A Project Report on

REAL TIME OBJECT DETECTION THROUGH DEEP LEARNING -SCALED YOLO V4

Submitted in partial fulfilment of the requirements for the award of the degree in

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

VELAGA BHARGAV	**18KP1A0594**
SHAIK AYESHA	**18KP1A0585**
RAVI VASAVI	**18KP1A0576**

Under the esteemed guidance of

GANDLA SOWMYA, MTech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NRI INISTITUTE OF TECHNOLOGY

(APPROVED BY AICTE, AFFILIATED TO JNTUK, KAKINADA)

Visadala(p), Medikonduru(M), GUNTUR-522438

2018 – 2022

NRI INSTITUTE OF TECHNOLOGY

(APPROVED BY AICTE, AFFILIATED TO JNTUK, KAKINADA)

VISADALA, PERECHERLA, GUNTUR-522438

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"REAL TIME OBJECT DETECTION THROUGH DEEPLEARNING – SCALED YOLO V4"** being submitted by **VELAGA BAHRGAV (18KP1A0594)**, **SHAIK AYESHA (18KP1A0585)** and **RAVI VASAVI (18KP1A0576)** in the partial fulfilment for the award of degree of Bachelor of Technology in Computer Science and Engineering, NRI Institute of Technology, affiliated to Jawaharlal Nehru Technology University Kakinada is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2018-2022.

INTERNAL GUIDE	**HEAD OF DEPARTMENT**
G. SOWMYA	**DR. J. CHANDRA SEKHAR**
Assistant Professor	Professor & HOD

External Examiner

DECLARATION

We hereby declare that the project report titled **"REAL TIME OBJECT DETECTION THROUGH DEEP LEARNING – SCALED YOLO V4"** under the guidance of **G. SOWMYA**, Assistant Professor is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project have not been submitted to any other university for the award of any other degree.

Project Members:

VELAGA BHARGAV	**18KP1A0594**
SHAIK AYESHA	**18KP1A0585**
RAVI VASAVI	**18KP1A0576**

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to our esteemed "**NRI INSTITUTE OF TECHNOLOGY**", which has provided an opportunity to fulfil our cherished desire.

We sincerely thank **Dr. K. SRINIVASU,** Principal, for his valuable suggestions in due to completion of the course.

We are much thankful to **Dr. J. CHANDRA SEKHAR, Head of the CSE Department**, for his constant encouragement and for sparing time for us from his hectic schedule and without which for his motivation during the project work.

Our whole hearted thanks to **G. SOWMYA,** our project guide, for her untiring guidance and valuable suggestions without which we could not have made this effort a success.

We would like to thank all the members of teaching and non-teaching staff of **Computer Science and Engineering** department for all their support in completion of this project.

Project Members:

VELAGA BHARGAV	**18KP1A0594**
SHAIK AYESHA	**18KP1A0585**
RAVI VASAVI	**18KP1A0576**

ABSTRACT

Object detection is an advanced form of image classification where a neural network predicts the objects in an image and points them out in the form of bounding boxes. Object detection thus refers to the detection and localization of objects in an image that belong to a predefined set of classes. Tasks like detection, recognition, or localization find widespread of applicability in real-world scenarios, making object detection (also referred as object recognition) a very important subdomain of Computer Vision which is characterized by its strong capability of feature learning and feature representation compared with the traditional object detection methods. Real time object detection and tracking are important and challenging tasks in many computer vision applications such as video surveillance, robot navigation and vehicle navigation. Object detection involves detecting the object in sequence of videos.

Every tracking mechanism requires object detection technique either in each frame or when an object appears newly on the video sequence. Object tracking is the process of locating an object or multiple objects using either static or dynamic camera. The availability of high-powered computers, high quality and inexpensive video cameras will increase need for automated video analysis. It has generated a great deal of interest in object detection and tracking algorithms. Even though high-powered computers are used for object detection and tracking algorithm, most of the object detection algorithms such as background subtraction, temporal difference, foreground extraction and simple differencing requires long time to detect object, requires more storage space and no robustness against illumination changes. Recently computer vision research has addressed the Multiple objects detection and tracking in dynamic environment.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 01

INTRODUCTION

Object detection typically leverage deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection to replicate this intelligence using a computer. Object detection in one of the most basic and central tasks in computer vision. Its task is to find all the interested objects in the image and determine the category and location of objects. It is widely used and has strong practical value and research prospects.

1.1 APPLICATIONS OF OBJECT DETECTION

Object detection has already been the significant research direction and the focus in the computer vision, which can be applied in the driverless cars, robotics, video surveillance and pedestrian detection. The emergence of deep learning technology has changed the traditional modes of object identification and object detection. The emergence of deep learning technology has changed the traditional modes of object identification and object detection.

Image net Large Scale Visual Recognition Challenge (ILSVRC):

In the Image net Large Scale Visual Recognition Challenge (ILSVRC) 2012, a global contest about computer vision, the Alex Net won the championship, which was the first successful deep convolution network in image recognition and its top5 accuracy surpassed the runner-up by 10%.

Moreover, the methods of deep learning topped the list in succeeding the LSVRCs.

Deep learning sub type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher-level features from data.

Neural networks, also known as Artificial Neural Networks (ANNs) or Simulated Neural Networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Object Detection:

Locate the presence of objects with a bounding box and types or classes of the located objects in an image

- Input: An image with one or more objects, such as a photograph.
- Output: One or more bounding boxes (e.g., defined by a point, width, and height), and a class label for each bounding box.

The performance of a model for image classification is evaluated using the mean classification error across the predicted class labels. The performance of a model for single-object localization is evaluated using the distance between the expected and predicted bounding box for the expected class. Whereas the performance of a model for object recognition is evaluated using the precision and recall across each of the best matching bounding boxes for the known objects in the image.

Now that we are familiar with the problem of object localization and detection, let's take a look at some recent top-performing deep learning models.

Problem Statement:

To achieve accuracy in the models the existing system proposed CNN, RCNN, SSD models but those models limited to low accuracy with higher time rate to detect objects which results in inefficiency.

Proposed System:

In our research and literature survey the Scaled You Only Look Once (YOLO) V4 models not only works fine but also overcomes all the problems faced in the previous existing systems.

A Gentle Introduction to Object Recognition with Deep Learning

It can be challenging for beginners to distinguish between different related computer vision tasks. For example, image classification is straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition.

Object detection 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 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 of these problems are referred to as object

recognition. In this, we can discover a gentle introduction to the problem of object recognition and state-of-the-art deep learning models designed to address it.

1.2 Object Recognition

What is Object Recognition?

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in digital photographs.

Image classification involves 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 abounding box around their extent. Object detection combines these two tasks and localizes and classifies one or more objects in an image.

When a user or practitioner refers to “object detection”, they often mean “object detection “.

we will be using the term object recognition broadly to encompass both image classification (a task requiring an algorithm to determine what object classes are present in the image) as well as object detection (a task requiring an algorithm to localize all objects present in the image).

As such, we can distinguish between these three computer vision tasks:

- **Image Classification**: Predict the type or class of an object in an image.
  - Input: An image with a single object, such as a photograph.
  - Output: A class label (e.g., one or more integers that are mapped to class labels).

- **Object Localization**: Locate the presence of objects in an image and indicate their location with a bounding box.
  - Input: An image with one or more objects, such as a photograph.
  - Output: One or more bounding boxes (e.g., defined by a point, width, and height).

- **Object Detection**: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
  - Input: An image with one or more objects, such as a photograph.

- Output: One or more bounding boxes (example: defined by a point, width, and height), and a class label for each bounding box.

One further extension to this breakdown of computer vision tasks is Object Segmentation, also called "Object Instance Segmentation" or "Semantic Segmentation," where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box. From this breakdown, we can see that object recognition refers to a suite of challenging computer vision tasks.

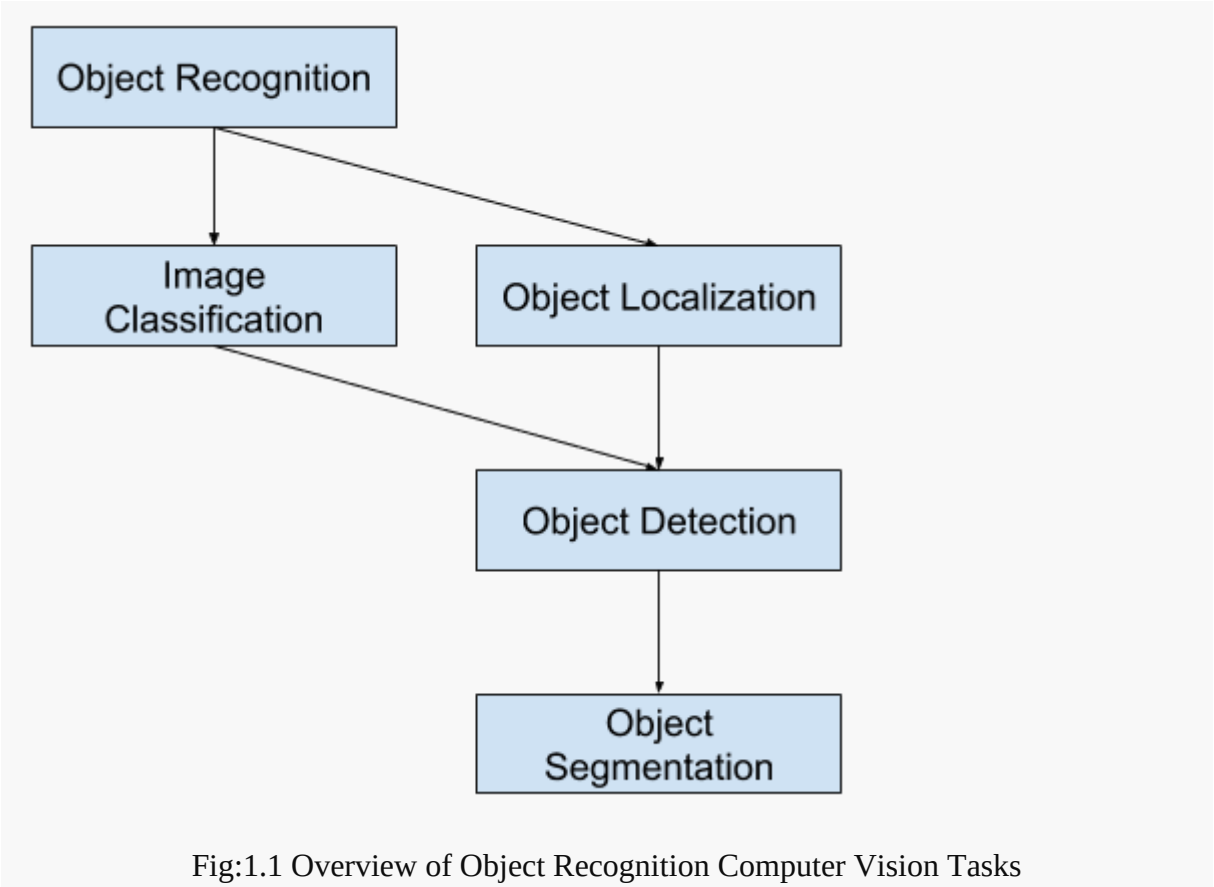

Fig:1.1 Overview of Object Recognition Computer Vision Tasks

Most of the recent innovations in image recognition problems have come as part of participation in the ILSVRC tasks.

