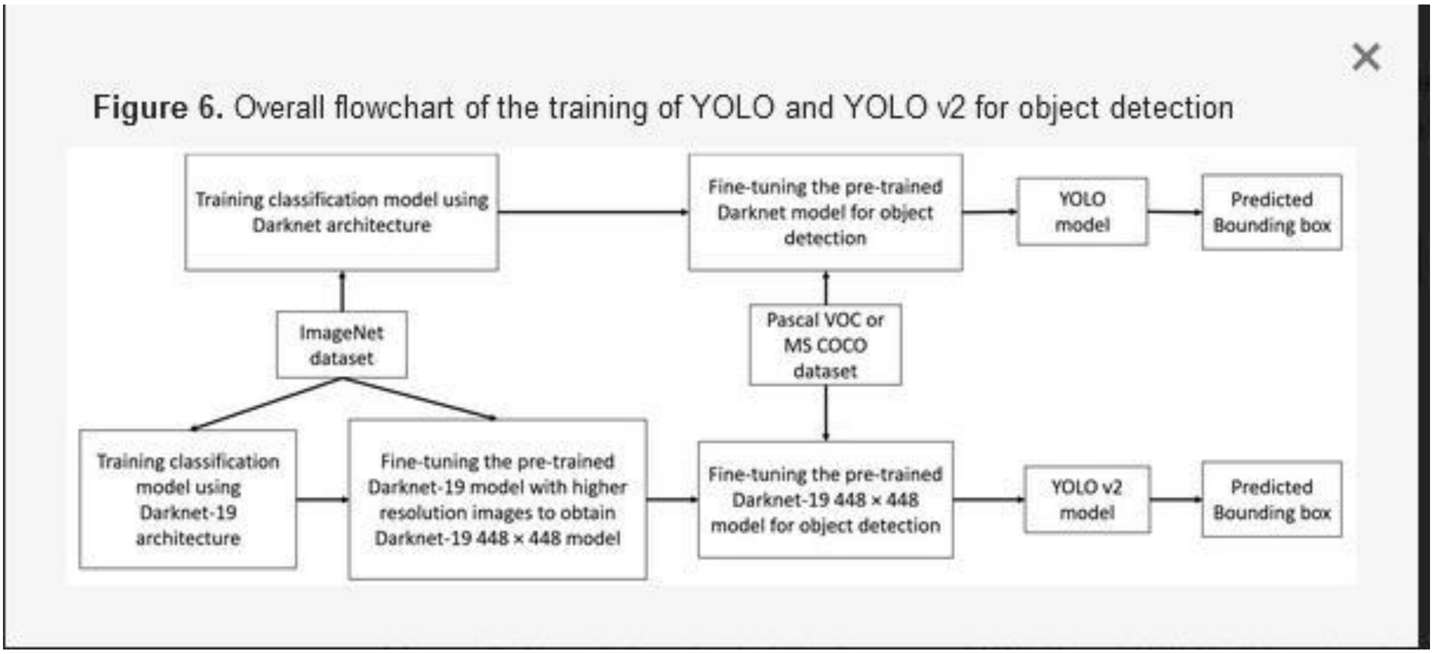


Figure 6. Overall flowchart of the training of YOLO and YOLO v2 for object detection

×
Training classification model using Darknet architecture
Fine-tuning the pre-trained Darknet model for object detection
YOLO model
Predicted Bounding box
ImageNet dataset
Pascal VOC or MS COCO dataset
Training classification model using Darknet-19 architecture
Fine-tuning the pre-trained Darknet-19 model with higher resolution images to obtain Darknet-19 448 × 448 model
Fine-tuning the pre-trained Darknet-19 448 × 448 model for object detection
YOLO v2 model
Predicted Bounding box

Fig 4.1: Architecture Diagram

First, an image is taken, and the YOLO algorithm is applied. The image is divided as grids of 3x3 matrices. We can divide the image into any number grids, depending on the complexity of the image. Once the image is divided, each grid undergoes classification and localization of the object. The objectness or the confidence score of each grid is found. If there is no proper object found in the grid, then the objectness and bounding box value of the grid will be zero or if there is found an object in the grid then the objectness will be 1 and the bounding box value will be its corresponding bounding values of the found object. The bounding box prediction is explained as follows. Also, Anchor boxes are used to increase the accuracy of object detection which is also explained below in detail.

YOLO ("You Only Look Once") is an effective real-time object recognition algorithm, first described in the seminal 2015 paper by Joseph Redmon et al. In this article we introduce the concept of object detection, the YOLO algorithm itself, and one of the algorithm's open source implementations.

Features

- First, they select regions of interest in an image.
- Second, they classify these regions using convolutional neural networks.
- Algorithms based on regression – instead of selecting interesting parts of an image,
- They predict classes and bounding boxes for the whole image **in one run of the algorithm**.
- We aim to predict a class of an object and the bounding box specifying object location.

Each bounding box can be described using four descriptors:

- ○ center of a bounding box (**bxby**)
- ○ width (**bw**)
- ○ height (**bh**)
- ○ value **c**is corresponding to a class of an object (such as: car, traffic lights, etc.).

- · Speed (45 frames per second — better than real-time)
- · Network understands generalized object representation (This allowed them to train the Network on real world images and predictions on artwork was still fairly accurate).
- · Faster version (with smaller architecture) — 155 frames per sec but is less accurate.

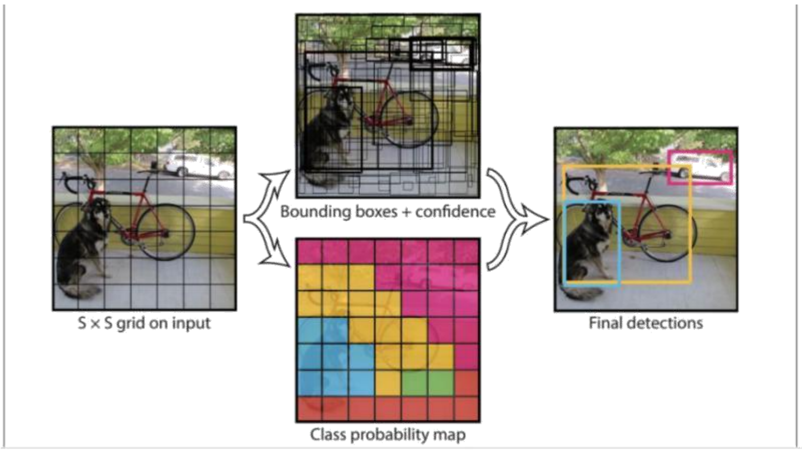


Fig 4.1.1 a YOLO v3 implementation

The algorithm deals well even with object representations.

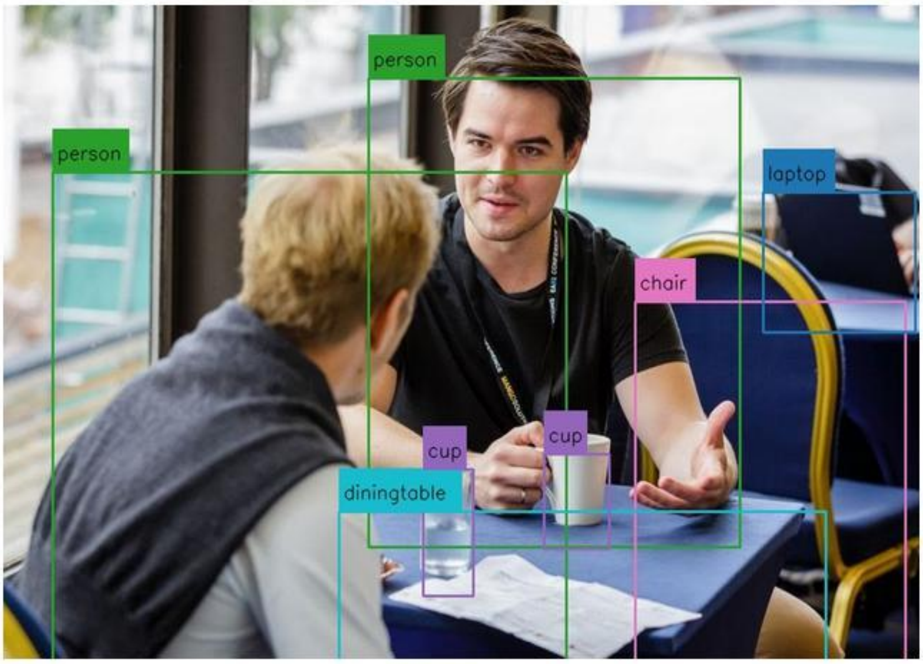


Figure 4.1.2 The algorithm deals well even with object representations.

YOLO can only detect objects belonging to the classes present in the dataset used to train the network. We will be using the official weight file for our detector. These weights have been obtained by training the network on COCO dataset, and therefore we can detect 80 object categories.

After installation, we can use a pre-trained model or build a new one from scratch.you can detect objects on your image using model pre-trained on COCO dataset.

After training our model we use the OpenCV module to open the camera in our laptop and when we place any image or anything in front of the camera then we can see that the objects detected in our image are the same as the above figure(4.2).

4.2 Modules

The modules help to understand the functional area. The project is divided into modules where each discrete unit has a particular functionality.

The modules can be used to logically separate the functionality for the project. We can normally have one or more than a single module in any project.

There are three main important modules that can be defined in our project: (a) Developer (b) Server and (3) User.

Developer: The developer plays the most important role. They are responsible for taking out tasks from the initial pre-processing of the dataset to the building of the model. The developer is also responsible for making further developments in the later stages as per the requirements of the user. The developer does the data

Server: The dataset used for our project is uploaded to the server and read by the algorithm. It is further continued to process. The final model is deployed into the server which the end user can use in for practical application.

End User: The end user also plays an important role. The model is built and changed accordingly to the requirements of them. The end user is the one who uses the final and provides feedback regarding any changes to be made.

4.3 UML Diagrams

The UML diagrams help in better visualization of any project. They help in understanding the project and also describe the purpose with easy visualization.

Some of the important uses of UML diagrams is described as follows:

It helps in understanding the behavior of the end resultant system.

We can also spot any mistakes or problems in the beginning stages itself.

It helps in communicating the proposed project effectively with no misconceptions.

It helps in getting to know the requirements and also helps to ideate from where we can implement i.e., the beginning step.

In the following section, we have discussed the different UML diagrams for our project.

4.3.1 Use Case Diagram

The use case diagram gives the details regarding the actors and their functionalities. It is basically a graphical representation. Following use case diagram consists of the following actors

1. User
2. System

Following is the use case diagram which provides information about the actors and their functionality or role:

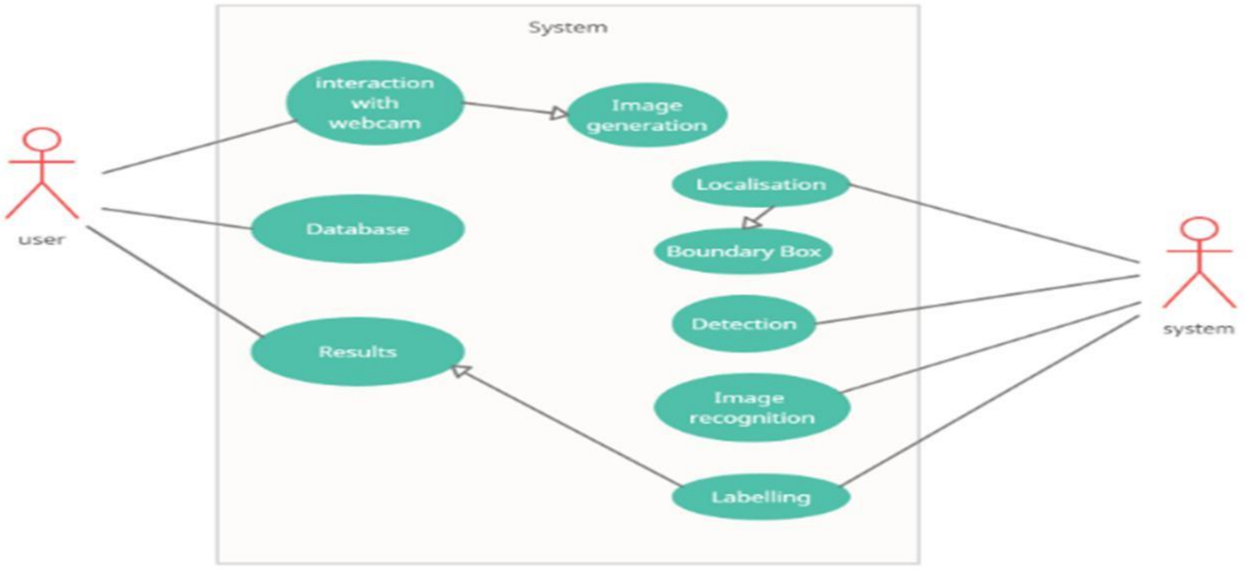


Fig 4.3.1: Use Case Diagram

4.3.2 Class Diagram

The class diagram plays a prominent role in object-oriented programming. The class diagram depicts the static structure which gives information about the classes in the system, the attributes and operations as well as the relationship amongst the various classes. The class diagram provides details about various classes in the system.

Following is the class diagram describing the different classes containing attributes and methods in our system.

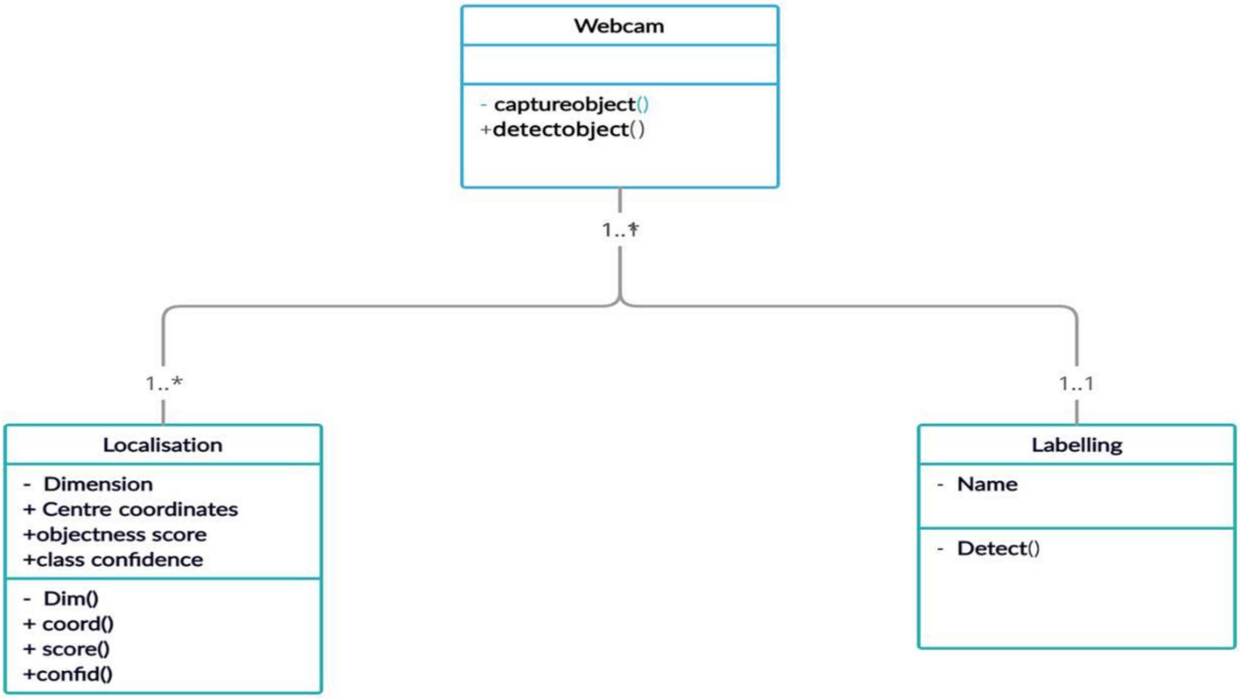


Fig 4.3.2: Class Diagram

4.3.3 Sequence Diagram

The sequence diagram in UML provides information about how the operations are carried out. In other words, they provide details on what messages are sent and when. It is an interaction diagram where details are organised according to time. It provides a dynamic view of the system i.e., the model.

