A PROJECT REPORT

ON

"OBJECT DETECTION USING YOLO"

Submitted to

BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY) COLLEGE OF ENGINEERING, PUNE, INDIA

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER ENGINEERING

BY

Exam No: 2214390045	SUYASH GAOPANDE	PRN No: 1814110046
Exam No: 2214390029	BHARAT HIRANI	PRN No: 1814110024
Exam No: 2214390122	KUMAR GAURAV	PRN No:1914110500

UNDER THE GUIDANCE OF
PROF. MAYURI H MOLAWADE

DEPARTMENT OF COMPUTER ENGINEERING

BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY) COLLEGE OF ENGINEERING

PUNE, INDIA - 411043

2021-2022

A PROJECT REPORT

ON

“OBJECT DETECTION USING YOLO”

Submitted to

BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY) COLLEGE OF ENGINEERING, PUNE, INDIA - 411043

In Partial Fulfillment of the Requirement for the Award of

BACHELOR’S DEGREE IN
COMPUTER ENGINEERING

BY

Exam No: 2214390045	SUYASH GAOPANDE	PRN No: 1814110046
Exam No: 2214390029	BHARAT HIRANI	PRN No: 1814110024
Exam No: 2214390122	KUMAR GAURAV	PRN No: 1914110500

UNDER THE GUIDANCE OF
PROF. MAYURI H MOLAWADE

DEPARTMENT OF COMPUTER ENGINEERING

BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY) COLLEGE OF ENGINEERING

PUNE, INDIA - 411043

2021-2022

BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY) COLLEGE OF ENGINEERING,

CERTIFICATE

This is certify that the project entitled

“OBJECT DETECTION USING YOLO “

submitted by

Exam No:2214390045	SUYASH GAOPANDE	PRN No: 1814110046
Exam No:2214390029	BHARAT HIRANI	PRN No: 1814110024
Exam No:2214390122	KUMAR GAURAV	PRN No: 1914110500

It is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Engineering at Bharati Vidyapeeth (Deemed To Be University) College of Engineering, Pune, India. This work is done during academic year 2021-2022.

Date: / /

Signature	Signature	Signature
Prof. Dr. S.B.Vanjale H.O.D, Computer Department	Prof. Mayuri H Molawade Project Guide	EXTERNAL EXAMINER

Acknowledgements

We would like to express deepest appreciation towards Dr. Vidula Sohoni, Principal, Bharati Vidyapeeth (Deemed To Be University) College of Engineering, Pune, India and Prof. Dr. S.B.Vanjale, Head of Department of Computer Engineering, who invaluable supported us in completing this project.

We are profoundly grateful to PROF. MAYURI H MOLAWADE, Project guide for his expert guidance and continuous encouragement throughout to see that this project rights, its target since its commencement to its completion.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

KUMAR GAURAV

BHARAT HIRANI

SUYASH GAOPANDE

ABSTRACT

When we look at images or videos, we can easily locate and identify the objects of our interest within moments. Passing on this intelligence to computers is nothing but object detection - locating the object and identifying it. Object Detection has found its application in a wide variety of domains such as video surveillance, image retrieval systems, autonomous driving vehicles and many more. Various algorithms can be used for object detection but we will be focusing on the YoloV3 algorithm. YOLO stands for "You Only Look Once". The YOLO model is very accurate and allows us to detect the objects present in the frame. YOLO follows a completely different approach. Instead of selecting some regions, it applies a neural network to the entire image to predict bounding boxes and their probabilities. YOLO is a single deep convolutional neural network that splits the input image into a set of grid cells, so unlike image classification or face detection, each grid cell in YOLO algorithm will have an associated vector in the output that tells us if an object exists in that grid cell, the class of that object, the predicted bounding box for that object. The model here is progressive so it learns more over time, increasing its prediction accuracy over time. The way the model works is that it makes many predictions in one frame and decides to use the most accurate prediction, thus discarding the other. The predictions are made randomly, so if the model feels like there is an object in the frame which is of a very small pixel it will take that also into consideration. To make it more precise and clearer, the model simply creates bounding boxes around everything in the frame, it would make predictions for each box and pick the one with the most confidence score. All this is done in a small-time frame, thus showing why this specific model is the best to use in a real time situation.