This is an annual academic competition with a separate challenge for each of these three problem types, with the intent of fostering independent and separate improvements at each level that can be leveraged more broadly. For example, see the list of the three corresponding task types below taken from them

- **Image classification**: Algorithms produce a list of object categories present in the image.
- **Single-object localization**: Algorithms produce a list of object categories present in the image, along with an axis-aligned bounding box indicating the position and scale of one instance of each object category.
- **Object detection**: Algorithms produce a list of object categories present in the image along with an axis-aligned bounding box indicating the position and scale of every instance of each object category.

We can see that "Single-object localization" is a simpler version of the more broadly defined "Object Localization," constraining the localization tasks to objects of one type within an image, which we may assume is an easier task.

Below is an example comparing single object localization and object detection, taken from the ILSVRC paper. Note the difference in ground truth expectations in each case.

Comparison Between Single Object Localization and Object Detection. Taken From: ImageNet Large Scale Visual Recognition Challenge.

The performance of a model for image classification is evaluated using the mean classification error across the predicted class labels. The performance of a model for single-object localization is evaluated using the distance between the expected and predicted bounding box for the expected class. Whereas the performance of a model for object recognition is evaluated using the precision and recall across each of the best matching bounding boxes for the known objects in the image.

Now that we are familiar with the problem of object localization and detection, let's take a look at some recent top-performing deep learning models.

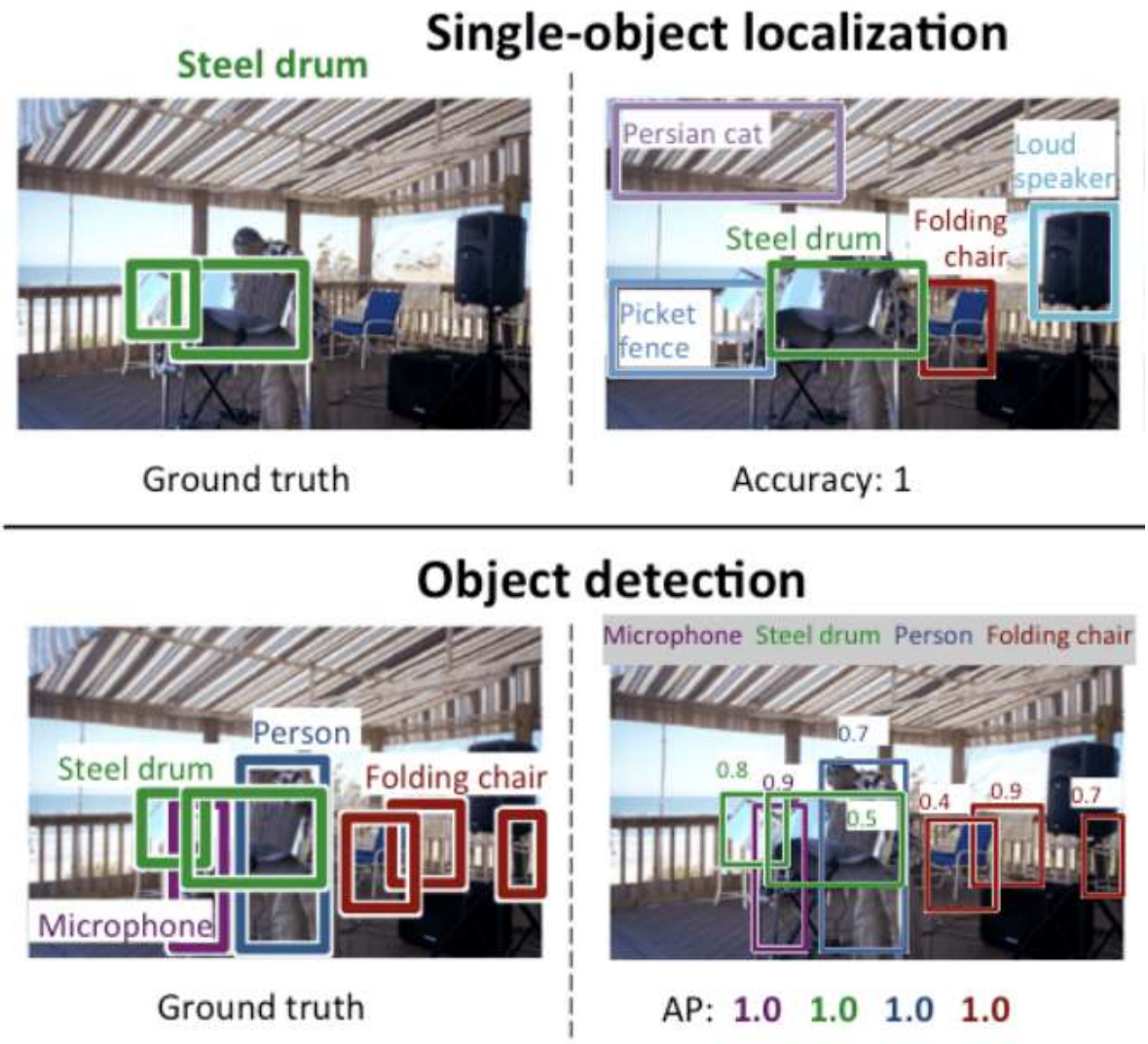

Fig: 1.2 Single-Object Detection, Object Detection

1.3 Aim of the Project

The project focuses on developing a model for detecting the objects in frame either it will be in image or video format, in order to resolve the issues occurred in previous ones such as

- Speed
- Accuracy
- Learning capabilities
- Error rate

The main purpose of object detection is **to identify and locate one or more effective targets from still image or video data**. It comprehensively includes a variety of important techniques, such as image processing, pattern recognition, artificial intelligence and machine learning.

1.4 Objective

The range of object detection is sky rocketing on daily basis in everyday life. The intent of our project is developing a model to detect objects with fast processing, high accuracy and more learning capabilities.

It can be achieved by implementing a deep learning algorithm named YOLO algorithm which stands for You Only Look Once.

CHAPTER 02

LITERATURE SURVEY

2.1 LITERATURE REVIEW

Several methods have been used in the literature for fire detection. Similar methods of object detection have been followed. By many authors which are: using the same YOLO models but with outdated ones. Further analyzing are applied to detect the object characteristics. In this work, a modern method of deep learning is used for real-time object detection depending on the base level of accuracy. A model is said to be legitimate when it detects the real-time object with fast inference while maintaining a base level of accuracy.

Numerous learning machine like Convolutional Neural Networks (CNN), Region based Convolutional Neural Networks (RCNN) and also various outdated YOLO's were used but, at last they are all results in the low accuracy and took more time to detect objects.

1. **Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi "You Only Look Once: Unified, Real-Time Object Detection" 2016 YOLO V1**

   A fast and simple approach to detecting real time images was introduced in this paper as You Only Look Once. The model was built to detect images accurately, fast and to differentiate between art and real images. In comparison with Object detection techniques that came before YOLO, like R-CNN, YOLO introduced a single unified architecture for regression go image into bounding boxes and finding class probabilities for each box. This meant that YOLO performed much faster and also provided more accuracy. It could also predict artwork correctly.

2. **Chengji Liu, Yufan Tao, Jiawei Liang, Kai Li1, Yihang Chen "Object Detection on YOLO Network" 2018 YOLO V3**

   A generalized object detection network was developed by applying complex degradation processes on training sets like noise, blurring, rotating and cropping of images. The model was trained with the degraded training sets which resulted in better generalizing ability and higher robustness. The experiment showed that the model trained with the standard sets does not have good generalization ability for the degraded images and has poor robustness. Then the model was trained using degraded images which resulted in improved

average precision. It was proved that the average precision for degraded images was better in general degenerative model compared to the standard model.

3. **Wenbo Lan, Jianwu Dang, Yang-ping Wang, Song Wang "Pedestrian Detection Based on YOLO Network Model" 2018 YOLO V3**

   The network structure of YOLO algorithm is improved and a new network structure YOLO-R was proposed to increase the ability of the network to extract the information of the shallow pedestrian features by adding passthrough layers to the original YOLO network. The YOLO v2 and YOLO-R network models were tested on the test set of the INRIA data set. The experimental results show that the YOLO-R network model is superior to the original YOLO v2 network model. The number of detection frames reached 25 frames/s, basically meeting the requirement of real-time performance.

4. **Y. Lecun, L. Bottou, Y. Bengio, and P. Haơner," Gradient-based learning applied to document recognition," Proceedings of the IEEE, Nov 1998, 86(11), pp. 2278-2324.**

   Multilayer neural networks trained with the back-propagation algorithm constitute the best example of a successful gradient-based learning technique. Given an appropriate network architecture, gradient-based learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns, such as handwritten characters, with minimal pre-processing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional neural networks, which are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.

5. **Rumin Zhang, Yifeng Yang "An Algorithm for Obstacle Detection based on YOLO and Light Filed Camera" 2018 YOLO V3**

   An obstacle detection algorithm in the indoor environment is proposed which combines the YOLO object detection algorithm and the light field camera and will classify objects into categories and mark them in the image. The images of the common obstacles were labelled and used for training YOLO. The object filter is applied to remove the unconcern obstacle. Different types of scenes, including pedestrian, chairs, books and so on, are demonstrated to prove the effectiveness of this obstacle detection algorithm

6. **Zhimin Mo1, Liding Chen1, Wen-jing You "Identification and Detection of Automotive Door Panel Solder Joints based on YOLO" 2019 YOLO V4**

A method for identifying the solder joints of automotive door panels based on YOLO algorithm that provides the type and location of solder joints in real time. For detecting the small solder joints more precisely, this paper adopts YOLO algorithm which adopts multi-level predictions, predicting on different size feature maps and combining the prediction results to obtain the final result. The YOLO algorithm, proposed identifies the position of the solder joints accurately in real time. This is helpful to increase the efficiency of the production line and it has a great significance for the flexibility and real-time of the welding of automobile door panels.

CHAPTER 03

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS AND FEASIBILITY STUDY

This chapter includes, existing system, proposed system, methodologies (or) algorithms, software development life cycle, feasibility study.

The problems identified in the literature survey are as follows:

- It required to use more and outdated algorithms.
- It takes more time for execution and checking.

These problems have been carried out for particular feasible solution.

3.1 EXISTING SYSTEM

Earlier system works involved models are by using CNN and RCNN in object detection system. In Existing system object detection is performed using a combination of Convolutional Neural Networks (CNN) and Recurrent Neural Network which are later compared with Long Short-Term Memory (LSTM) the results obtained by using these higher than those of the previous models but are not satisfactory.