Following is the sequence diagram for our model:

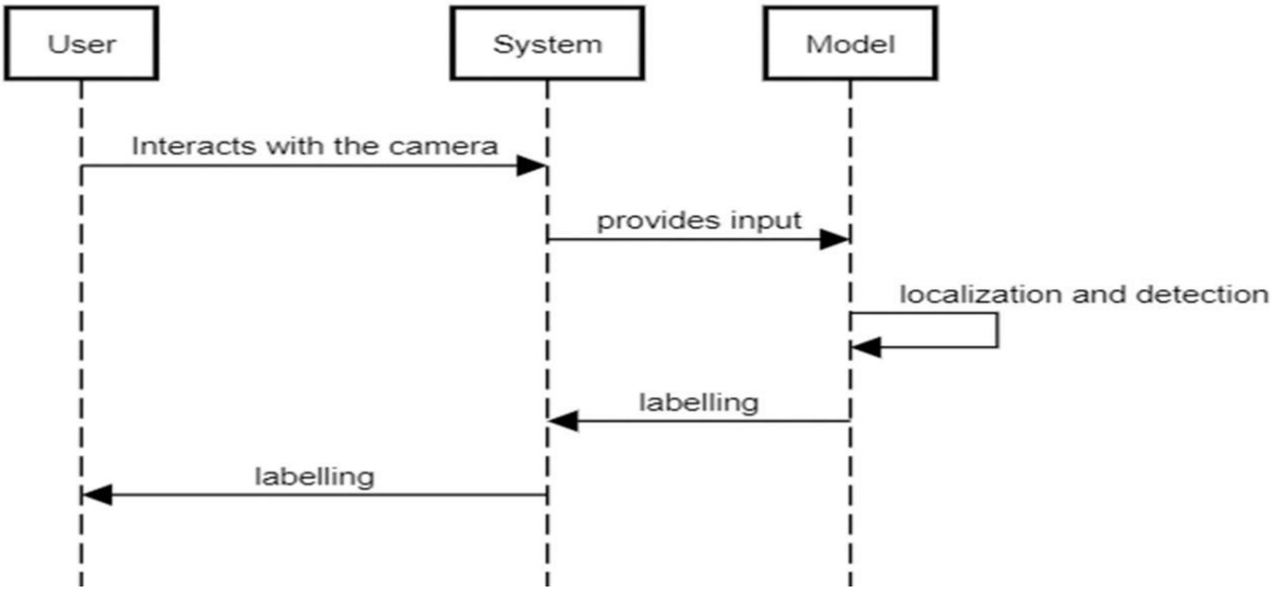


Fig 4.3.3: Sequence Diagram

4.3.4 Activity Diagram

The activity diagram is used for representing the workflow of the activities. It helps in showing the stepwise activities for a model. It shows the control flow from the beginning point until the finishing point depicting various decision paths that are present when the activity is executed.

The activity diagram consists of rounded rectangles which are used for representing actions, bars which are used for representing the start or end of the activities and a black circle depicting the start and end of the control flow.

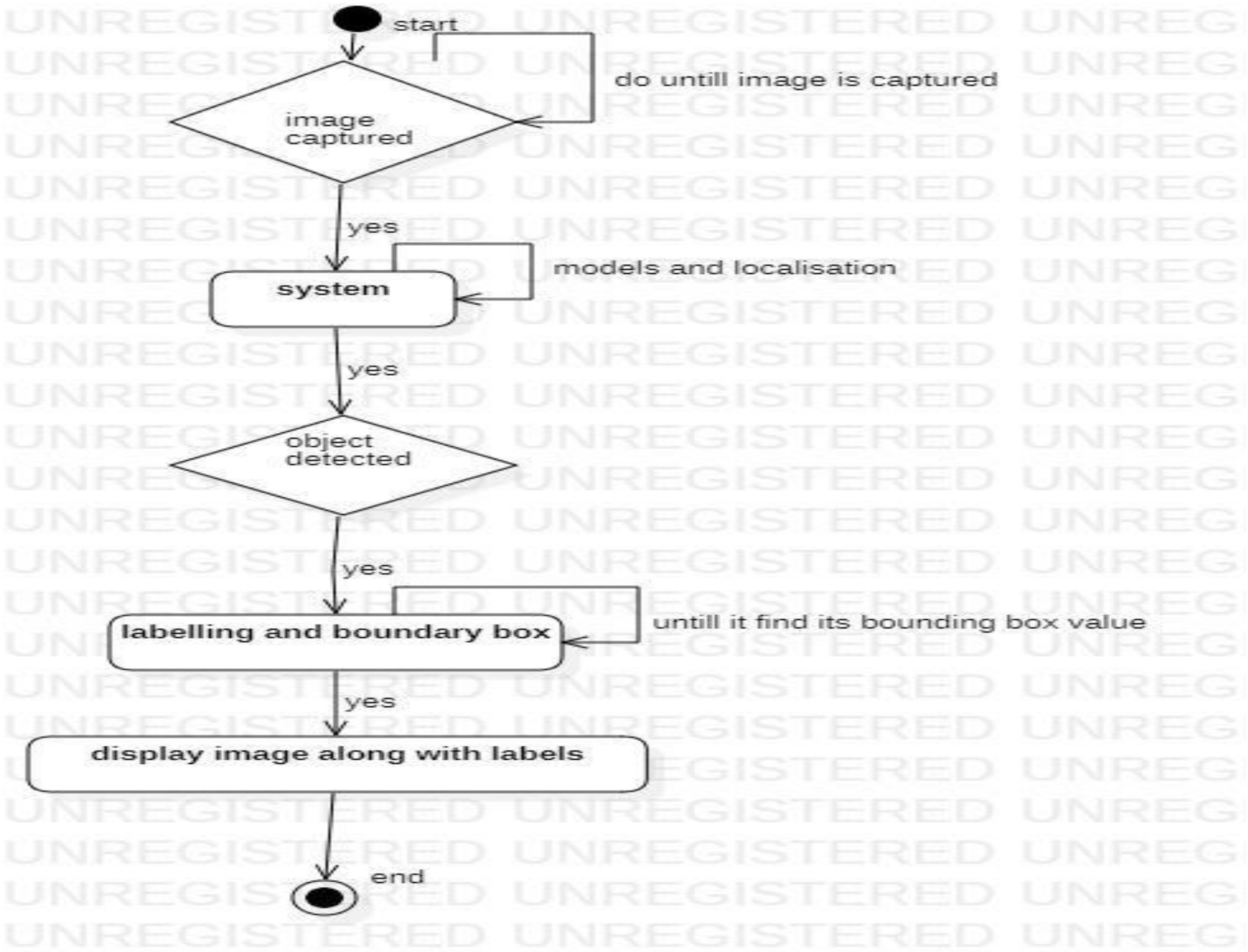

start
do untill image is captured
image captured
yes
models and localisation
system
yes
object detected
yes
untill it find its bounding box value
labelling and boundary box
yes
display image along with labels
end

Fig 4.3.4 : Activity Diagram

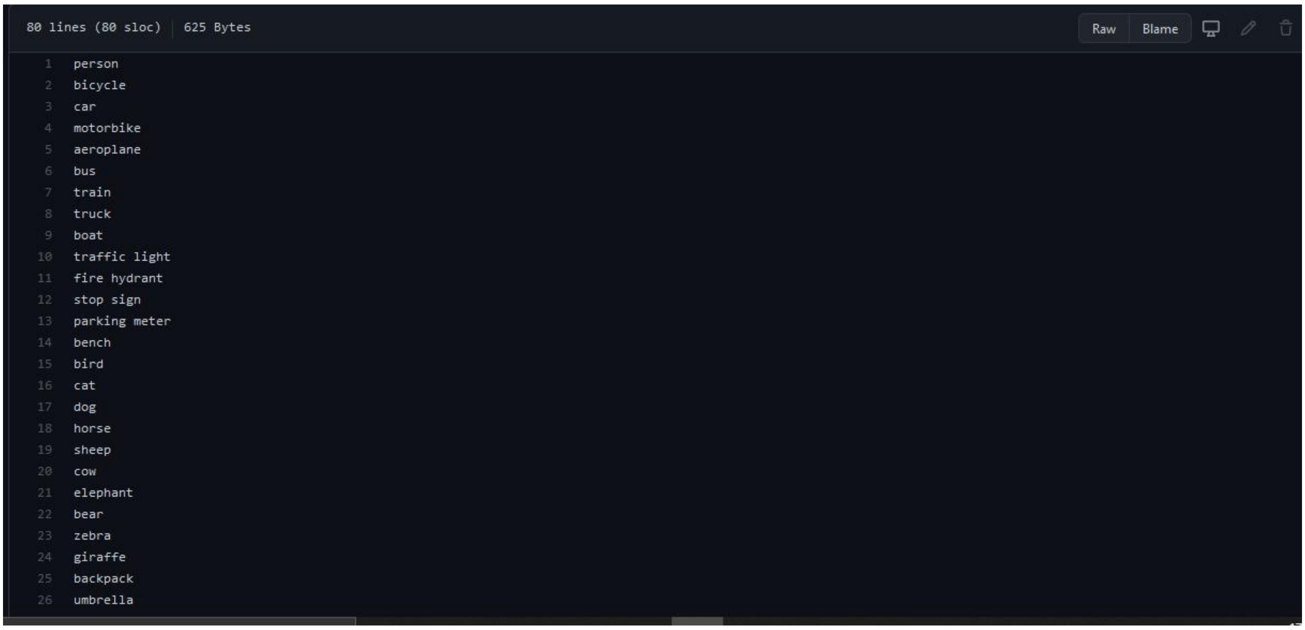
5. IMPLEMENTATION

5.1 Methodology

5.1.1 Dataset

The dataset used for the system is the “Indian Liver Patient Dataset" from the UCI machine repository. This dataset comprises 10 attributes having information of 583 patients. In the above dataset, the patients were assigned either 1 or 2 based on the presence or absence of the disease. The data set consists of 9 attributes of numerical type and one attribute of nominal type. We convert the Gender feature into numerical value either 1 or 0 in the data pre-processing.

Following figure gives the details regarding the various attributes for the dataset:



```
80 lines (80 sloc) | 625 Bytes
```

```
person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
```

Fig 5.1.1 Dataset

The result attribute is represented as a numerical value either 1 or 2 representing the presence of disease or absence. We change it to 0 or 1 for our convenience in the data pre-processing step.

Figure 5.1.1.1 Dataset image 1

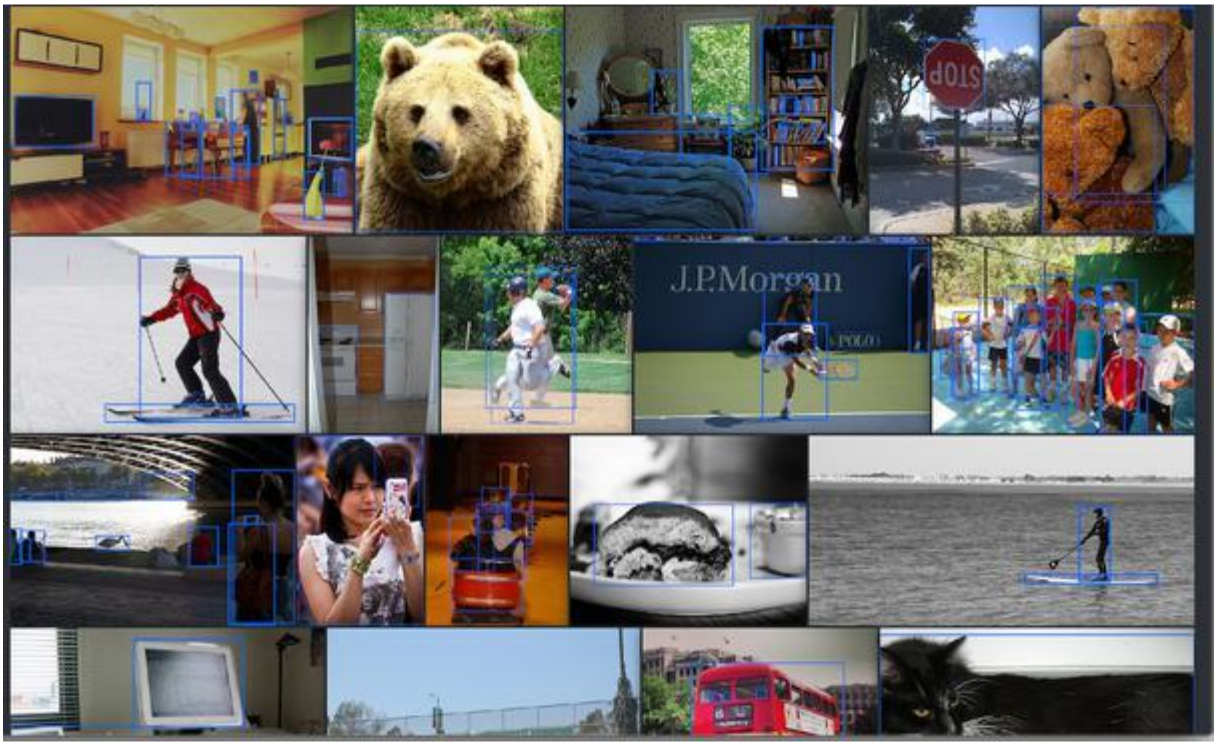


Figure 5.1.1.2 Dataset image 2

In the above figure, we show the dataset which is in the form coco data.