Convolutional neural networks (CNNs) are likely to extract local and deep features from natural language. It has been shown that CNN has gotten improved results in sentence classification (Kim, 2014). Recurrent neural networks (RNNs) are various kinds of time- recursive neural network that is able to learn the long-term dependencies in sequential data. Seeing that we can view the words in a sentence as a sequence from left to right, RNNs can be modelled in accordance with people's reading and understanding behaviour of a sentence.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- The existing system suffers from low accuracy.
- Time consumption for Training is high as it involves various algorithms.
- Overall time complexity of execution is high.

3.2 PROPOSED SYSTEM

It is rare and more productive in order to deal with intensive kind of image processing, there is no need of training this model unlike the competitive ones makes it more feasible and legible.

You Only Look Once (YOLO):

YOLO models are also called as single-shot models. The first model would process 45 frames per second in real-time using features from the entire image to predict bounding boxes. YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of the image, it predicts classes and bounding boxes for the whole image in one run of the algorithm using single neural network.

Table: 3.1 YOLO Versions

YOLO Version	Year of Release	Developers	Development
YOLO V1	2016	Joseph Redmon	Unified, Real-time detection
YOLO V2	2017	Joseph Redmon, Ali Farhadi	Better, Stronger, faster
YOLO V3	2018	Joseph Redmon, Ali Farhadi	An incremental improvement
YOLO V4	April 2020	Alexey Bochkovsky	Optimal Speed and Accuracy
YOLO V5	May 2020	Glenn Jocher	Similar to YOLO V4

Challenges in initial implementation

- Only predict one class.
- Inefficiency in small objects.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- High accuracy
- Low time complexity
- Optimized results
- Higher execution speeds

3.3 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Four key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility
- Operational Feasibility

3.3.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.3.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he

is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.3.4 Operational Feasibility

The project is operationally feasible as the user having basic knowledge about computer and Internet. Mask Predictor is based on client-server architecture where client is users and server are the machine where datasets are stored.

3.4 SOFTWARE DEVELOPMENT LIFE-CYCLE

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and life cycle management of an application or piece of software. SDLC is the process consisting of a series of planned activities to develop or alter the software products.

Benefits of the SDLC Process

The intent of a SDLC process it to help produce a product that is cost-efficient, effective, and of high quality. Once an application is created, the SDLC maps the proper deployment and decommissioning of the software once it becomes a legacy.

The SDLC methodology usually contains the following stages: Analysis (requirements and design), construction, testing, release, and maintenance (response). Veracode makes it possible to integrate automated security testing into the SDLC process through use of its cloud-based platform.

Requirement Gathering:

In this phase we gather all the requirements from the client, i.e., what are the client expected input, output……

Analysis:

In this phase based upon the client requirements we prepare one documentation is called “High Level Design Document”. It contains Abstract, Functional Requirements, Non- Functional Requirements, Existing System, Proposed System, SRS ……

Design:

It is difficult to understand the high-level Design Document for all the members understand easily we use “Low Level Design Document”. To design this document, we use UML (Unified Modelling Language). In this we have Use case, Sequence, Collaboration…….

Coding:

In this phase we develop the coding module by module. After developing all the modules, we integrate them.

Testing:

After developing we have to check weather client requirements are satisfied or not. If not, we are again going to develop.

Implementation:

After developing we have to check weather client requirements are satisfied or not. If not, we are again going to develop.

Maintenance:

After deployment, if at all any problems come from the client side; we are providing maintenance for that application.

CHAPTER 04

SYSTEM REQUIREMENT SPECIFICATIONS

4.1 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form "system shall do". The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is:

user/stakeholder request -> feature -> use case -> business rule

Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. Functional requirements may be technical details, data manipulation and other specific functionality of the project is to provide the information to the user.

The following are the Functional requirements of our system:

- We are providing one query then we will get efficient result.
- The search of query is based on major intention of user.
- We are having the effective ranking methodology.
- A novel framework to exploit the user's social activities for personalized image search, such as annotations and the participation of interest groups.

4.1.1 USER REQUIREMENT ANALYSIS

User Requirement Analysis is the process of determining user expectations for a new one when the dataset is uploaded. These features must be quantifiable, relevant and detailed. The main user requirements of our

project are as follows.

- Upload the dataset
- Give image Path to recognize

4.2 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is detailed in the system architecture.

The non-functional requirements are "system shall be ".

Non-functional requirements are often called qualities of a system. The following are the Non-functional requirements for our system:

AVAILABILITY: A system's "availability" or "uptime" is the amount of time that is operational and available for use. It's related to is the server providing the service to the users in displaying images. As our system will be used by thousands of users at any time our system must be available always. If there are any cases of updating, they must be performed in a short interval of time without interrupting the normal services made available to the users.

EFFICIENCY: Specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth etc. All of the above-mentioned resources can be effectively used by performing most of the validations at client side and reducing the workload on server by using JSP instead of CGI which is being implemented now.

PORTABILITY: Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run. By using appropriate server versions released for different platforms our project can be easily operated on any operating system, hence can be said highly portable.

SCALABILITY: Software that is scalable has the ability to handle a wide variety of system configuration sizes. The non-functional requirements should specify the ways in which the system may be expected to scale up (by increasing hardware capacity, adding machines etc.). Our system can be easily expandable.

Any additional requirements such as hardware or software which increase the performance of the system can be easily added. An additional server would be useful to speed up the application

INTEGRITY: Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software. Certain features access must be disabled to normal users such as adding the details of files, searching etc. which is the sole responsibility of the server. Access can be disabled by providing appropriate logins to the users for only access.

USABILITY: Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users. Hyperlinks will be provided for each and every service the system provides through which navigation will be easier. A system that has high usability coefficient makes the work of the user easier.

PERFORMANCE: The performance constraints specify the timing characteristics of the software. Making the application form filling process through online and providing the invigilation list information and examination hall list is given high priority compared to other services and can be identified as the critical aspect of the system.

• In our system introduced user specific detection performance.

• The query related detection is effective it provides within short period results, so the speed of system is very high.

RESPONSE TIME: Response time is the time a system or functional unit takes to react to a given input.

IMPLEMENTATION: The implementation of the project is, it is implemented by using the SCALED YOLO V4. It gives the best accuracy when they are applied in the model.

4.3 SYSTEM REQUIREMENTS

4.3.1 SOFTWARE REQUIREMENTS:

- Operating System : Windows 7 Ultimate or above
  Linux
  MAC OS
- Coding Languages : Python, JavaScript
- Note books : Google colaboratory, Kaggle
- Packages : matplotlib, CV2, numpy, Python Imaging Library (PIL), html

4.3.2 HARDWARE REQUIREMENTS:

- Processor : intel i3 or above
  AMD ryzen 3500 or above
- Ram : 8GB DDR4 or higher
- Hard Disk : 500GB or higher
- GPU : Integrated or Dedicated GPU’s
- Webcam : Standard camera

CHAPTER 05
METHDOLOGY

5.1 DATA COLLECTION

Data collection has been collected from the internet. An opensource dataset named Common Objects in Context (COCO) is a database that aims to enable future research for object detection, instance segmentation, image captioning and person keypoints localization which operates by Microsoft.

5.2 ALGORITHM

The yolo algorithm works by dividing the image into N grids, each having an equal dimensional region SxS. Each of these N grids is responsible for the detection and localization of the object it contains.

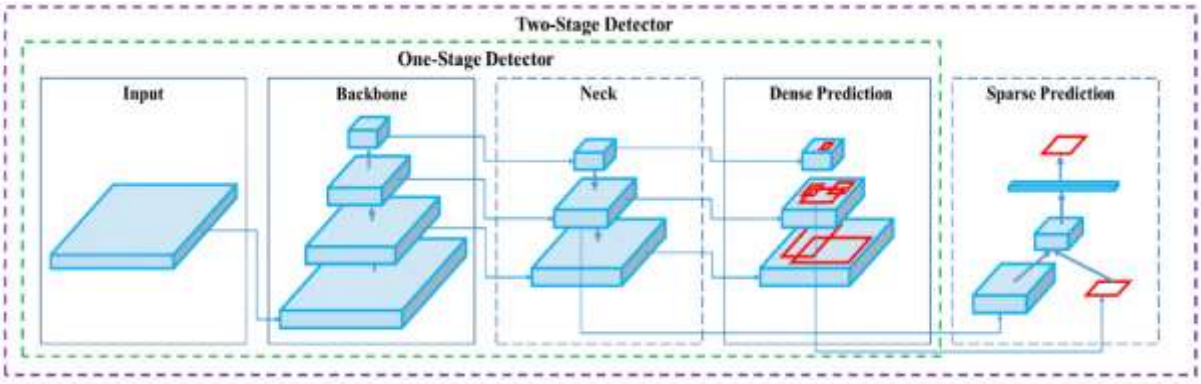

Fig: 5.1 Architecture

Basically, object detector algorithm consists of four different components. They are:

Input: The input to the detector can be an image or video based on the use cases specified in the research.

Backbone: The backbone of the object detector contains models, these models can be Resnet, Dense net, YOLO.

Neck: The neck in the detector acts as an extra layer, which goes in parallel to the backbone and the head.

Head: The head is the network that is in charge of the detection of objects based on bounding boxes.

- Generally, the detectors create bounding boxes around the objects in the image or in a particular frame of a video as shown aside.

- Ground truth bounding box:
  The hand labelled bounding boxes from the testing set that specify where in the image our object is.
- Predicted bounding box:
  A bounding box being detected by the model itself.
- The two common evaluation metrics are Intersection Over Union (IOU) and Average Precision (AP).

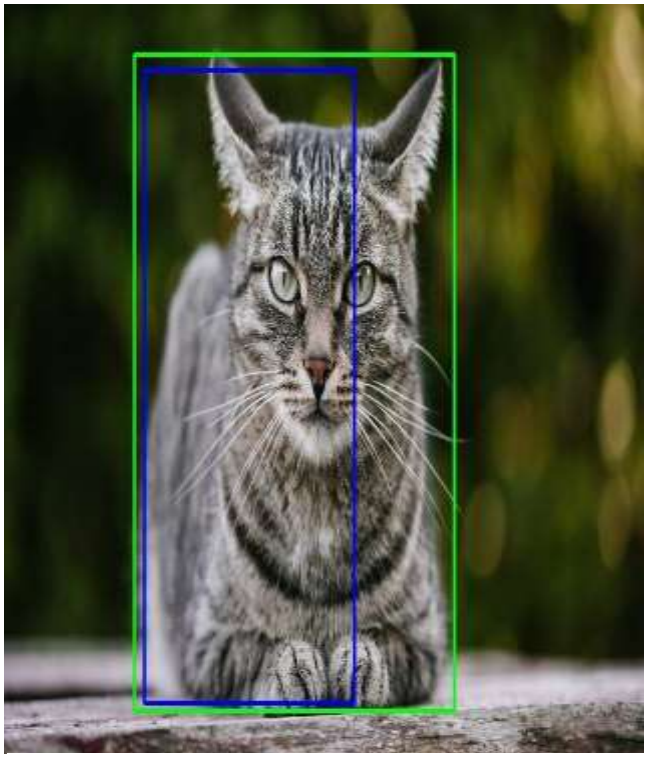

Fig: 5.2 Bonding Box

- The IOU is the overlap between ground truth bounding boxes and the predicted bounding boxes.
- The resulting value is a number between 0 and 1. The higher the number, the higher they overlap.
- Mean Average Precision (MAP) is a metric to measure the performance efficiency.

5.3 ALGORITHM IMPLEMENTATION

Initially, the raw image is taken as given as Fig 5.3.0.1 and then the model Extracts the features from the image by following steps.

Step 1:

The dataset for sample will be limited to only 3 objects for better understanding.

They are C1. Pedestrian
C2. Car
C3. Motorcycle

Fig: 5.3 Sample

Step 2:

The input image is divided into some SxS grid. Here the sample image is classified into 3x3 grid. Shown in the below figure.

Fig: 5.4

Step 3:

Consider the grids as y matrix which values: y = 3x3x2x8

Y= 3x3x16

where,

3x3 = number of grids

2 = number of anchor boxes

8 = one set of parameters in matrix

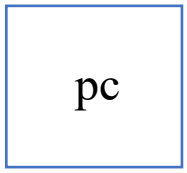

Fig: 5.5 anchor box(square)

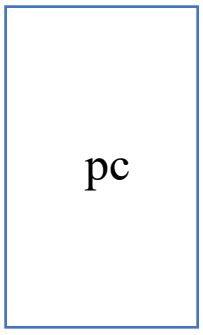

Fig: 5.6 Anchor box(rectangle)

Here, anchor boxes represent the bounding boxes detected by model in an image. In this case anchor boxes are Square and Rectangle.

The 8 attributes which are used mainly to detect object are

pc = anchor box

bx, by = x & y coordinates of the object

bh = height of the object

bw = width of the object

c1, c2, c3 = classes in dataset used to identify the object

Later, by reviewing all these attributes our model forms a matrix consisting 16 elements in total are shown below

$$y=\begin{pmatrix} Pc \\ bx \\ by \\ bh \\ bw \\ c1 \\ c2 \\ c3 \\ pc \\ bx \\ by \\ bh \\ bw \\ c1 \\ c2 \\ c3 \end{pmatrix}$$

Step 4:

The model starts from the first grid and moves towards the last grid by checking for the objects.

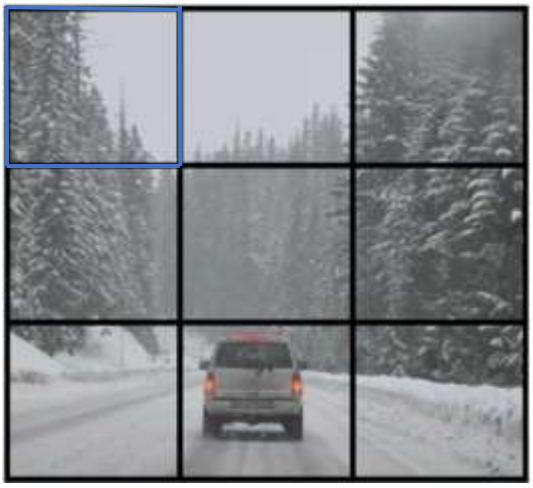

Fig:5.7

$$y = \begin{pmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix}$$

Here in the first grid, there is no object which results in the empty matrix by denoting the anchor boxes as zeros and remaining all with some garbage values.

- By moving from the first grid as mentioned above model carrion with the other grids as same as it deals with the first grid.
- But when comes to a grid which consists of an object then the process will differ from the above ones without objects.

Step 5:

By moving from grids without objects to with objects the flow of the model will change as shown below.

After reaching the 8th grid the model searches for the object as usually but the overall scenario was changed after detecting the object in that grid.

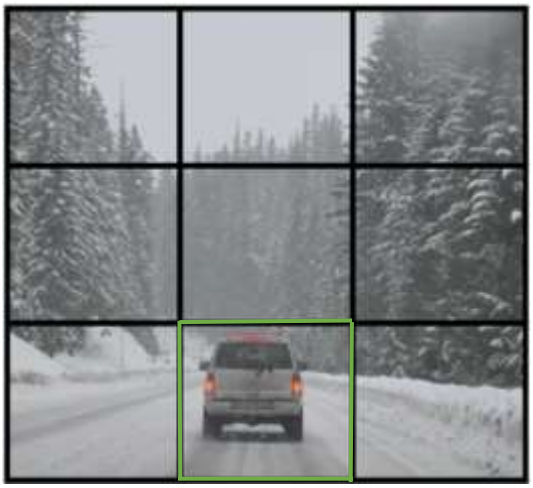

Fig: 5.8

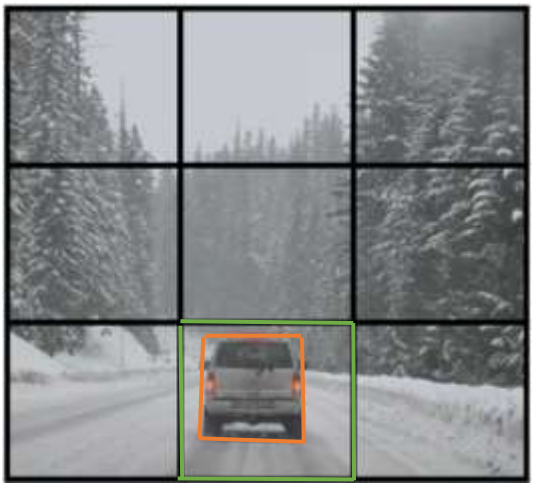

Fig: 5.9

After detecting the object in the 8th grid the model again creates the matrix in order to find the name of the object.

$$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ bx \\ by \\ bh \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Here, the object fits only into the square type of anchor box.

So, that's why second part of the matrix is only been filled with the respective values.

As mentioned above bx, by are x & y coordinates of object whereas bh, bw are height and width of object.

Among all the classes **c1, c2, c3 – c2** is the only stored with value 1 which means true by stating that the object **car**.

The object name can be finalized by referring with the dataset taken initially.

LANGUAGES AND TOOLS USED:

Our project will utilize cv2, numpy, PIL, io, html, time, matplotlib, base64 and several packages present in google colabs.

Google colaboratory: A cloud-based IDE by google.

CHAPTER 06

DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analysed, system design is the first of the three technical activities design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

6.1 INPUT AND OUTPUT DESIGN

6.1.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a

way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur. In out proposed system, the input is human face which can be taken from live video stream.

6.1.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

In our system, it provides live output of a feeding from any kind of camera linked to the system and detects the objects shown in Infront of the system.

6.2 UML DIAGRAMS

A diagram is a graphical presentation of a set of elements, most often is rendered as a graph of vertices and arcs. A UML diagram can contain nine types in it where vertices are denoted as things and arcs are denoted as relationships.

The Unified Modelling Language diagram is that is, the unified modelling language is probably the most widely known and used notation for object-oriented analysis and design. The Unified Modelling Language is used to visualize, specify, construct and document the artifacts. A Modelling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. Modelling is the designing of software applications before coding. According to uml no one diagram can capture the different elements of a system in its entirety. Hence uml is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system in its entirety. Hence uml is made

up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system. The nine diagrams include:

- Use case diagram
- Class diagram
- State diagram
- Activity diagram
- Sequence diagram
- Collaboration diagram
- Component diagram
- Deployment diagram

6.2.1 USECASE DIAGRAM

Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse**.**

Actors: An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

System boundary boxes (optional): A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

Include: In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviours from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behaviour of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

Extend: In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behaviour of the extension

use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

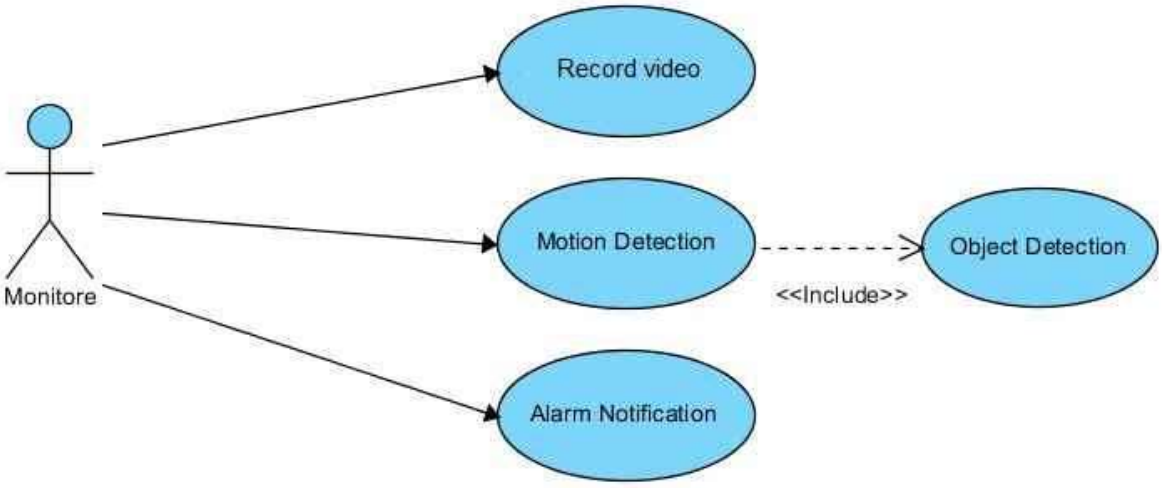

Fig: 6.1 Use case Diagram

6.2.2 ACTIVITY DIAGRAMS

An activity diagram is a behavioural diagram i.e., it depicts the behaviour of a system. This is to illustrate the flow of control in a system and refer to the steps involved in the execution of a use on condition case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens.

Step 1: Figure out the action steps from the use case

Here you need to identify the various activities and actions your business process or system is made up of.

Step 2: Identify the actors who are involved

If you already have figured out who the actors are, then it’s easier to discern each action they are responsible for.

Step 3: Find a flow among the activities

Figure out in which order the actions are processed. Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time and whether you need to add any branches in the diagram. And do you have to complete some actions before you can proceed to others?

Step 4: Add swimlanes

You have already figured out who is responsible for each action. Now it's time to assign them a swimlane and group each action they are responsible for under them.