This has to be loaded into the algorithm for performing the training and testing operations. These entries are to be pre-processed which we will be discussing in the next section

5.1.2 Data Pre-processing

The most important part of every machine learning based system is data pre-processing. The dataset obtained may be in raw format and cannot be fed directly into the machine. Almost all the datasets must be cleaned, processed and transformed in order to make them more precise and usable. This is because any un-processed dataset consists of missing values, duplicates, outliers and null values etc. Such a dataset when used without processing leads to less accurate results which are disastrous. Generally, every dataset is different and hence we can face various challenges. There may be different kinds of data which have to be made homogeneous accordingly with respect to the technologies used. In the case of our model, the missing values, duplicate values and the null values are removed in the data pre-processing. The outliers are also removed. Further the Object which is a categorical attribute is converted in terms of numerical value to either 0 or 1. Out of the total 80 Labels, which are used to name the input image using the coco dataset . The columns containing null values are modified by replacing the with mean values of the column.

The data pre-processing consists of the following process:

1.Data Rescaling

2.Data Conversion to binary format.

3.ata Standardization

4.Data Cleaning

5.Handling missing data

6.Handling null values

7.Handing duplicate values

8.Managing the outliers

Data Rescaling:

The data in our project comprises attributes with varying scales. If we rescale the data, many machine learning algorithms can benefit from this process. It involves the use of standardized range features of the data. It helps in further optimization in our work. Data scaling is useful in the case of K-Nearest neighbor and regression algorithms. We rescaled our dataset attributes in our project using scikit-learn library.

Conversion to Binary Format:

Binarizing the data or converting the data into binary format is easier to maintain when we have probabilities. In our project, we transformed the data using the binary threshold. We transformed the attribute to binary format for ease of use and understanding. It also helps in easy addition of new attributes and during feature engineering. We used the scikit-learn library in python for binarizing our dataset.

Data standardization:

Data standardization is also a vital data pre-processing technique as it helps in eliminating or removing the inconsistent data. The process of data standardization helps in making it consistent internally. It is handy in transforming attributes acquiring a Gaussian distribution which differ means standard deviation into a Gaussian Standard Gaussian Distribution. In our project we standard our data in the dataset using the scikit-learn library.

Data cleaning:

Data cleaning is one in all necessary steps in the attaining good accuracy in machine learning. It is also an essential step in our project. Data cleaning, although having great importance is mostly a hidden step that is not mentioned by most of the professionals. It helps in uncovering and cleaning the missing values, outliers the inconsistent data which may interrupt in successful execution of the model. In general, if we are using better data which is clean there is a chance of better results even when we use simple algorithms. Our dataset contains different attributes of different types. Hence it would require different methods for cleaning the dataset.

The data cleaning techniques we used in our project are elaborated as follows:

Handling missing data: Missing data could be misleadingly a tough issue in the field of machine learning. These may lead to miss interpretation and wrong results. We cannot simply remove the record in order to overcome this issue. We need to identify a method that would help in filling up these missing values so that the important records could not be missed. We deal with this issue in two ways. The former is by removing the entry of the missing values and the latter is by filling it up based on the past observations. The former step can be performed if the data is less quality or unimportant. This method may not be recommended as the information which is important would be lost. The latter is inserting the missing values using appropriate statistical methods. We can insert values in the missing columns by finding out the mean and then filling the column.

Handling null values: The values also possess the same issue as the missing values; the null values can be avoided by simply deleting the record or filling up the entry by appropriate statistical method. In our project, we used the mean of the attribute column or the previous entries in or to deal with null values. We used an appropriate method available from the library available in python to perform the statistical operations.

Removing the duplicates: The presence of the duplicate entries in the dataset would have a significant effect on our model or any other machine learning model. Hence the issue of duplicates must be resolved in order to successfully execute the model. In our project, we used the drop duplicates method in order to remove the duplicates in our dataset. This method is elaborate in the upcoming sections.

Managing the outliers: The outliers also possess a challenge in developing a machine learning model. For instance, if we take the presence of outliers in a particular model, the linear regression could be less fast or accurate than the decision-tree. It is not reasonable to have unlikely data or unreasonable data in the dataset. For example, we cannot have an age of 300 entry in our dataset as it would be practically impossible and also has a negative effect on our model. Hence it is a better idea or approach to remove the outliers and work on the project. For our project, we plotted different graphs on the attributes in order to identify and detect the outliers for the attributes. Significantly we also transformed and modified the data for improving our dataset.

Hence, we applied the methods available in the python library to study the data in the dataset so that we could identify any abnormal entry or anomalies in the data. Python libraries used have provided us with appropriate methods to visualize the data and detect the missing values and outliers and overcome the problems in our dataset.

5.1.3 Language Used for Implementation

For the implementation of the system, we used Google Colab to write the python code for the system. Google **Colab** allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.**Google** have released Colaboratory: a web **IDE** for **python**, to enable Machine Learning with storage on the cloud — this internal tool had a pretty quiet public release in late 2017, and is set to make a huge difference in the world of machine learning, artificial intelligence and data science work. It supports various libraries Pandas, Numpy, Sklearn and many more libraries.

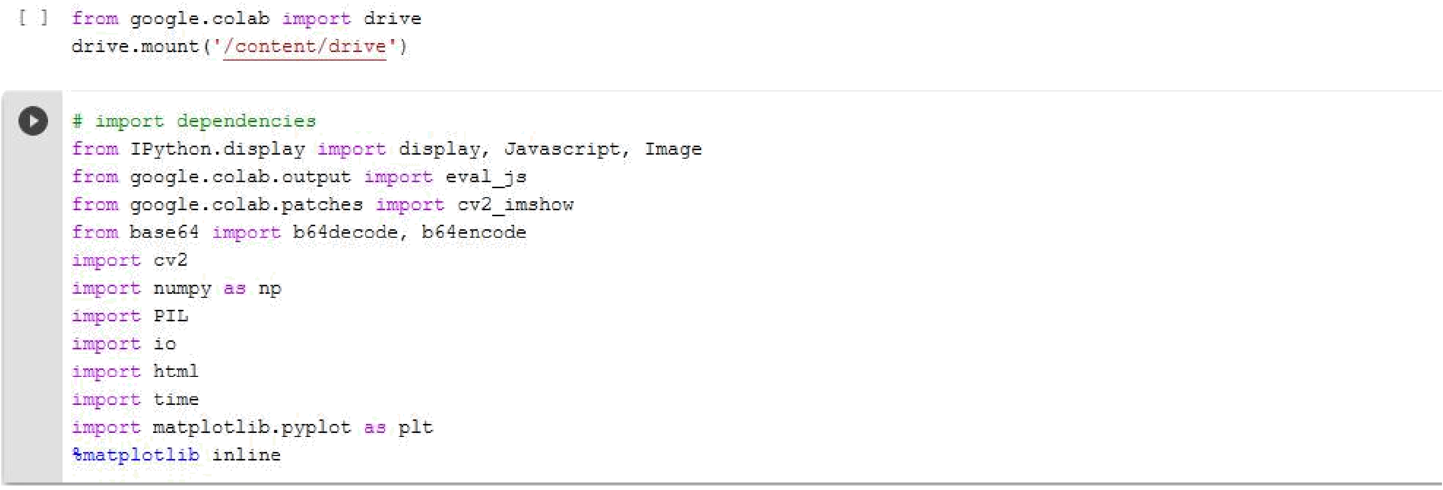
Some of the libraries used in our system are cv2, Matplotlib, Numpy, time and pyplot. **NumPy** is a **Python** library that provides a simple yet powerful data structure: the n-dimensional array. This is the foundation on which almost all the power of **Python's** data science toolkit is built, and learning **NumPy** is the first step on any **Python** data scientist's journey.

5.2 Implementation

Exploratory Data Analysis

This is an important and more interesting step in the entire process. Exploratory Data Analysis is a typical design thinking step. In this step, we study the data in depth in order to identify hidden insights in it. For this reason, it is said to be the brainstorming stage of Machine Learning. In this stage, we use various data visualization techniques in the form of graphs and plots in order to gain insights and various correlations between the variables. This will walk through all the steps for performing YOLOv4 object detections on your webcam while in Google Colab. We will be using scaled-YOLOv4 (yolov4-csp) for this tutorial, the fastest and most accurate object detector there currently is.

The dataset we considered for the project is 80 labels and coco dataset.

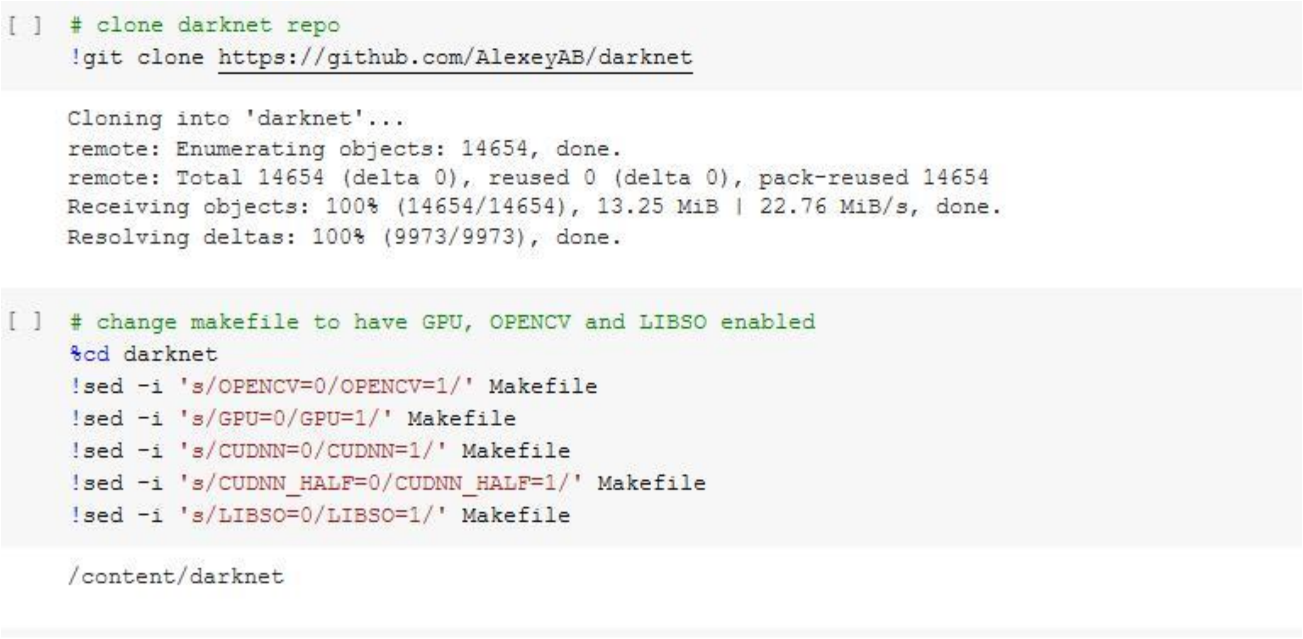


```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
# import dependencies
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

Figure 5.2.1 Identifying and Importing the required Libraries

As exploratory data analysis plays a vital role in data visualization and understanding the hidden insights, we used different methods from the libraries available in order to visualize the data. Cloning and Setting Up Darknet for YOLOv4

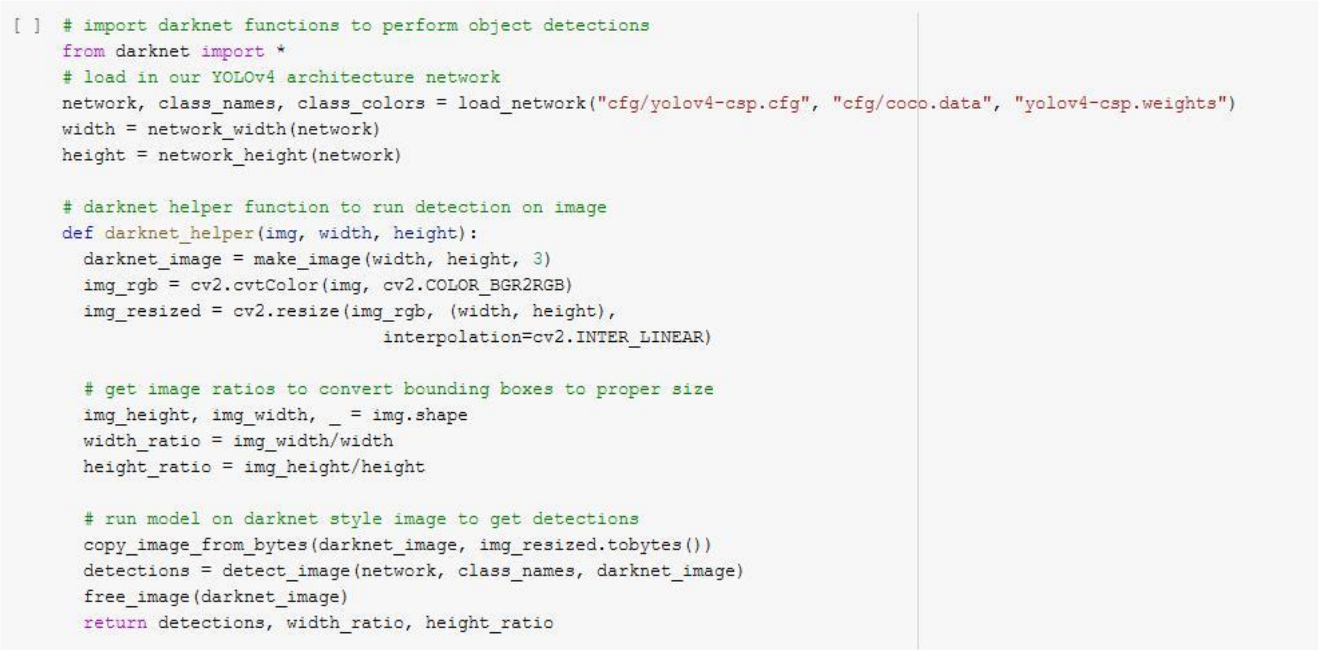


```
[ ] # clone darknet repo
    !git clone https://github.com/AlexeyAB/darknet

    Cloning into 'darknet'...
    remote: Enumerating objects: 14654, done.
    remote: Total 14654 (delta 0), reused 0 (delta 0), pack-reused 14654
    Receiving objects: 100% (14654/14654), 13.25 MiB | 22.76 MiB/s, done.
    Resolving deltas: 100% (9973/9973), done.
```

```
[ ] # change makefile to have GPU, OPENCV and LIBSO enabled
    %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
    !sed -i 's/LIBSO=0/LIBSO=1/' Makefile

    /content/darknet
```

Figure 5.2.2 Code snippet for Cloning the dataset

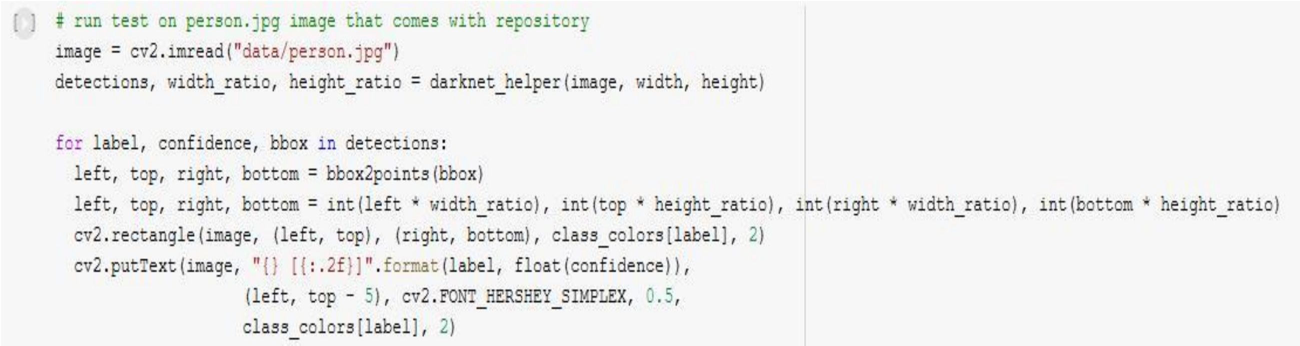
In order to utilize YOLOv4 with Python code we will use some of the pre-built functions found within darknet.py by importing the functions into our workstation. Feel free to check out the darknet.py file to see the function definitions in detail!