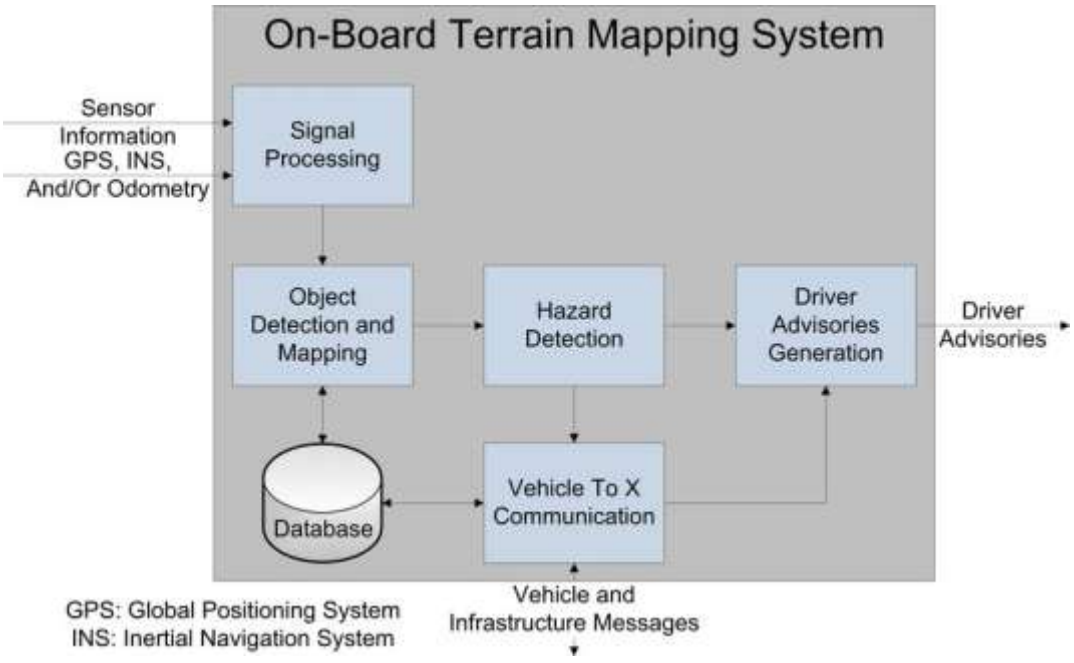

Fig: 6.2 Activity Diagram

6.2.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and

in what order. It is a construct of a Message Sequence Chart. The important aspect of the sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing the “messages”.

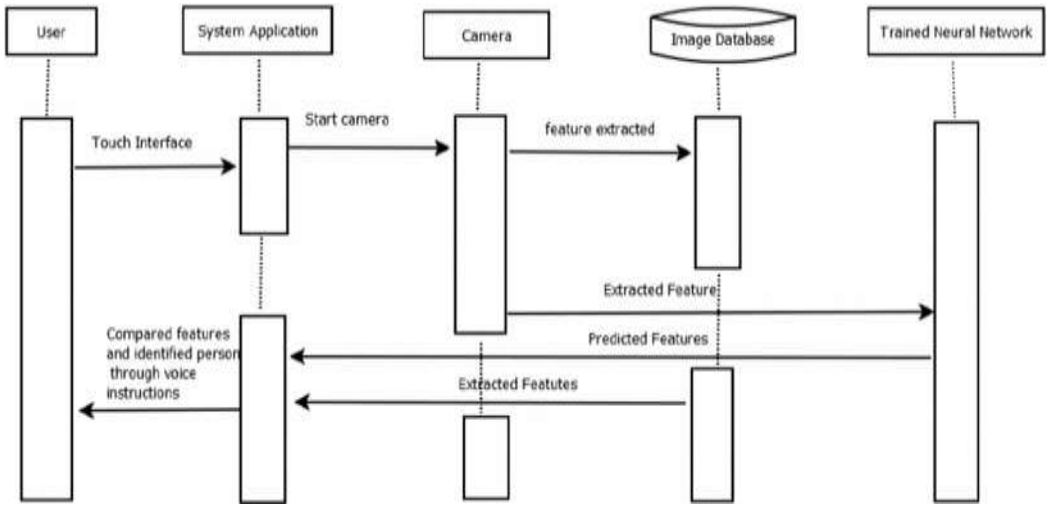

Fig: 6.3 Sequence Diagram

6.2.4 COMPONENT DIAGRAM

Object Management Group OMG UML defines a component as “a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces.”

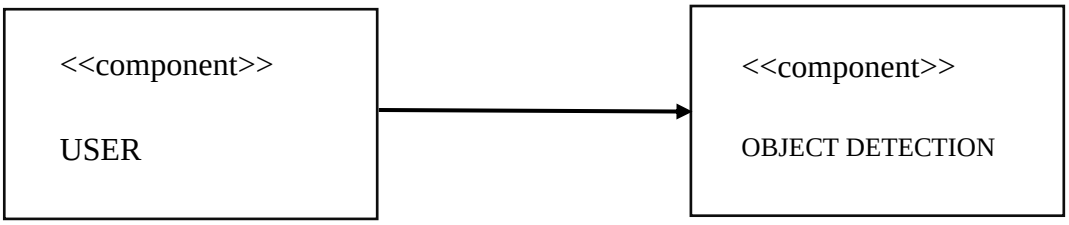

Fig: 6.4 Component Diagram

6.2.5 DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So, deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

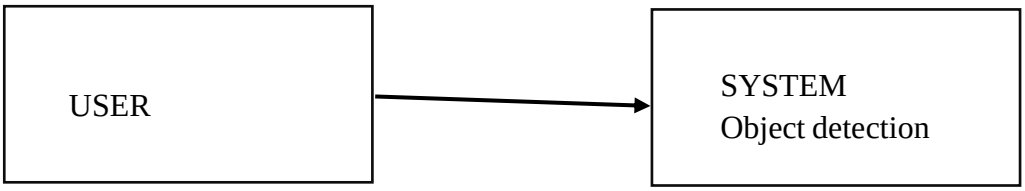

Fig: 6.5 Deployment Diagram

CHAPTER 07
IMPLEMENTATION

7.1 INTRODUCTION

Google Colaboratory:

Colaboratory or "Colab" for short, is a product from google research. Colab allows anybody to write and execute arbitrary python code through the browser. It is especially well suited to machine, data analysis and education. More technically, Colab is a hosted jupyter notebook service that requires no setup to use, while providing access to free of charge to computing resources including GPU's.

Unlike other notebooks, Colab provide numerous features to its users like by providing virtual GPU's, no need to install required libraries, Security. Simply any person with internet in any part of the world can access their team members project conveniently. But there are some restrictions while sharing the project depending on the levels of security.

Google Colab features:

- Platform independency
- User-friendly
- No need of installing packages
- Security

The main reason Colab for creation of this IDE was for machine learning and python programming. As Colab is a cloud-based notebook, we can run the project on various platforms like windows, Linux, macOS and even in android mobile phones too. The IDE comprises code analysis tools, debugger testing tools, and also version control options. It allows us to work with several databases directly without getting it integrated with other tools. Although, it is specially designed for Python, JavaScript, html files can also be implemented within this notebook. It also comes with the beautiful user-friendly interface that can be customized according to the need using plugins.

Google Colab is extremely popular for its new technology implementations in its environment. A notebook features a code editor and compiler for writing and compiling programs in one or many programming languages.

Furthermore, a notebook comes with a galore of features that facilitate comprehensive software development. This notebook allocates different colors to different programming entities, typically known as syntax highlighting, it becomes more accessible to:

- Differentiate between various programming entities such as a class and a function, and to spot them.
- Look for the wrong keywords.
- Read and comprehend the code.

Most notebook feature an autocomplete and auto save features that produce suggestions when writing code. This makes writing code more efficient, quick and less prone to errors and typos.

Few packages needed for python programming

numpy:

NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. It is also useful in linear algebra, random number capability etc.,

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 the year 2002.

7.2 CODING (SOURCE CODE)

This code will walk through all the steps for performing YOLOv4 Real-time object detections on your webcam while in Google Colab. We will be using scaled-YOLOv4 (yolov4-csp) which is the fastest and most accurate object detector there currently shown in the below graph.

The below graph depicts us the accuracy differences between various algorithms when they all are connected to the tesla v100 GPU in Colab notebook and plotted for average precision(accuracy) and batch latency. Among all of them Scaled-YOLO V4 performs more efficiently than others. So, we chose Scaled-YOLO V4 csp 640x640.

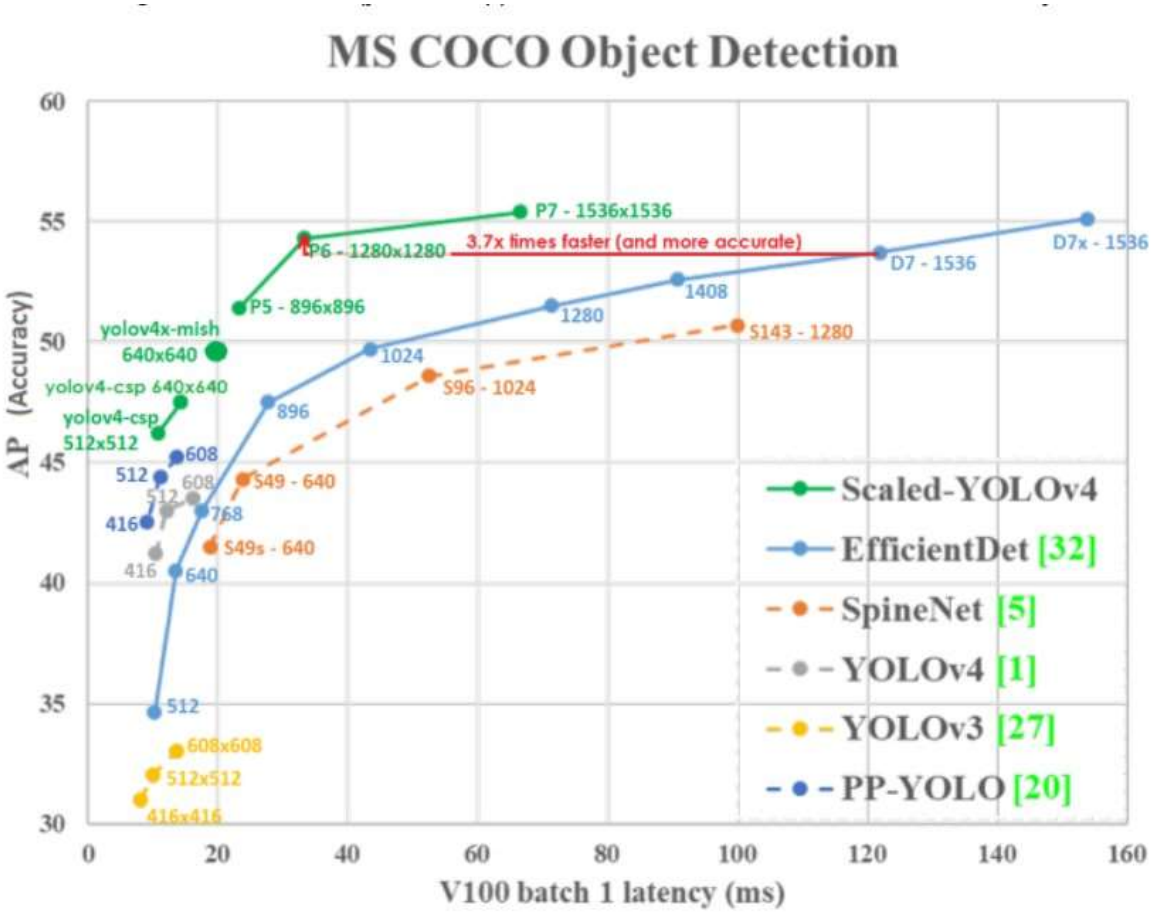

Fig: 7.1 Accuracy and batch latency graph

Virtual GPU

As Colab provides virtual GPU, in order to know the name of the GPU which is being assigned, we need to run the following command.

!nvidia-smi

```
Mon May 30 12:52:34 2022 +-----------------------------------------------------------------------------+ | NVIDIA-
SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2
| |-------------------------------+----------------------+----------------------+ |
GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
|                               |                      |                      MIG                     M.                      |
|===============================+======================+======================
==|
| 0 Tesla T4 Off | 00000000:00:04.0 Off | 0 |
 | N/A 70C P8 11W / 70W | 0MiB / 15109MiB | 0% Default |
| | | N/A |
+-------------------------------+----------------------+----------------------+  +-----------------------------------------
--------------------------------+
 | Processes: |
 | GPU GI CI PID Type Process name GPU Memory |
```

```
|                  ID                  ID                  Usage                  |
|=============================================================================
==| | No running processes found |
```

IMPORTING PACKAGES

First, we are importing all the required dependencies/packages that's going to download all of the necessary files and packages we needed automatically.

```
# 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
```

Cloning the AlexyAB's Darknet repository for YOLOv4

We will be using the famous AlexeyAB's darknet repository in this model to perform YOLOv4 detections, which is the backbone of the model needed to run smoothly on our both webcam images and videos. Just by giving the following command.

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

```
Cloning into 'darknet'...
remote: Enumerating objects: 15420, done.
remote: Total 15420 (delta 0), reused 0 (delta 0), pack-reused 15420
Receiving objects: 100% (15420/15420), 14.05 MiB | 23.54 MiB/s, done.
Resolving deltas: 100% (10362/10362), done.
```

Working with repository

The below command just going to step into the repository and edit the make file to enable the GPU and opencv as well as this LIBSO allows us to actually get access into the darknet using python.

```
#@title Default title text
# Change makefile to have GPU, OPENCV and LIBSO enabled
%cd darknet
! Just by giving the following command.
!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
/bin/bash: Just: command not found
```

Make function

By running the below "make" command, this is going to build all the binary files and get darknet properly running so that we can access the darknet.py file and run python functions in order to have yoloV4 detections running. Might take little longer than usual.

```
# Make darknet (buildsa darknet so that you can then use the darknet.py file and have its dependencies)
!make

./src/blas_kernels.cu:1130:5: warning: variable ‘step’ set but not used [-Wunused-but-set-variable]
int step = 0;
 ^~~~
nvcc -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=[sm_50,compute_50] -gencode arch=compute_52,code=[sm_5
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are deprecated, and m
nvcc -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=[sm_50,compute_50] -gencode arch=compute_52,code=[sm_5
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are deprecated, and m
nvcc -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=[sm_50,compute_50] -gencode arch=compute_52,code=[sm_5
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are deprecated, and m
nvcc -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=[sm_50,compute_50] -gencode arch=compute_52,code=[sm_5
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are deprecated, and m
 ./src/network_kernels.cu(379): warning: variable "l" was declared but never referenced
```

Wget function

Now run the "wget" command to grab the scaled yolo V4 weights file, model file that's pretrained on over 80 classes or 80 different objects that it can detect which we will utilize but if we wanted to use on custom trained dataset in this spot, we can do that by just uploading our custom trained model file where it can be is commented below.

```
# Get bthe scaled yolov4 weights file that is pre-trained to detect 80 classes (objects) from shared google drive
!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies. txt --keep-session-cookies --no-check-certificate 'https://docs.google.com/uc?export=download&id=1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq' -O- | sed -rn 's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq" -O yolov4-csp.weights && rm -rf /tmp/cookies.txt

--2022-05-30 12:54:15-- https://docs.google.com/uc?export=download&confirm=t&id=1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq
Resolving docs.google.com (docs.google.com)... 74.125.142.100, 74.125.142.113, 74.125.142.102, ...
Connecting to docs.google.com (docs.google.com)|74.125.142.100|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-14-84-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/seh684469vo6259sndkbvppu3ta29fdc/1653915225000/17800843676226924807/*/1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq?e=download [following]
Warning: wildcards not supported in HTTP.
--2022-05-30 12:54:15-- https://doc-14-84-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/seh684469vo6259sndkbvppu3ta29fdc/1653915225000/17800843676226924807/*/1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq?e=download
Resolving doc-14-84-docs.googleusercontent.com (doc-14-84-docs.googleusercontent.com)... 74.125.142.132, 2607:f8b0:400e:c08::84
Connecting to doc-14-84-docs.googleusercontent.com (doc-14-84-docs.googleusercontent.com)|74.125.142.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 211944840 (202M) [application/octet-stream]
Saving to: ‘yolov4-csp.weights’
yolov4-csp.weights 100%[==================>] 202.13M 305MB/s in 0.7s
2022-05-30 12:54:16 (305 MB/s) - ‘yolov4-csp.weights’ saved [211944840/211944840]
```

Darknet for python

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
```

Dataset exploration

As we loaded the dataset, we can have a quick glimpse on the dataset just by the following commands. They return the length of the dataset and their respective object names. Below creates a empty list and then applies read operation on the dataset.

```
# Dataset opeations
objects = []
with open("/content/darknet/data/coco.names", 'r') as f:
objects = f.read().splitlines()
```

```
len(objects)
objects

80
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
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
```

bottle
wine glass
cup
fork
knife
spoon
bowl
banana
apple
sandwich
orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
sofa
pottedplant
bed
diningtable
toilet
tvmonitor
laptop
mouse
remote
keyboard
cell phone
microwave
oven
toaster
sink
refrigerator
book
clock
vase
scissors
teddy bear
hair drier
toothbrush

Furthermore, if you need to know more about the dataset just go through the following link which redirects you to the official site of dataset. https://cocodataset.org/#home

YOLOv4 Example on Test Image

Let's make sure our model has successfully been loaded and that we can make detections properly on a test image which means testing the model.

```
#Load the data
from google.colab import files # Use to load data on Google Colab
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this

```
# Run test on person.jpg image that comes with repository
image = cv2.imread("/content/darknet/data/dog.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)
cv2_imshow(image)
```

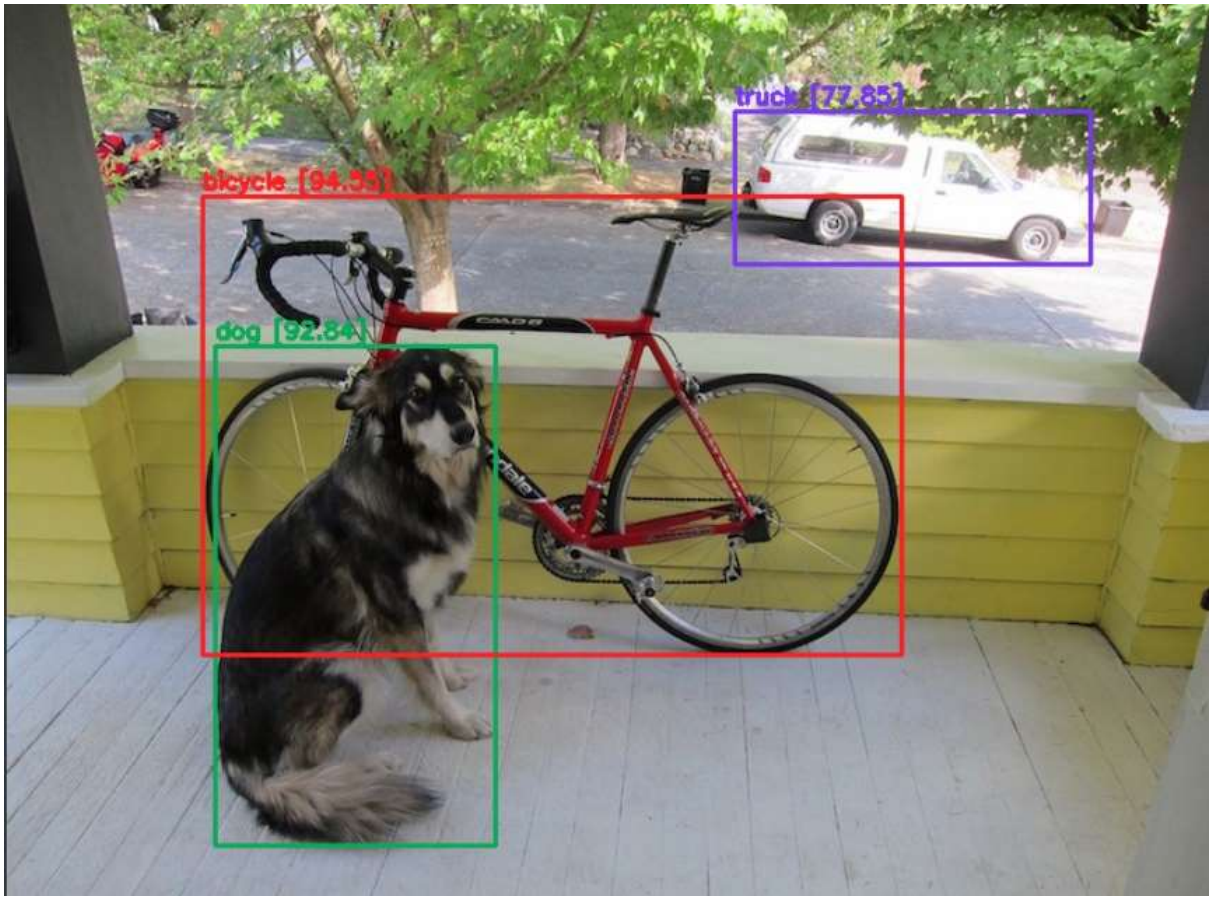

Fig: 7.2 Test Image

Helper functions

Here are a few helper functions defined that will be used to easily convert the images from JavaScript objects into OpenCV and vice versa because to actually utilize the webcam within google Colab workstation we actually have to run JavaScript codes.

```
# 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
# Function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
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 = '{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))
return bbox_bytes
```

YOLO V4 on webcam images

Running YOLOv4 on images taken from 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);
const canvas = document.createElement('canvas');
canvas.width = video.videoWidth;
canvas.height = video.videoHeight;
canvas.getContext('2d').drawImage(video, 0, 0);
stream.getVideoTracks()[0].stop();
div.remove();
return canvas.toDataURL('image/jpeg', quality);
}
''')
display(js)
# Get photo data
data = eval_js('takePhoto({})'.format(quality))
# Get OpenCV format image
img = js_to_image(data)
# Call our darknet helper on webcam image
detections, width_ratio, height_ratio = darknet_helper(img, width, height)
# Loop through detections and draw them on webcam image
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(img, (left, top), (right, bottom), class_colors[label], 2)
cv2.putText(img, "{} [{:.2f}]".format(label, float(confidence)),
(left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
class_colors[label], 2)
# Save image
cv2.imwrite(filename, img)
return filename
```

Webcam images

Webcam Access

Here, the default image name was photo.jpg and modifies when the new image is given to the model through the webcam. If error occurs it will display err as an exception. At last, it saves and replaces the previous photo.jpg.