```
# import darknet functions to perform object detections
from darknet import *
# load in our YOLOv4 architecture network
network, class_names, class_colors = load_network("cfg/yolov4-csp.cfg", "cfg/coco.data", "yolov4-csp.weights")
width = network_width(network)
height = network_height(network)

# darknet helper function to run detection on image
def darknet_helper(img, width, height):
  darknet_image = make_image(width, height, 3)
  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_resized = cv2.resize(img_rgb, (width, height),
                              interpolation=cv2.INTER_LINEAR)

  # get image ratios to convert bounding boxes to proper size
  img_height, img_width, _ = img.shape
  width_ratio = img_width/width
  height_ratio = img_height/height

  # run model on darknet style image to get detections
  copy_image_from_bytes(darknet_image, img_resized.tobytes())
  detections = detect_image(network, class_names, darknet_image)
  free_image(darknet_image)
  return detections, width_ratio, height_ratio
```

Fig 5.2.3 Darknet for Python.

The following is the code snippet used to make sure our model has successfully been loaded and that we can make detections properly on a test image.



```
# run test on person.jpg image that comes with repository
image = cv2.imread("data/person.jpg")
detections, width_ratio, height_ratio = darknet_helper(image, width, height)

for label, confidence, bbox in detections:
  left, top, right, bottom = bbox2points(bbox)
  left, top, right, bottom = int(left * width_ratio), int(top * height_ratio), int(right * width_ratio), int(bottom * height_ratio)
  cv2.rectangle(image, (left, top), (right, bottom), class_colors[label], 2)
  cv2.putText(image, "{} [{:.2f}]".format(label, float(confidence)),
                    (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    class_colors[label], 2)
```

Fig 5.2.4 YOLOv4 on Test Image

The output for the yolo v4 on the test image is the label of the objects inside the images provided . The figure below displays it.

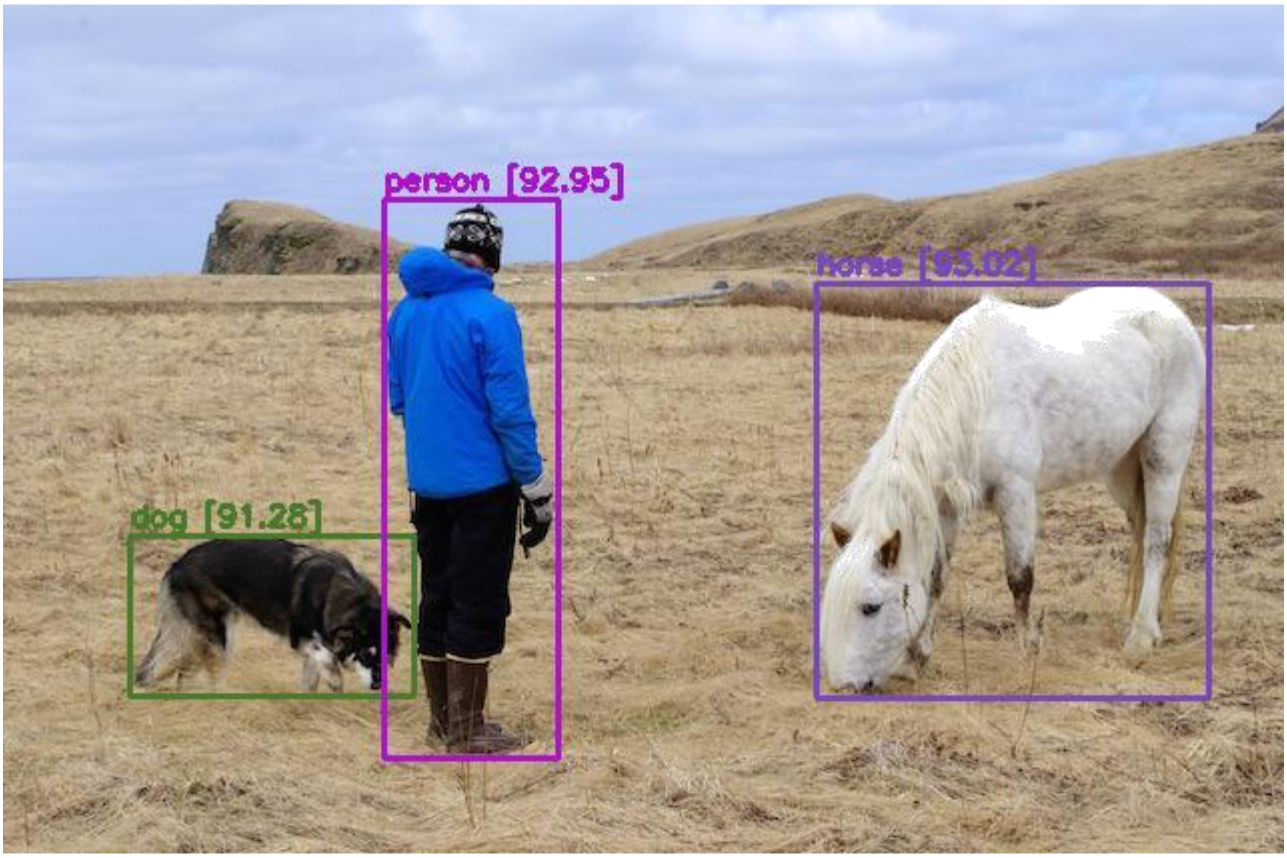
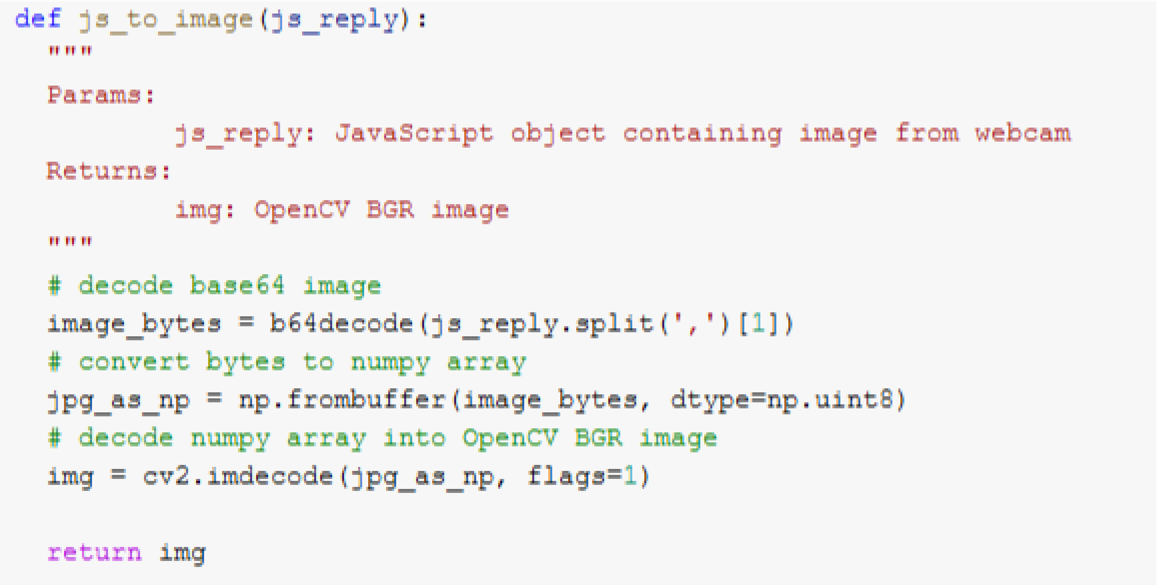


Fig 5.2.5 Labels to the objects and persons in the image

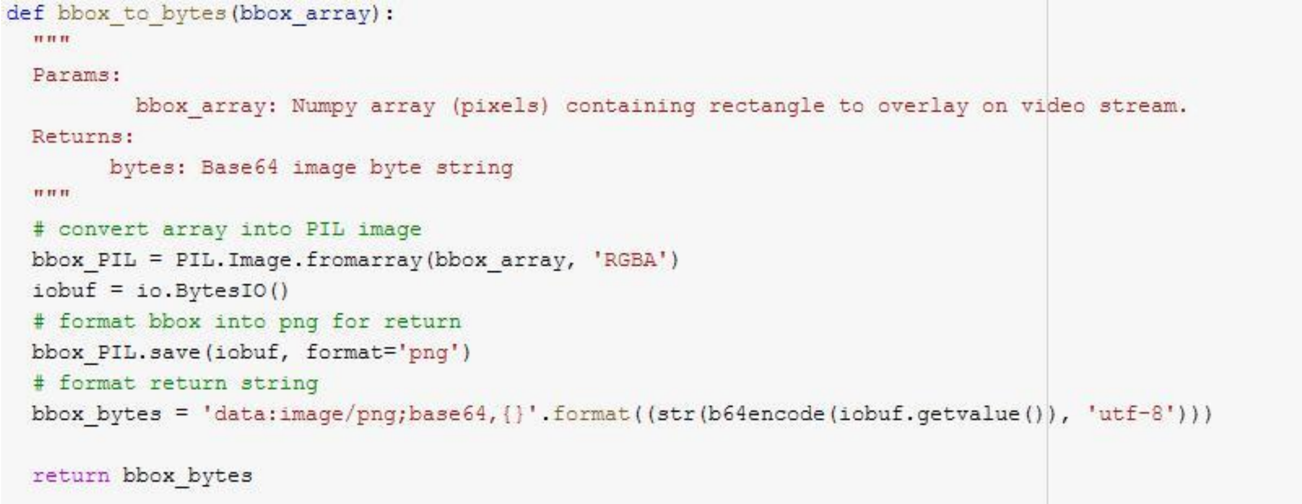
Here are a few helper functions defined that will be used to easily convert between different image types within our later steps.function to convert the JavaScript object into an OpenCV image.



```
def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webcam
  Returns:
          img: OpenCV BGR image
  """
  # decode base64 image
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)

  return img
```

Figure 5.2.6 Helper Functions.

The following is the code snippet for function
to convert OpenCV Rectangle bounding box image into base64 byte string to be overlaid on video str eam.

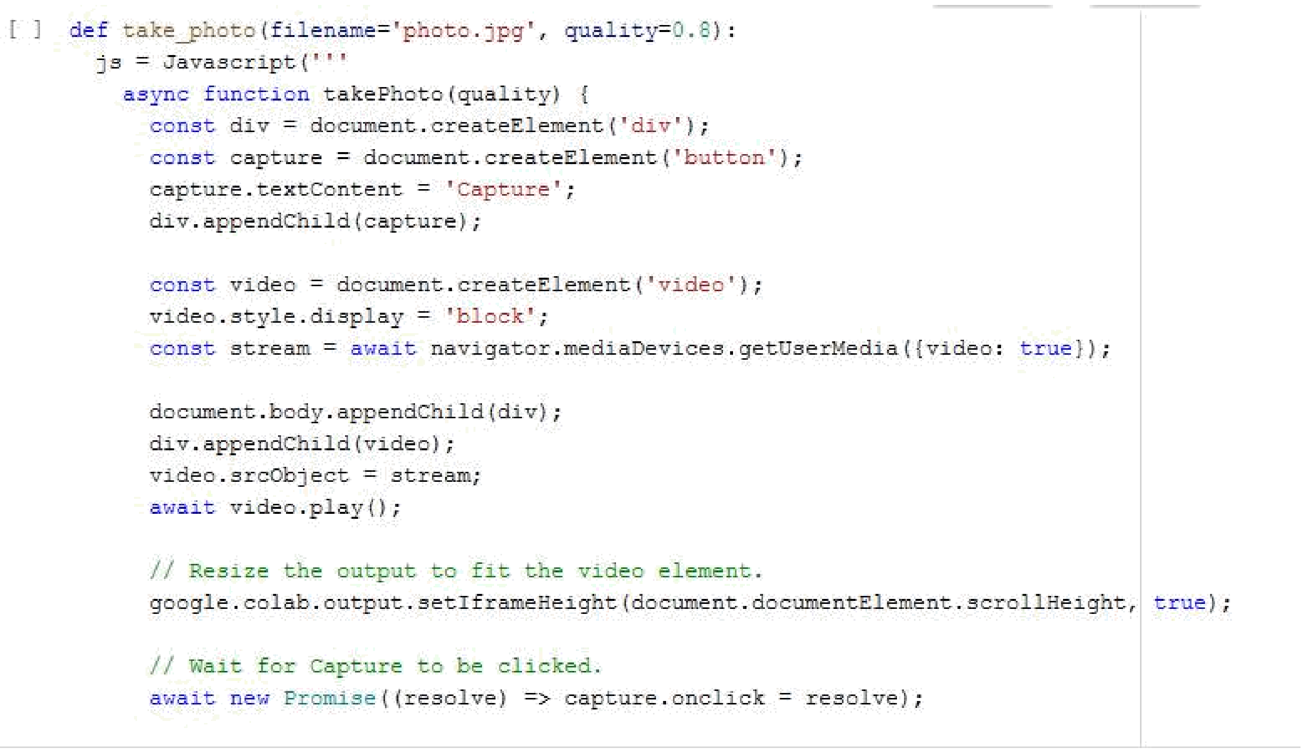


```
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

  return bbox_bytes
```

Figure 5.2.7 Helper functions(2).

Running YOLOv4 on images taken from a webcam is fairly straight-forward. We will utilize code within Google Colab's **Code Snippets** that has a variety of useful code functions to perform various tasks.We will be using the code snippet for **Camera Capture** which runs JavaScript code to utilize your computer's webcam. The code snippet will take a webcam photo, which we will then pass into our YOLOv4 model for object detection.

Below is a function to take the webcam picture using JavaScript and then run YOLOv4 on it.



```
[ ]  def take_photo(filename='photo.jpg', quality=0.8):
       js = Javascript('''
         async function takePhoto(quality) {
           const div = document.createElement('div');
           const capture = document.createElement('button');
           capture.textContent = 'Capture';
           div.appendChild(capture);

           const video = document.createElement('video');
           video.style.display = 'block';
           const stream = await navigator.mediaDevices.getUserMedia({video: true});

           document.body.appendChild(div);
           div.appendChild(video);
           video.srcObject = stream;
           await video.play();

           // Resize the output to fit the video element.
           google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

           // Wait for Capture to be clicked.
           await new Promise((resolve) => capture.onclick = resolve);
```

Figure 5.2.8 YOLOv4 on Webcam Images code snippet.

The output for the above webcam images that are provided as inputs:

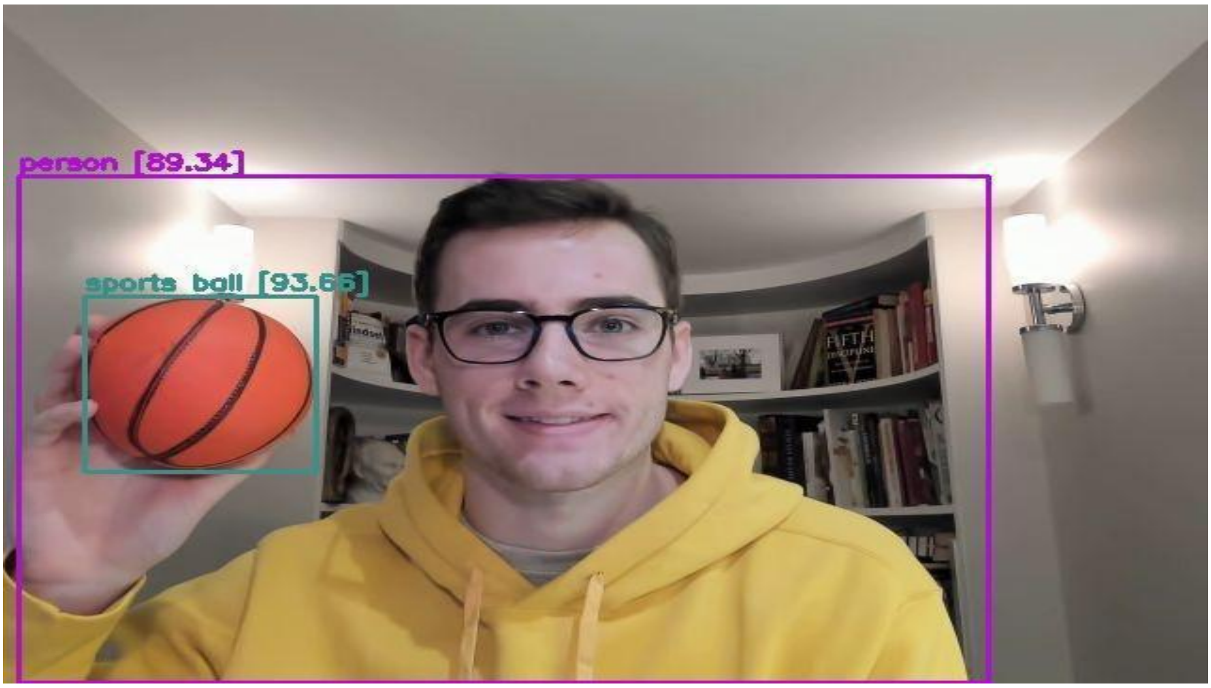


Figure 5.2.9 YOLOv3 on Webcam Images.

Running YOLOv3 on webcam video is a little more complex than images. We need to start a video stream using our webcam as input. Then we run each frame through our YOLOv3 model and create an overlay image that contains a bounding box of detection(s). We then overlay the bounding box image back onto the next frame of our video stream.

YOLOv3 is so fast that it can run the detections in real-time!

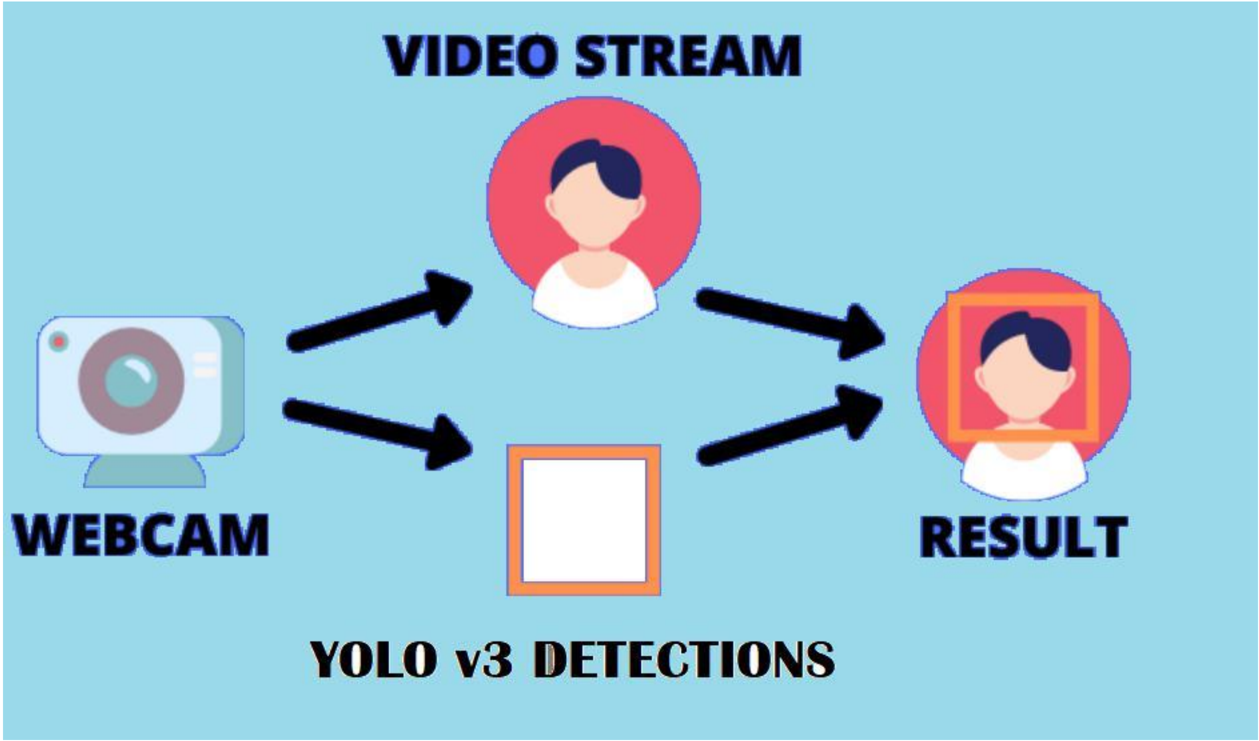


Figure 5.2.10 YOLOv3 on Webcam Videos

Algorithms Implemented

The main purpose of the project is to identify an algorithm which provides better accuracy than the existing system. In this case, we implement different machine learning models and identify the algorithm which has the highest accuracy. The performance evaluation is then performed on the classification model and then accordingly we build a concluding trained model. The model is then deployed in the server using the appropriate framework. Following are the different classification algorithm we implemented in our system:

YOLO - You Only Look Once:

All the previous object detection algorithms have used regions to localize the object within the image.

The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region based algorithms which are seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

Algorithms based on regression – instead of selecting interesting parts of an image, they predict classes and bounding boxes for the whole image **in one run of the algorithm**. The two best known examples from this group are the **YOLO (You Only Look Once)** family algorithms and SSD (Single Shot Multibox Detector). They are commonly used for real-time object detection as, in general, they trade a bit of accuracy for large improvements in speed.

To understand the YOLO algorithm, it is necessary to establish what is actually being predicted. Ultimately, we aim to predict a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:

- center of a bounding box (**bxby**)
- width (**bw**)
- height (**bh**)
- value **c**is corresponding to a class of an object (such as: car, traffic lights, etc.).

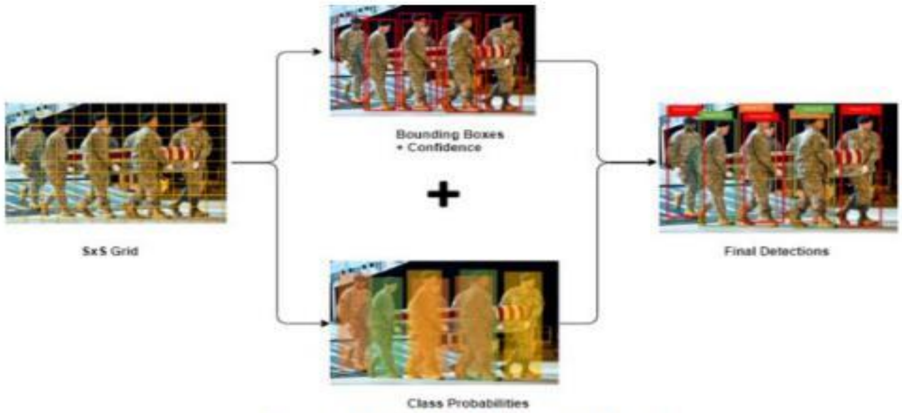


Fig -4: YOLOv3 on custom database

Figure 5.2.11 Yolo Algorithm Workflow

YOLO works by taking an image and splitting it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding boxes, the network gives an output a class probability and offset values for the bounding box. The bounding boxes have the class probability above which a threshold value is selected and used to locate the object within the image.

YOLO is orders of magnitude faster(45 frames per second) than any other object detection algorithms.

The limitation of YOLO algorithm is that it struggles with the small objects within the image, for

For example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm

1.1 Bounding box predictions:

The YOLO algorithm is used for predicting the accurate bounding boxes from the image. The image divides into S x S grids by predicting the bounding boxes for each grid and class probabilities. Both image classification and object localization techniques are applied for each grid of the image and each grid is assigned with a label. Then the algorithm checks each grid separately and marks the label which has an object in it and also marks its bounding boxes. The labels of the grid without object are marked as zero.

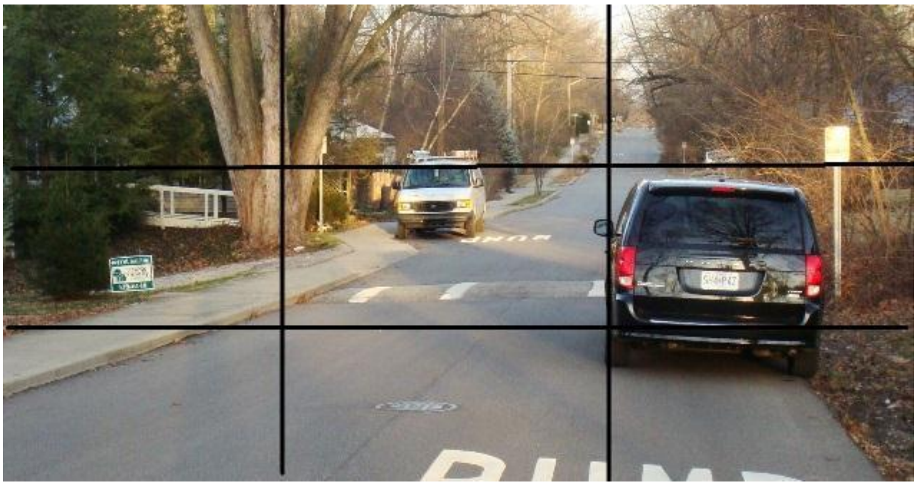


Figure 5.2.12 Example image with 3x3 grids

In addition, we have to predict the pc value, which is the probability that there is an object in the bounding box.

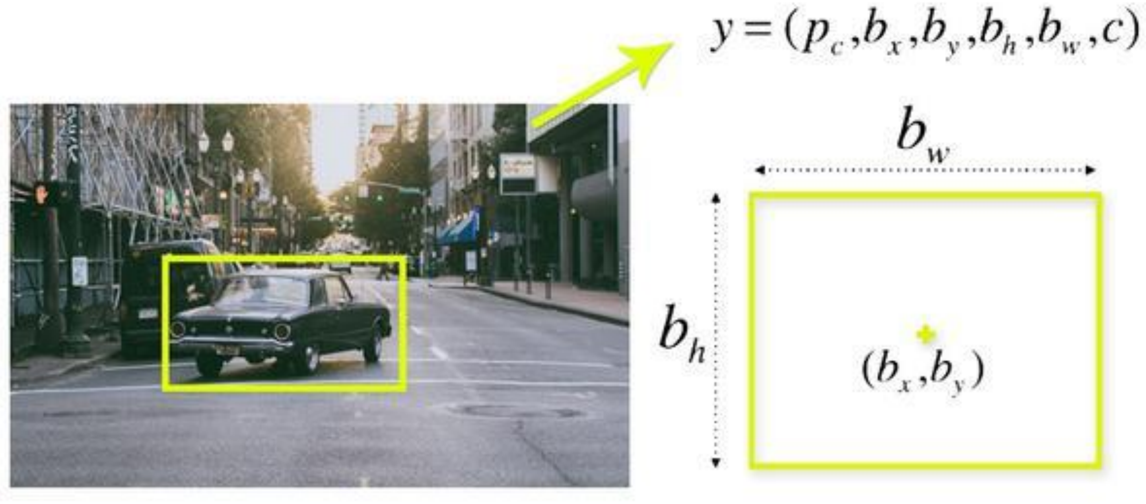


Figure 5.2.13 Image with the bounding probabilities

As we mentioned above, when working with the YOLO algorithm we are not searching for interesting regions in our image that could potentially contain an object.

Instead, we are splitting our image into cells, typically using a 19×19 grid. Each cell is responsible for predicting 5 bounding boxes (in case there is more than one object in this cell). Therefore, we arrive at a large number of 1805 bounding boxes for one image.

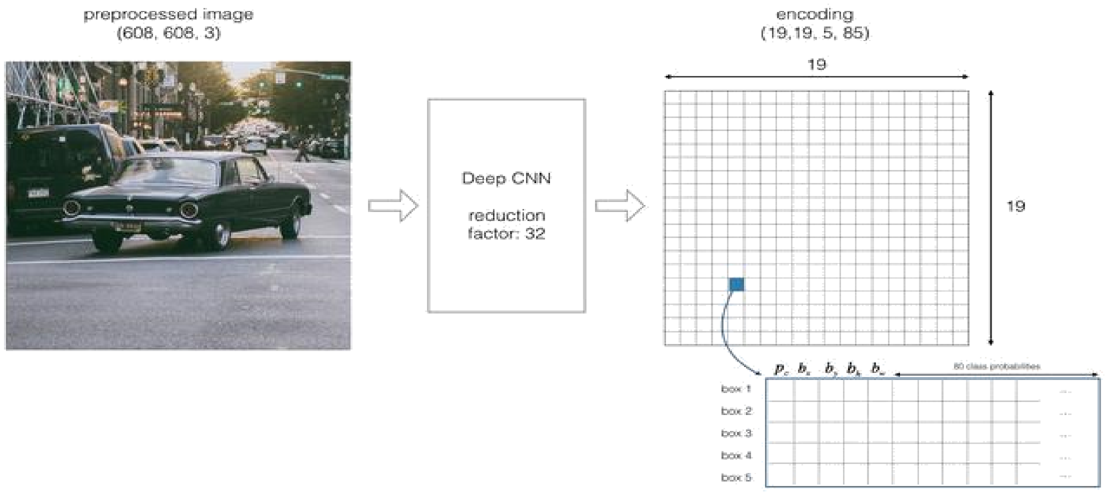


Figure 5.2.14 Image with the bounding probabilities(2)