```
try:
filename = take_photo('photo.jpg')
print('Saved to {}'.format(filename))
# Show the image which was just taken.
display(Image(filename))
except Exception as err:
# Errors will be thrown if the user does not have a webcam or if they do not
# Grant the page permission to access it.
print(str(err))
```

Saved to photo.jpg

Fig:7.3 Live Image

YOLO V4 on webcam videos

Running YOLOv4 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 YOLOv4 model and create an overlay image that contains bounding box of detection(s). We then overlay the bounding box image back onto the next frame of our video stream. YOLOv4 is so fast that it can run the detections in real-time!

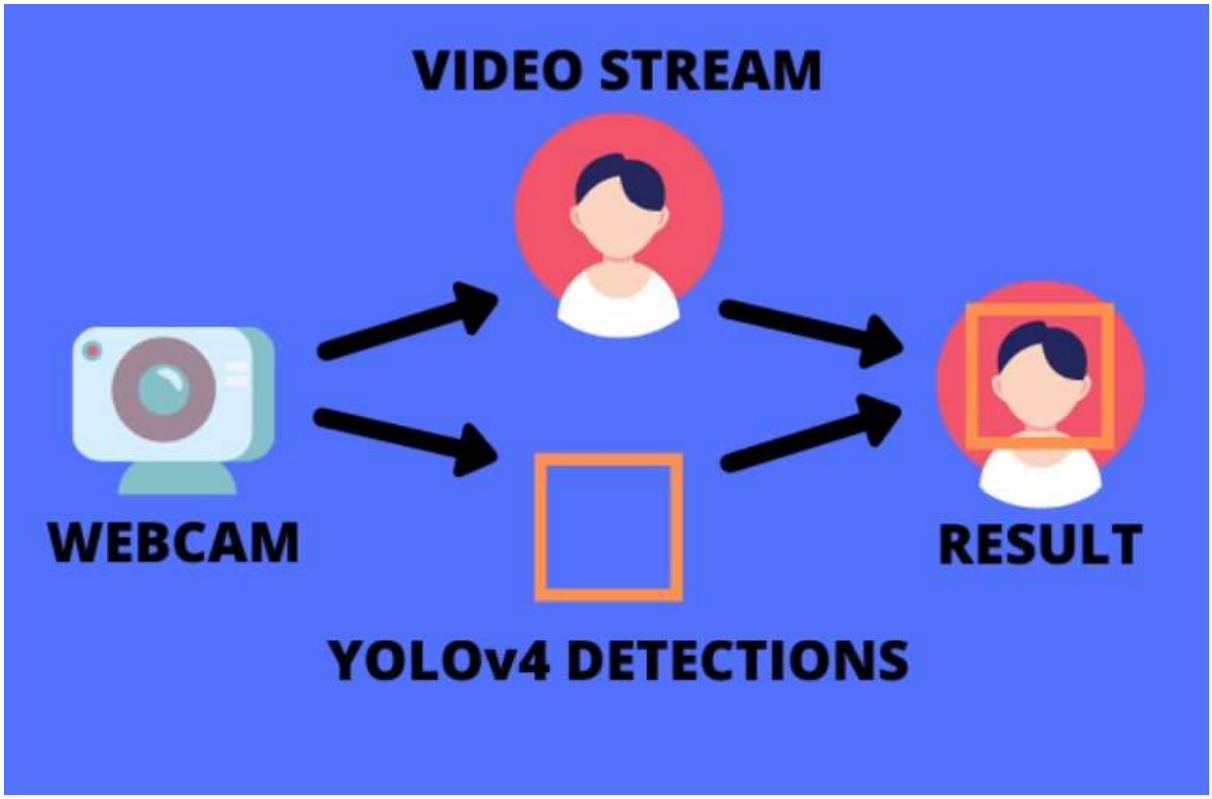

Fig: 7.4 Video Processing Detections

Below are the helper functions to start up the video stream using similar JavaScript as was used for images. The video stream frames are fed as input to YOLOv4.

Helper function

```
# JavaScript to properly create our live video stream using our webcam as input
def video_stream():
js = Javascript('''
var video;
var div = null;
var stream;
var captureCanvas;
var imgElement;
var labelElement;
var pendingResolve = null;
var shutdown = false;
function removeDom() {
stream.getVideoTracks()[0].stop();
video.remove();
div.remove();
video = null;
div = null;
```

```
stream = null;
imgElement = null;
captureCanvas = null;
labelElement = null;
}
function onAnimationFrame() {
if (!shutdown) {
window.requestAnimationFrame(onAnimationFrame);
}
if (pendingResolve) {
var result = "";
if (!shutdown) {
captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
result = captureCanvas.toDataURL('image/jpeg', 0.8)
}
var lp = pendingResolve;
pendingResolve = null;
lp(result);
}
}
async function createDom() {
if (div !== null) {
return stream;
}
div = document.createElement('div');
div.style.border = '2px solid black';
div.style.padding = '3px';
div.style.width = '100%';
div.style.maxWidth = '600px';
document.body.appendChild(div);
const modelOut = document.createElement('div');
modelOut.innerHTML = "<span>Status:</span>";
labelElement = document.createElement('span');
labelElement.innerText = 'No data';
labelElement.style.fontWeight = 'bold';
modelOut.appendChild(labelElement);
div.appendChild(modelOut);
video = document.createElement('video');
video.style.display = 'block';
video.width = div.clientWidth - 6;
video.setAttribute('playsinline', '');
video.onclick = () => { shutdown = true; };
stream = await navigator.mediaDevices.getUserMedia(
{video: { facingMode: "environment"}});
```

```
div.appendChild(video);
imgElement = document.createElement('img');
imgElement.style.position = 'absolute';
imgElement.style.zIndex = 1;
imgElement.onclick = () => { shutdown = true; };
div.appendChild(imgElement);
const instruction = document.createElement('div');
instruction.innerHTML =
'<span style="color: red; font-weight: bold;">' +
'When finished, click here or on the video to stop this demo</span>';
div.appendChild(instruction);
instruction.onclick = () => { shutdown = true; };
video.srcObject = stream;
await video.play();
captureCanvas = document.createElement('canvas');
captureCanvas.width = 640; //video.videoWidth;
captureCanvas.height = 480; //video.videoHeight;
window.requestAnimationFrame(onAnimationFrame);
return stream;
}
async function stream_frame(label, imgData) {
if (shutdown) {
removeDom();
shutdown = false;
return '';
}
var preCreate = Date.now();
stream = await createDom();
var preShow = Date.now();
if (label != "") {
labelElement.innerHTML = label;
}
if (imgData != "") {
var videoRect = video.getClientRects()[0];
imgElement.style.top = videoRect.top + "px";
imgElement.style.left = videoRect.left + "px";
imgElement.style.width = videoRect.width + "px";
imgElement.style.height = videoRect.height + "px";
imgElement.src = imgData;
}
var preCapture = Date.now();
var result = await new Promise(function(resolve, reject) {
pendingResolve = resolve;
});
```

```
shutdown = false;
return {'create': preShow - preCreate,
'show': preCapture - preShow,
'capture': Date.now() - preCapture,
'img': result};
}
'')
display(js)
def video_frame(label, bbox):
data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
return data
```

Running on webcam video

Below one opens the video stream and then do the detections on each frame. So, while true it's going to keep on looping through those detections and saving into the blank array bbox[] this is the transparent overlay and then saving into the rectangle bounding box and the text of the class onto the blank overlay and then it's just going to write it bbox update the overlay eachframe that then gets passed into the video frame.

```
# Start streaming video from webcam
video_stream()
# Label for video
label_html = 'Capturing...'
# Initialze bounding box to empty
bbox = ''
count = 0
while True:
js_reply = video_frame(label_html, bbox)
if not js_reply:
break
# Convert JS response to OpenCV Image
frame = js_to_image(js_reply["img"])
# Create transparent overlay for bounding box
bbox_array = np.zeros([480,640,4], dtype=np.uint8)
# Call our darknet helper on video frame
detections, width_ratio, height_ratio = darknet_helper(frame, width, height)
# Loop through detections and draw them on transparent overlay image
```