Most of these cells and bounding boxes will not contain an object. Therefore, we predict the value pc, which serves to remove boxes with low object probability and bounding boxes with the highest shared area in a process called **non-max suppression**.

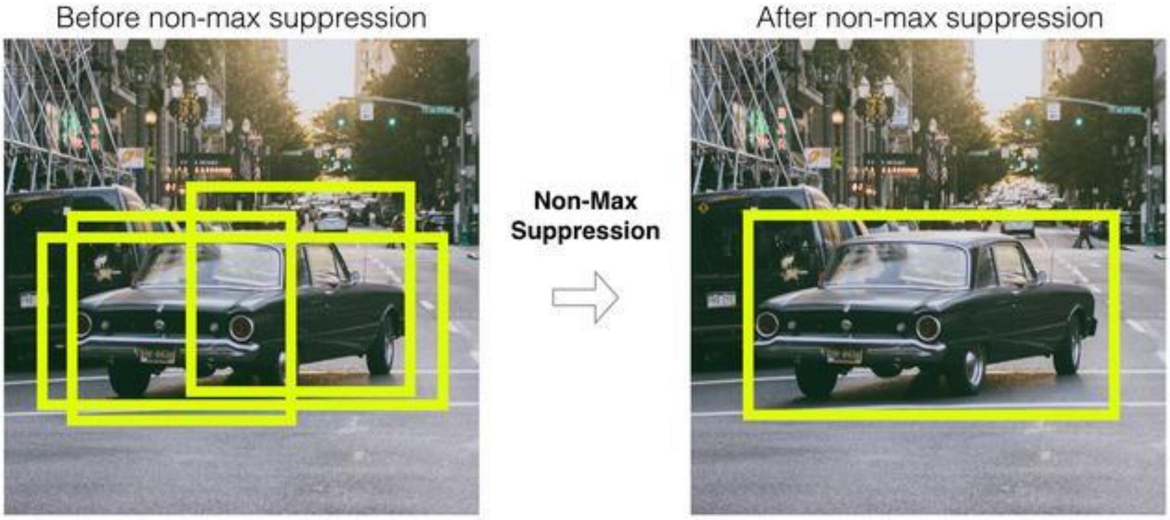


Figure 5.2.15 Image with non-max suppression before and after.

ACCURACY IMPROVEMENT:

ANCHOR BOX: By using Bounding boxes for object detection, only one object can be identified by a grid. So, for detecting more than one object we go for Anchor box.

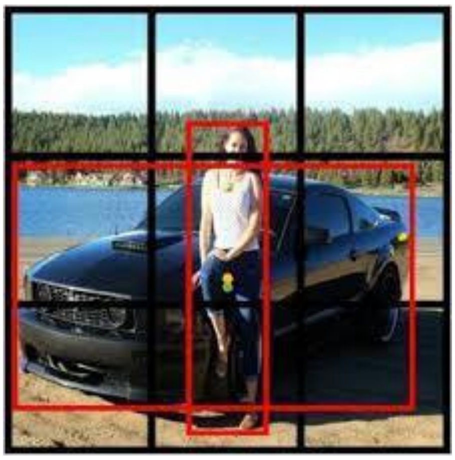


Figure 5.2.16 An example image for anchor box

Consider the above picture, in that both the human and the car's midpoint come under the same grid cell. For this case, we use the anchor box method. The red color grid cells are the two anchor boxes for those objects. Any number of anchor boxes can be used for a single image to detect multiple objects. In our case, we have taken two anchor boxes.In this type of overlapping object detection, the label Y contains 16 values i.e, the values of both anchor boxes.

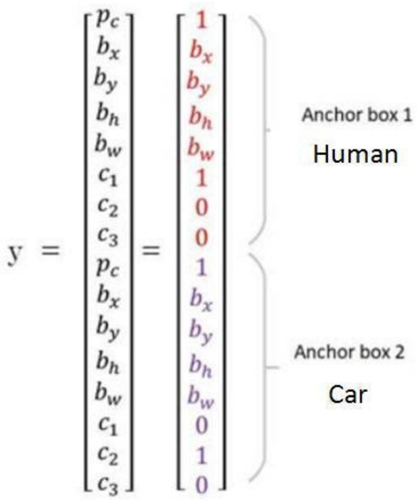


Figure 5.2.17 Anchor box prediction values

Pc in both the anchor box represents the presence of the object. bx, by, bh, bw in both the anchor boxes represent their corresponding bounding box values. The value of the class in anchor box 1 is (1, 0, 0) because the detected object is a human.

In the case of anchor box 2, the detected object is a car so the class value is (0, 1, 0). In this case, the matrix form of Y will be Y= 3x3x16 or Y= 3x3x2x8. Because of two anchor boxes, it is 2x8.

Loss function of YOLO algorithm:

For a single grid cell, the algorithm predicts multiple bounding boxes. To calculate the loss function we use only

one bounding box for object responsibility. For selecting one among the bounding boxes we use the high IoU value. The box with high IoU will be responsible for the object. Various loss functions are:

- Classification loss function
- Localization loss function
- Confidence loss function

Localization loss means the error between the ground truth value and predicted boundary box. Confidence loss is the objectness of the box. Classification loss calculated as, the squared error of the class conditional probabilities for each class:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

The localization loss is the measure of errors in the predicted boundary box locations and the sizes. The box which is responsible for the object is only counted.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right]$$
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right]$$

The Confidence loss, if the object is found in a box the confidence loss is,

If the object is not detected then the confidence loss will be,

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

5.3 Result

In our project, we have tried to determine an algorithm having accuracy greater than the existing system. The data pre-processing helps in furnishing the dataset and the exploratory data analysis helps in gaining better insights. After implementing the dataset on the machine learning model, we determine the algorithm with highest accuracy as the winning model. The accuracy rates for the different models are shown in the following table:

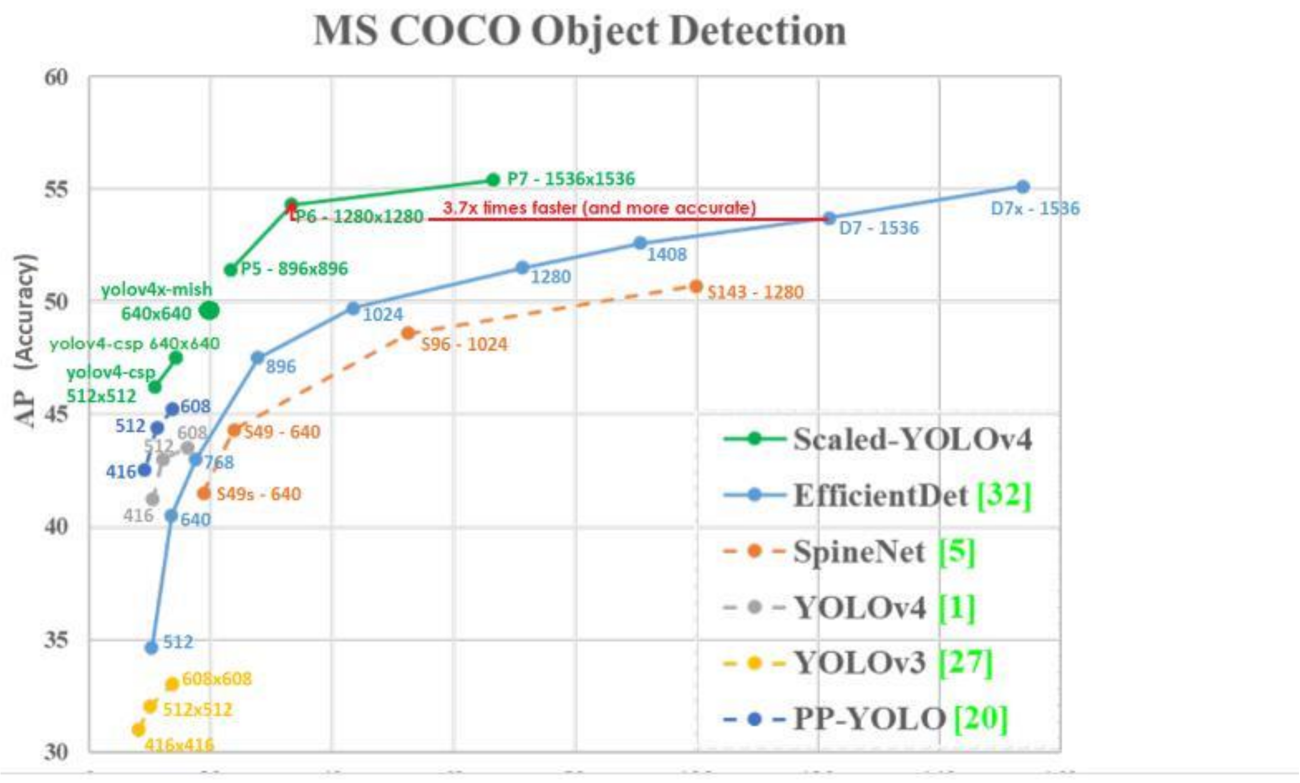
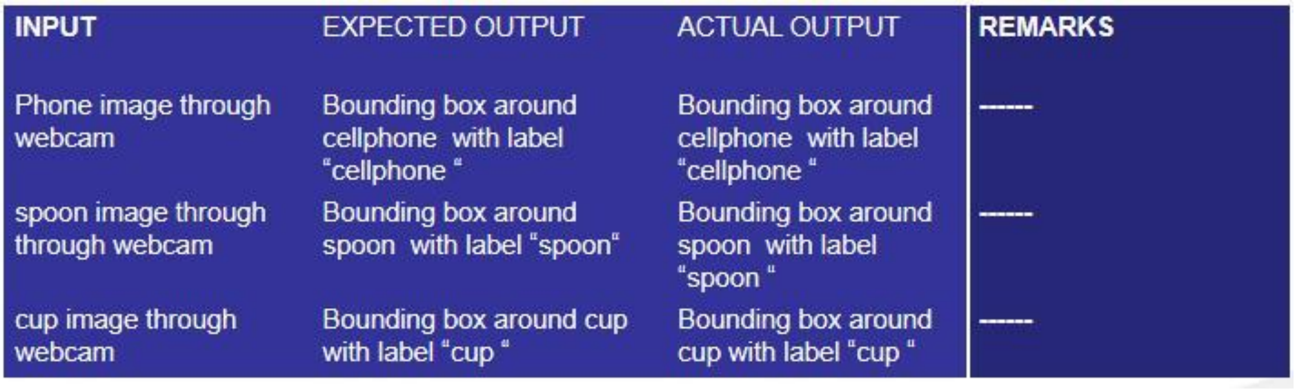


Fig 5.3.1 Accuracy of the Models Implemented

From the above, the YOLO Algorithm gives a high accuracy rate. This is the winning model and in order to run this we have used vs code using python.

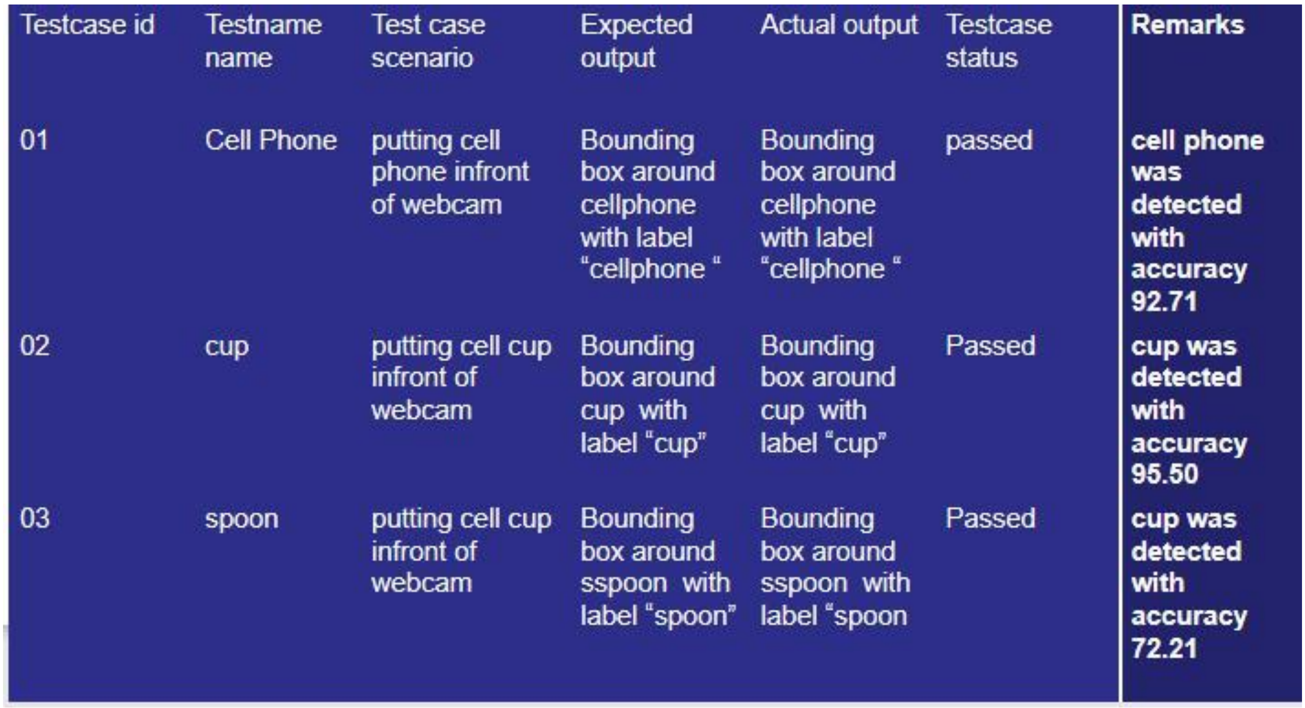
We gave several inputs like cellphone, cup, spoon, apple etc.it detected successfully by drawing a bounding box around the object with a label detected with certain accuracy.



INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
Phone image through webcam	Bounding box around cellphone with label “cellphone “	Bounding box around cellphone with label “cellphone “	------
spoon image through through webcam	Bounding box around spoon with label “spoon“	Bounding box around spoon with label “spoon “	------
cup image through webcam	Bounding box around cup with label “cup “	Bounding box around cup with label “cup “	------

Fig 5.3.2 Input with expected output table

Below are the test cases provided with the expected output using the phone or webcam.



Testcase id	Testname name	Test case scenario	Expected output	Actual output	Testcase status	Remarks
01	Cell Phone	putting cell phone infront of webcam	Bounding box around cellphone with label “cellphone “	Bounding box around cellphone with label “cellphone “	passed	cell phone was detected with accuracy 92.71
02	cup	putting cell cup infront of webcam	Bounding box around cup with label “cup”	Bounding box around cup with label “cup”	Passed	cup was detected with accuracy 95.50
03	spoon	putting cell cup infront of webcam	Bounding box around sspoon with label “spoon”	Bounding box around sspoon with label “spoon	Passed	cup was detected with accuracy 72.21

Fig 5.3.3 Test cases for the input provided

The output for the above test cases that we could display the predict for our project.

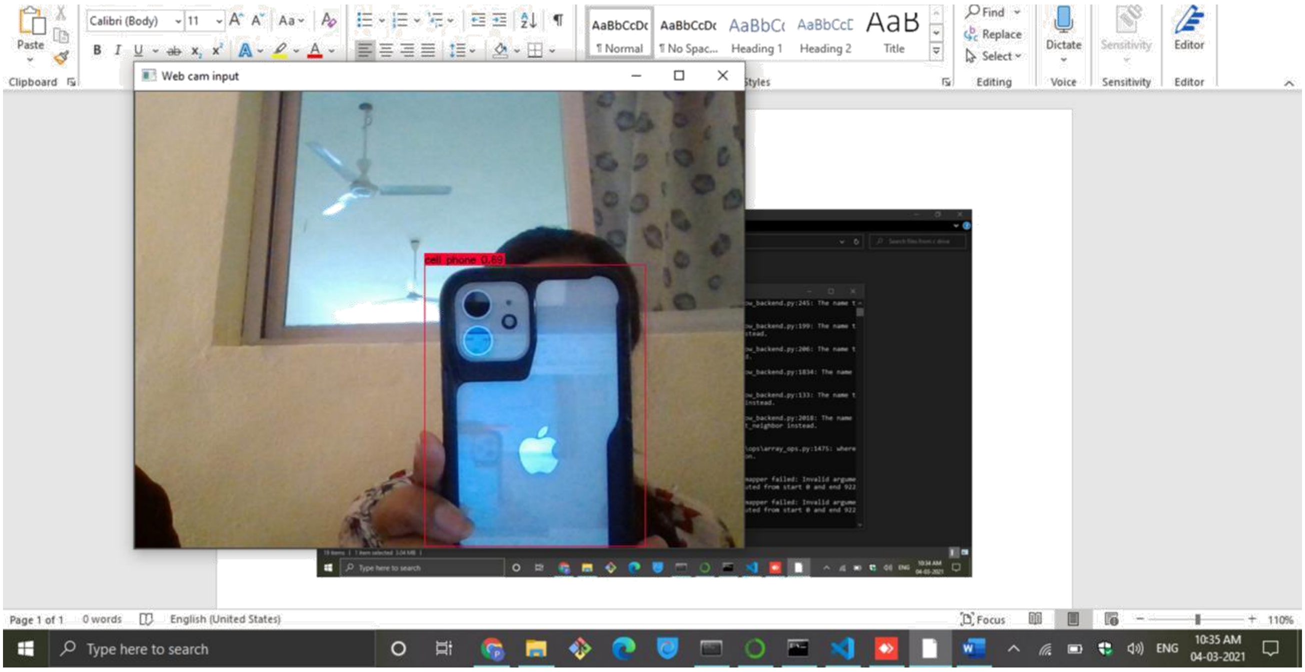
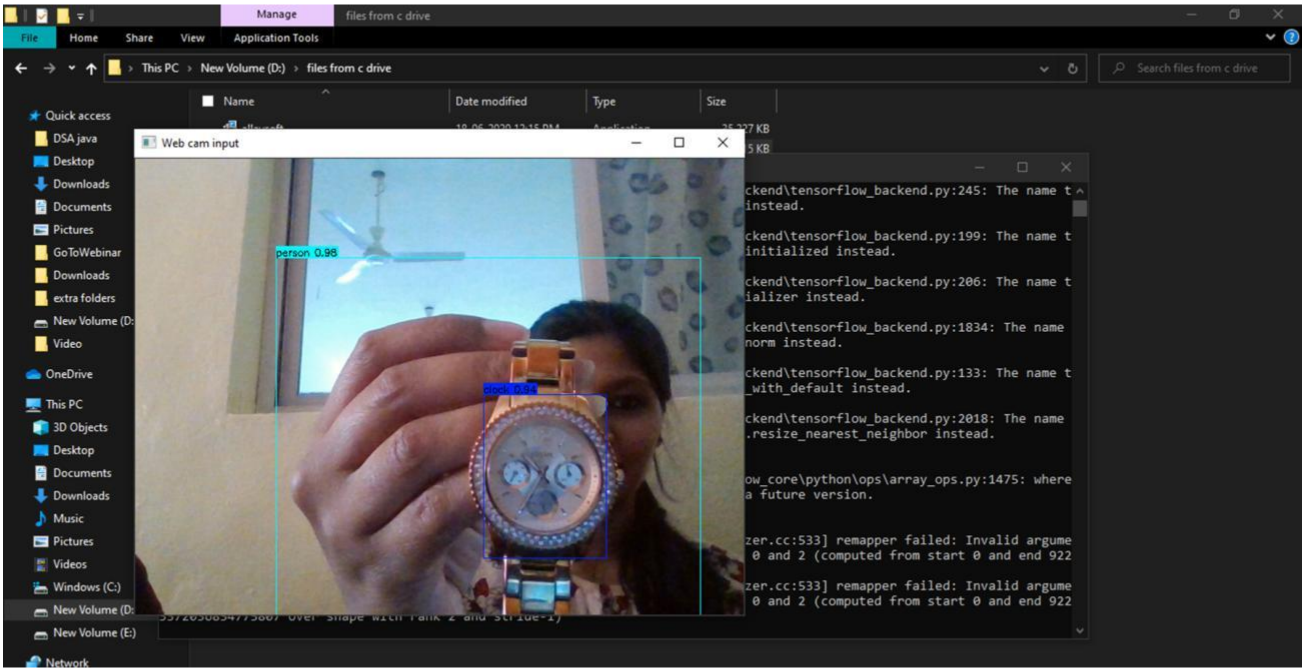
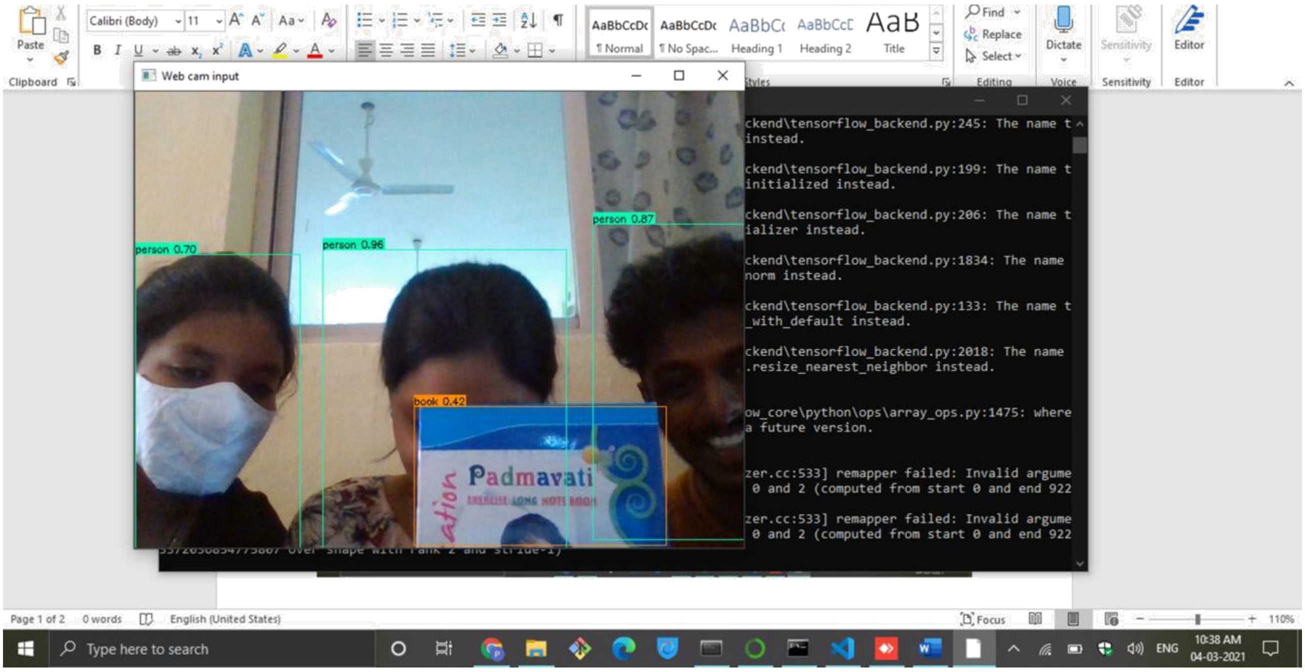


Fig 5.3.4 output of image showing the labels for person and phone with the accuracy



```
ckend\tensorflow_backend.py:245: The name t
instead.

ckend\tensorflow_backend.py:199: The name t
initialized instead.

ckend\tensorflow_backend.py:206: The name t
ializer instead.

ckend\tensorflow_backend.py:1834: The name
norm instead.

ckend\tensorflow_backend.py:133: The name t
_with_default instead.

ckend\tensorflow_backend.py:2018: The name
.resize_nearest_neighbor instead.


ow_core\python\ops\array_ops.py:1475: where
a future version.


zer.cc:533] remapper failed: Invalid argume
 0 and 2 (computed from start 0 and end 922

zer.cc:533] remapper failed: Invalid argume
 0 and 2 (computed from start 0 and end 922
```

Fig 5.3.5 output of image showing the labels for watch with the accuracy



```
ckend\tensorflow_backend.py:245: The name t
instead.

ckend\tensorflow_backend.py:199: The name t
initialized instead.

ckend\tensorflow_backend.py:206: The name t
ializer instead.

ckend\tensorflow_backend.py:1834: The name
norm instead.

ckend\tensorflow_backend.py:133: The name t
_with_default instead.

ckend\tensorflow_backend.py:2018: The name
.resize_nearest_neighbor instead.

ow_core\python\ops\array_ops.py:1475: where
a future version.

zer.cc:533] remapper failed: Invalid argume
 0 and 2 (computed from start 0 and end 922