```
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_rati
bbox_array = cv2.rectangle(bbox_array, (left, top), (right, bottom), class_colors[label], 2)
bbox_array = cv2.putText(bbox_array, "{} [{:.2f}]".format(label, float(confidence)),
(left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
class_colors[label], 2)
bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
# Convert overlay of bbox into bytes
bbox_bytes = bbox_to_bytes(bbox_array)
# Update bbox so next frame gets new overlay
bbox = bbox_bytes
```

CHAPTER 08

TESTING

8.1 TESTING INTRODUCTION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

8.2 TESTING PROCESS

Testing is a disciplined process of finding and debugging defects to produce defect-free software. During testing, a test plan is prepared that specifies the name of the module to be tested, reference modules, date and time, location, name of the tester, testing tools, etc. Software engineers design test cases while writing the source codes, Testers may also involve in test case design. Test cases include the input, output, and the conditions. Software tester runs the program using test cases and observes results.

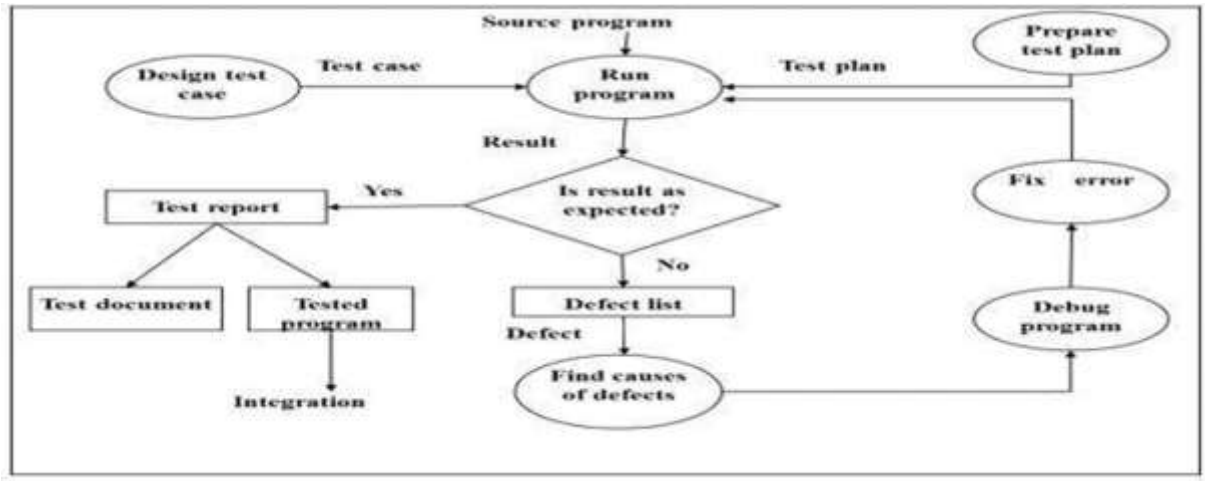

Fig: 8.1 Testing Process

8.3 LEVELS OF TESTING

Testing is a defect detection technique that is performed at various levels. Testing begins once a module is fully constructed. Although software engineers test source codes after it is written, but it is not an appealing way that can satisfy customer's needs and expectations.

Software is developed through a series of activities, i.e., customer needs, specification, design, and coding. Each of these activities has different aims. Therefore, testing is performed at various levels of development phases to achieve their purpose.

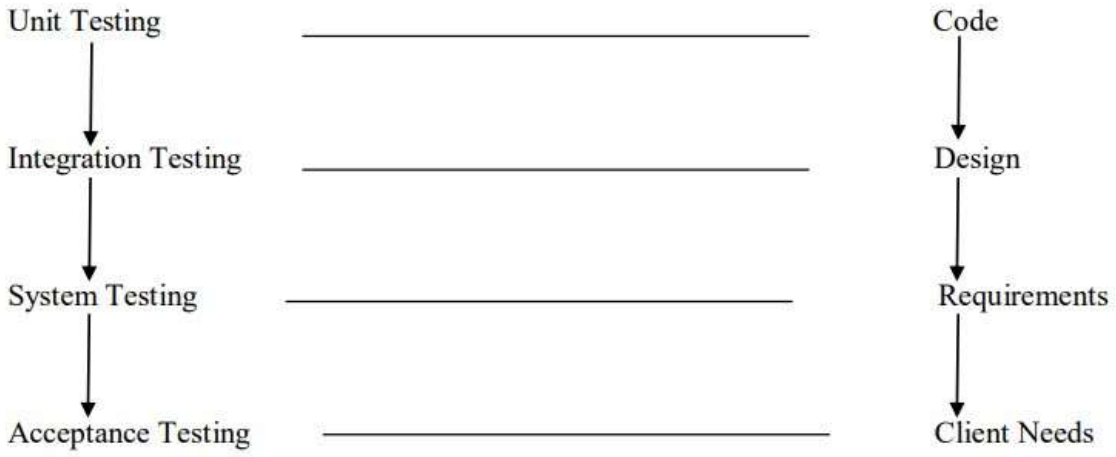

Fig 8.2 Testing Levels

8.3.1 UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

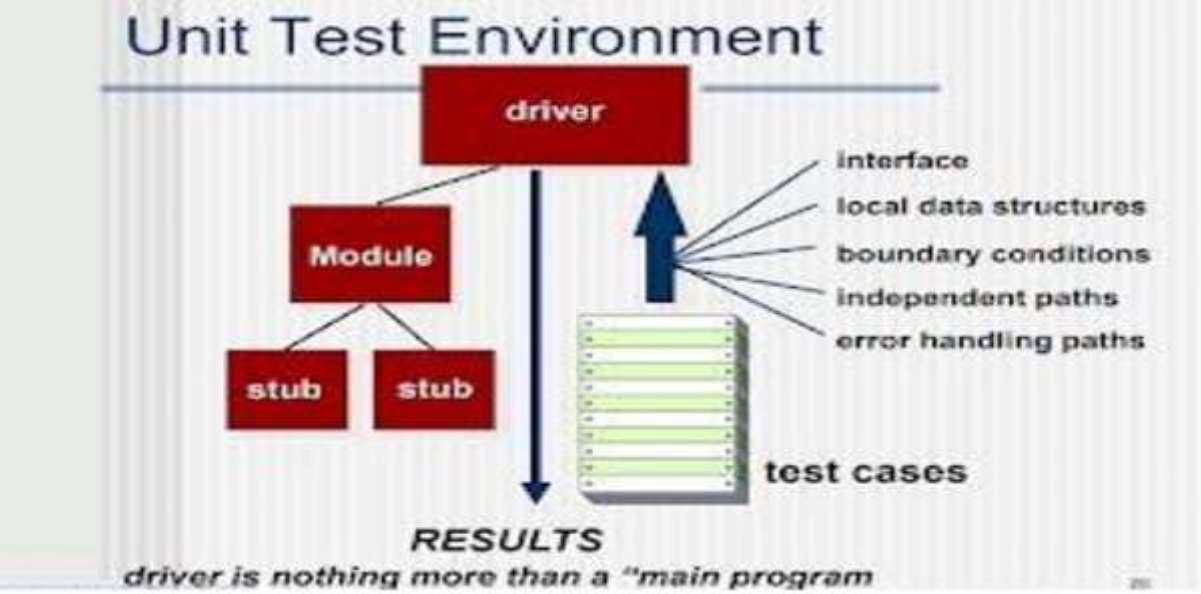

Fig: 8.3 Unit Testing

8.3.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

The following are the types of Integration Testing:

1. Top-Down Integration:

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner. In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-Up Integration:

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from

the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

The bottom-up integration strategy may be implemented with the following steps:

• The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.

• A driver (i.e.) the control program for testing is written to coordinate test case input and output.

• The cluster is tested.

• Drivers are removed and clusters are combined moving upward in the program structure The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

8.3.3 SYSTEM TESTING

Software once validated must be combined with other system elements (e.g., Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

8.3.4 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

- Acceptance testing is performed at two levels, i.e.,
- Alpha testing
- Beta testing

8.3.5 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software

under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.3.6 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.4 TEST CASE

We now know, test cases are integral part of testing. So, we need to know more about test cases and how these test cases are designed. The most desired or obvious expectation from a test case is that it should be able to find most errors with the least amount of time and effort.

In this project test refers to the real-world objects so then we can place any kind objects in front of systems camera in the name of testing.

Testing results during the testing phase.

Fig: 8.4 Testing

CHAPTER 09
CONLUSION AND FUTURESCOPE

9.1 CONCLUSION

Deep learning has gained tremendous success in image classification tasks. Our architecture, deep learning which is based on YOLO algorithm works as feature extractor eliminating the need to apply processing technique. The predefined dataset has successfully helped to conduct experiments and the accuracies are astonishing due to dataset and model we implemented in this project.

The project developed on google Colab will be useful to any kind of person out there with any device to detect the objects.

9.2 Future Scope

The future perspective of the approach is to detect vast number of objects which can be done by COCO (Common Objects in Context) dataset under Microsoft proprietorship and to infuse the presented technique into a mobile application so, that it proves to be a great utility for a people with zero knowledge on technologies.

The application areas that can be beneficial through the proposed approach include in achieving level 05 self-driving cars, agricultural purpose (detecting crop diseases and suggesting proper medical advisory), autonomous traffic monitoring.

CHAPTER 10

REFERENCES

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi "You Only Look Once: Unified, Real-Time Object Detection" 2016 YOLO V1

[2] Chengji Liu, Yufan Tao, Jiawei Liang, Kai Li1, Yihang Chen "Object Detection on YOLO Network" 2018 YOLO V3

[3] Wenbo Lan, Jianwu Dang, Yang-ping Wang, Song Wang "Pedestrian Detection Based on YOLO Network Model" 2018 YOLO V3

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haơner," Gradient-based learning applied to document recognition," Proceedings of the IEEE, Nov 1998, 86(11), pp. 2278-2324.

[5] Rumin Zhang, Yifeng Yang "An Algorithm for Obstacle Detection based on YOLO and Light Filed Camera" 2018 YOLO V3

[6] Zhimin Mo1, Liding Chen1, Wen-jing You "Identification and Detection of Automotive Door Panel Solder Joints based on YOLO" 2019 YOLO V4

[7] Chen Yan-yan, Chen Ning, Zhou Yu-yang, Wu Ke-han, and Zhang Wei-wei, ‖ Pedestrian Detection and Tracking for Counting Applications in Metro Station‖, Discrete Dynamics in Nature and Society, Volume 2014, Article ID 712041, 11 pages, http://dx.doi.org/10.1155/2014/712041,2014

[8] Diego Fustes, Diego Cantorna, Carlos Dafonte, Bernardino Arcay, Alfonso Iglesias and Minia Manteiga, ‖ A cloud-integrated web platform for marine monitoring using GIS and remote sensing. Application to oil spill detection through SAR images‖, Journal of Future Generation Computer Systems, Elsevier, 2013.

[9] L. Riazuelo, Javier Civera and J.M.M. Montiel, ‖ C2TAM: A Cloud framework for cooperative tracking and mapping‖, Journal of Robotics and Autonomous Systems, 401–413, Elsevier,2013.

[10] Jianxin Wu, Nini Liu, Christopher Geyer and James M. Rehg, ‖ C4: A Real-time Object Detection Framework‖, IEEE Transaction on Image Processing, 2013.

[11] Jamal Raiyan, ―Detection of Objects in Motion – A Survey of Video Surveillance‖, Journal Advances in Internet of Things, 2013, 3, 73-78.

[12] Ziyan Wu and Richard J. Radke, ―Improving Counter Flow Detection in Dense Crowds with Scene Features‖, Pattern Recognition Letters 152–160, Elsevier, 2013.

[13] Rajini Nema & Dr. A. K. Sazena, ‖ Modified Approach for Object Detection in Video Sequences. ‖, American International Journal of Research in Science, Technology, Engineering and Mathematics, ISSN:2328-3491,122-126,2013.

[14] Lawrence O'Gorman, Yafeng Yin and Tin Kam Ho, ―Motion feature filtering for event detection in crowded scenes‖ Pattern Recognition Letters, Elsevier, pp:80–87,2013.

[15] B. Karasulu and S. Korukoglu, ‖ Moving Object Detection and Tracking in Videos‖, Performance Evolution Software, Springer Briefs in Computer Science, DOI: 10.1007/978 -1-4614 -6534- 8_2,2013.

[16] Chia-Hung Yeh, Chih-Yang Lin, Kahlil Muchtar and Li-Wei Kang, ‖ Real-time background modelling based on a multi-level texture Description‖, Journal of Information Sciences, Elsevier, PP: 106–127,2013.

[17] Gustavo Moreira, Bruno Feijo, H´elio Lopes and Raul Queiroz Feitosa, ‖ Real-time Object Tracking in High-Definition Video Using Frame Segmentation and Background‖, 26th SIBGRAPI – IEEE Conference on Graphics, Patterns and Images (SIBGRAPI), PP:75- 82,2013.

[18] Manisha Chaple & Prof. S. S. Paygu, ‖ Vehicle Detection & Tracking from video frame sequence. ‖, International Journal of Scientific and Engineering Research, Volume 4, Issue 3, March-2013.

[19] Jeba Veera Singh and Nancy Emymal., ‖ A Critical Survey of Moving Object Detection Techniques and Related Proposed research‖, International Journal of Computer Applications in Engineering Sciences, Volume III, Issue I, ISSN:2231- 4946, March 2013.

[20] Bineng Zhong, Xiaotong Yuan, Rongrong Ji, Yan Yan, Zhen Cui, Xiaopeng Hong, Yan Chen, Tian Wang, Duansheng Chen and Jiaxin Yu, ‖ Structured Partial Least Squares for Simultaneous Object Tracking‖, Journal of Nerocomputing, Elsevier, 2014.