zer.cc:533] remapper failed: Invalid argume
 0 and 2 (computed from start 0 and end 922
```

Fig 5.3.6 output of image showing the labels for book and person with the accuracy

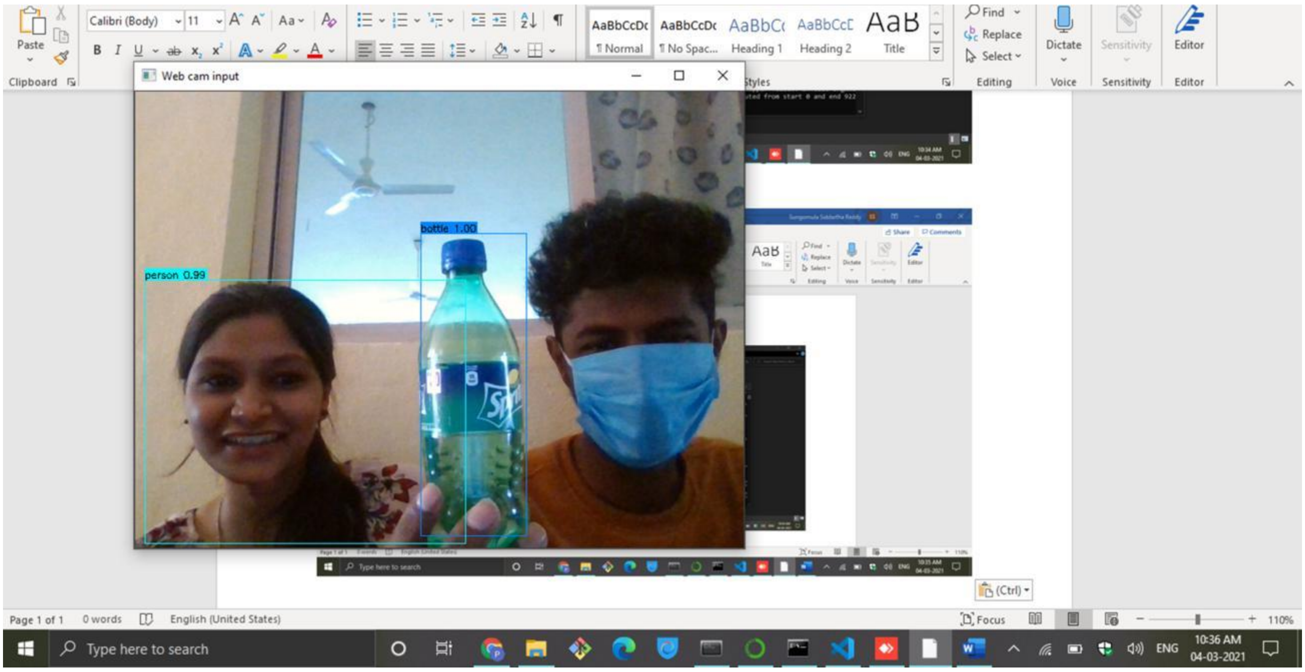


Fig 5.3.7 output of image showing the labels for person and bottle with the accuracy

6. TESTING

6.1 Types of Testing

For our project, we have implemented Integration testing and Unit testing methodologies to test its working. In the case of unit testing, we tested every individual module in order to verify if the individual parts are correct. We have tested the individual internal code modules. In the case of Integration testing, we tested the code for each module and then integrated them based on the hierarchy to top and finally tested the entire code to verify whether it passed all of the test cases. While performing Integration testing, we identified a few bugs in the final code and then we modified the code and retested it again and again until it satisfied all test cases.

6.2 List of Test Cases

S.NO	Test Name	Input	Expected output	Actual Output	Test Case Result
1	Cell Phone	Dataset	Dataset Loaded	Dataset read	PASS
2	Person and watch	Training and testing dataset	Splitting of the data into training and testing	Dividing data into training and testing	PASS
3	Book and Person	Training set, parameters	Trained with the provided set	Trained model	PASS
4	Person and Bottle	No of entries from testing data	Validation of the model with best fit	Model generated	PASS

Tab 6.2.1 List of Test Cases

7. CONCLUSION

7.1 Conclusion

The primary objective of the project was to determine YOLO algorithm to detect objects using a single neural network.This algorithm is generalized, it outperforms different strategies once generalizing from natural pictures to different domains. The algorithm is simple to build and can be trained directly on a complete image.Region proposal strategies limit the classifier to a particular region. YOLO accesses the entire image in predicting boundaries. And also it predicts fewer false positives in background areas.Comparing to other classifier algorithms this algorithm is a much more efficient and fastest algorithm to use in real time. By using this thesis and based on experimental results we are able to detect objects more precisely and identify the objects individually with exact location of an object in the picture in x,y axis.This paper also provides experimental results on different methods for object detection and identification and compares each method for their efficiencies.

7.2 Future Work

The object recognition system can be applied in the area of surveillance systems, face recognition, fault detection, character recognition etc. The objective of this thesis is to develop an object recognition system to recognize the 2D and 3D objects in the image. The performance of the object The recognition system depends on the features used and the classifier employed for recognition. This research work attempts to propose a novel feature extraction method for extracting global features and obtaining local features from the region of interest. Also the research work attempts to hybrid the traditional classifiers to recognize the object.

8. BIBLIOGRAPHY

[1] https://appsilon.com/object-detection-yolo-algorithm

[2] https://journal.utem.edu.my/index.php/jtec/article/view/3599

[3] https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc

[3] https://ieeexplore.ieee.org/abstract/document/9133581/

[4] https://arxiv.org/abs/1709.05943

[5] https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

APPENDIX-A: Python Modules

About Python:

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured , object oriented and functional programming.

Python is often described as a "batteries included" language due to its comprehensive standard library

About NumPy:

NumPy is mostly written in C language, and it is an extension module of Python. It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single-dimensional array elements. The calculations using NumPy arrays are faster than the normal Python array.It is also capable of handling a vast amount of data and convenient with Matrix multiplication and data reshaping.

About Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Matplotlib comes with a wide variety of plots. Plots help to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. There are thousands of libraries in Python, and Matplotlib is one of the most powerful tools for data visualization in Python. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code.

APPENDIX-B: UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. It is intended for use with all development methods, lifecycle stages, application domains, and media. The modeling language is intended to unify past experience about modeling techniques and to incorporate current software best practices into a standard approach. UML includes semantic concepts, notation, and guidelines. It has static, dynamic, environmental, and organizational parts. It is intended to be supported by interactive visual modeling tools that have code generators and report writers. The UML specification does not define a standard process but is intended to be useful with an iterative development process. It is intended to support most existing object-oriented development processes.

The UML captures information about the static structure and dynamic behavior of a system. A system is modeled as a collection of discrete objects that interact to perform work that ultimately benefits an outside user. The static structure defines the kinds of objects important to a system and to its implementation, as well as the relationships among the objects. The dynamic behavior defines the history of objects over time and the communications among objects to accomplish goals.

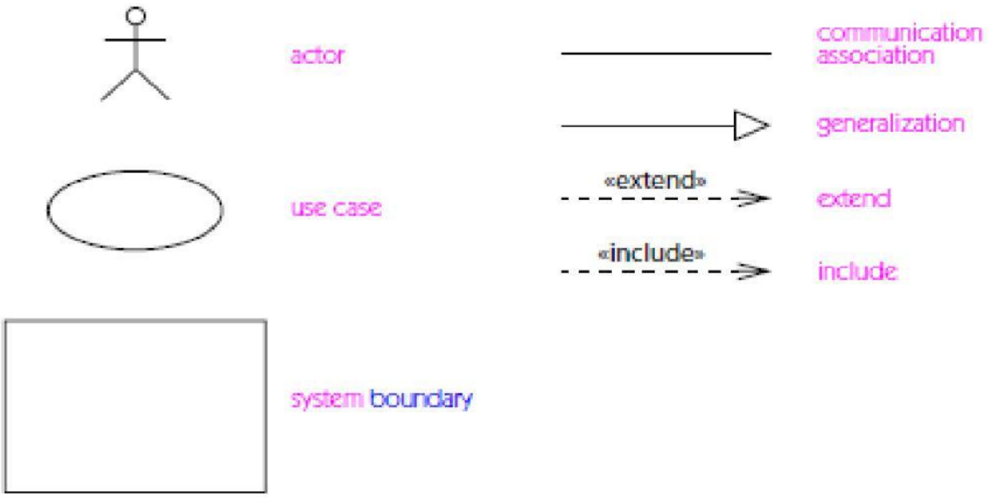
Modeling a system from several separate but related viewpoints permits it to be understood for different purposes.

The UML also contains organizational constructs for arranging models into packages that permit software teams to partition large systems into workable pieces, to understand and control dependencies among the packages, and to manage the versioning of model units in a complex development environment. It contains constructs for representing implementation decisions and for organizing run-time elements into components.

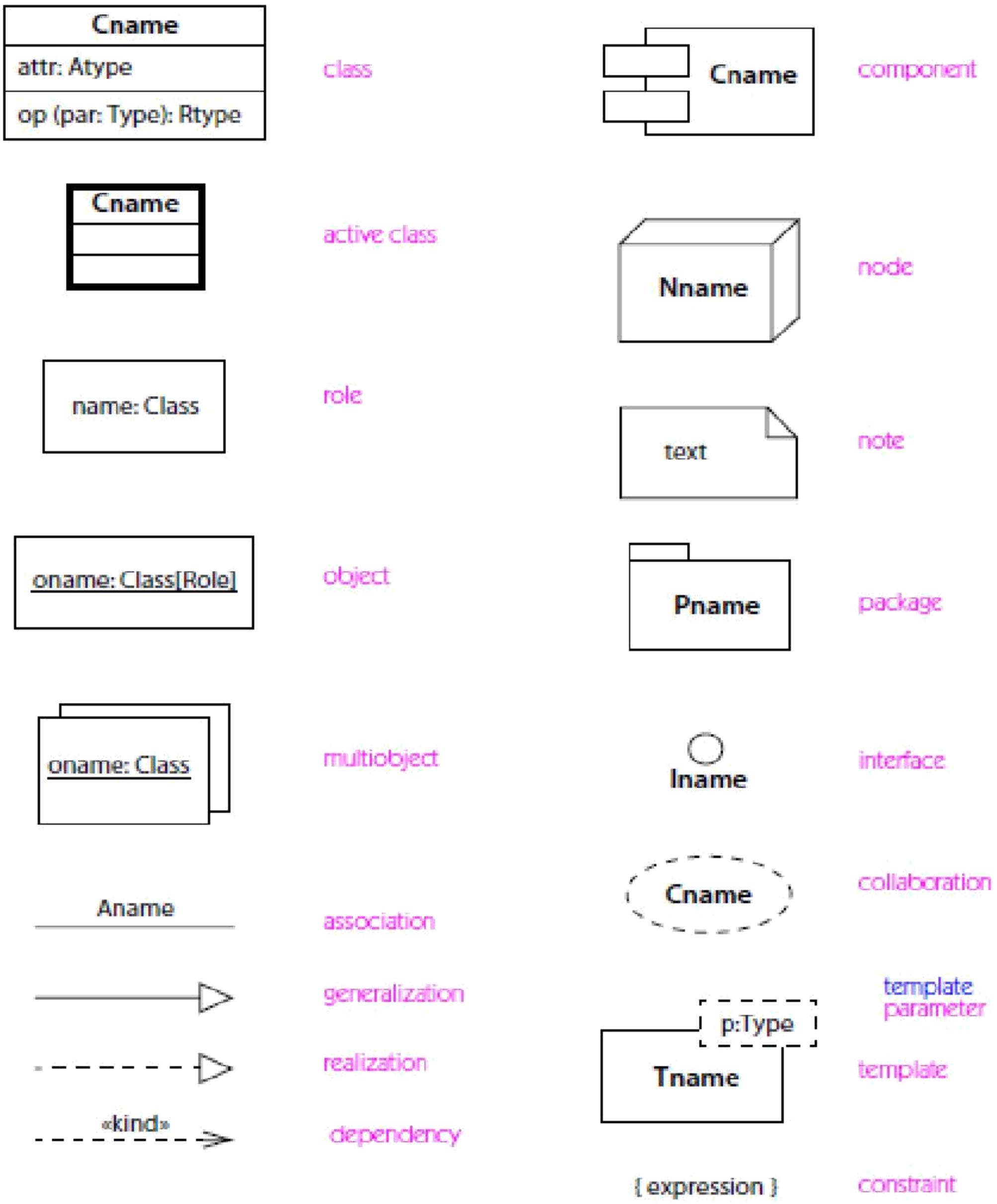
UML is not a programming language. Tools can provide code generators from UML into a variety of programming languages, as well as construct reverse engineered models from existing programs. The UML is not a highly formal language intended for theorem proving. There are a number of such languages, but they are not easy to understand or to use for most purposes. UML is a general-purpose modeling language. For specialized domains, such as GUI layout, VLSI circuit design, or rule-based artificial intelligence, a more specialized tool with a special language might be appropriate. UML is a discrete modeling language.

It is not intended to model continuous systems such as those found in engineering and physics. UML is intended to be a universal general-purpose modeling language for discrete systems such as those made of software, firmware, or digital logic.

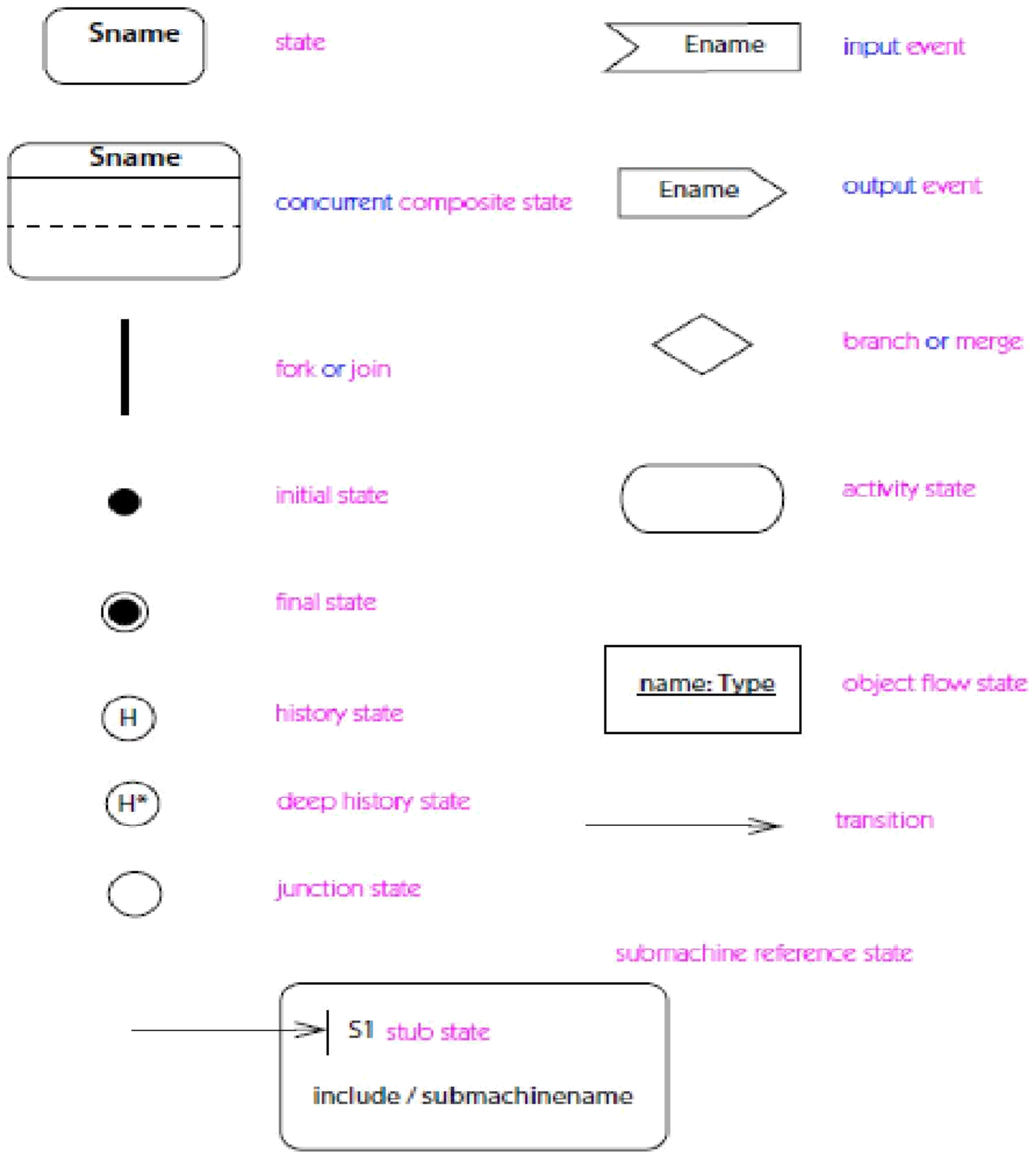
Icons on use case diagrams



Icons on class, component, deployment, and collaboration diagrams


Cname
attr: Atype
op (par: Type): Rtype
class
Cname
component
Cname
active class
Nname
node
name: Class
role
text
note
oname: Class[Role]
object
Pname
package
oname: Class
multiobject
Iname
interface
Cname
collaboration
Aname
association
generalization
template
parameter
p:Type
realization
Tname
template
«kind»
dependency
{ expression }
constraint

Icons on statechart and activity diagrams


Sname
state
Ename
input event
Sname
concurrent composite state
Ename
output event
branch or merge
fork or join
initial state
activity state
final state
name: Type
object flow state
H
history state
H*
deep history state
transition
junction state
submachine reference state
S1 stub state
include / submachinename