Keywords:
Object Detection, YOLO, OpenCV, Python, Mobilenet-SSD, CN

Contents

List of Figures

`

Chapter 1

1.1 Introduction

Humans can easily detect and identify objects present in an image. The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. With the availability of large amounts of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy.
An image classification or image recognition model simply detect the probability of an object in an image. In contrast to this, object localization refers to identifying the location of an object in the image. An object localization algorithm will output the coordinates of the location of an object with respect to the image. In computer vision, the most popular way to localize an object in an image is to represent its location with the help of bounding boxes. With this, we come to the end of the introduction to object detection. We now have a better understanding of how we can localize objects while classifying them in an image. We also learned to combine the concept of classification and localization with the convolutional implementation of the sliding window to build an object detection system.

1.1.1 Problem Statement

The main aim of this project is to build a system that detects objects from the image or a stream of images given to the system in the form of previously recorded video or the real time input from the camera. Bounding boxes will be drawn around the objects that are being detected by the system. The system will also classify the object to the classes the object belongs. Python Programming and a Machine Learning Technique named YOLO (You Only Look Once) algorithm using Convolutional Neural Network is used for the Object Detection.

1.1.2 Motivation

Image recognition/image processing is in the forefront of Artificial Intelligence today. It is however far from perfection. Seemingly simple scenarios, such as object detection, face recognition, removing motion blur, etc. and more complex scenarios such as compression arte f acts scratch detection, sensor noise, and spilling detection are applications of image recognition/image processing.

1.1.3 Objective, Goal and scope of the research work

The Objective is to detect of objects using You Only Look Once (YOLO) approach. This method has several advantages as compared to other object detection algorithms. In other algorithms like Convolutional Neural Network, Fast Convolutional Neural Network the algorithm will not look at the image completely but in YOLO the algorithm looks the image completely by predicting the bounding boxes using convolutional network and the class probabilities for these boxes and detects the image faster as compared to other algorithms.

In this project, we are using a YOLO-COCO Detector for our task. We have created 2 different folders one for keeping the detection and configuration and other consisting of the pre-trained model. YOLO-COCO file consists of coco.names, yolov3.cfg, yolov3.weights. We are using the pre-trained models for object detection. Other than the two folders we have kept in the main folder in the main folder we have the main python file which we are going to run in order to check if the program is running successfully or not. The main python file is interconnected internally with the other two files which has configuration and the one which use the trained model to detect the people and objects and label them. When we call the main file it access the webcam of the laptop to take the input of the video file.

Object detection is a well-known computer technology connected with computer vision and image processing that focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital images and videos. There are various applications of object detection that have been well researched including face detection, character recognition, and vehicle calculator. Object detection can be used for various purposes including retrieval and surveillance. In this study, various basic concepts used in object detection while making use of OpenCV library of python 3, improving the efficiency and accuracy of object detection arepresented.

Computer vision has advanced considerably but is still challenged in matching the precision of human perception. This article belongs to computer vision. Here we will learn from scratch. It can be challenging for beginners to distinguish between different related computer vision tasks.

Humans can easily detect and identify objects present in an image. The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detecting obstacles with little conscious thought.

Chapter 2
Literature Survey

2 Literature Review

2.1.1 Review of Existing Models, Approaches, Problems

Object detection is a well-known computer technology connected with computer vision and image processing that focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital images and videos. Thereare various applications of object detection that have been well researched including face detection, character recognition, and vehicle calculator. Object detection can be used for various purposes including retrieval and surveillance. In this study, various basic concepts used in object detection while making use of OpenCV library of python 3., improving the efficiency and accuracy of object detection are presented.

2.1.2 Significance of Models, Approaches, Problems

YOLO algorithm performs real-time object detection using convolutional neural network(CNN). As the name suggests, the algorithm solely needs a single forward propagation through a neural network to detect objects. This means that prediction in the entire image is performed in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously. The major features of YOLO are speed, high accuracy and its learning capability.

2.1.3 State of Art: Review

Our proposed model depends on the MobileNet SSD architecture. One reason why we chose this architecture is on the because that as shown in the paper , it gives good object detection accuracy while being quicker than different architectures, for example, R-CNN.

2.1.4 Object Tracking

An object tracker aims to generate trajectory of an object over time by locating its position in every frames of video [1]. Generally, object tracking has two main tasks. One is to detect the object and another one is to establish the correspondence between the object in every frames of video. These tasks can be performed either separately or jointly. The first case, the object in region has been detected by detector mechanism and later connection between objects in sequence of frame has been done by tracker. In the later case, the object region and correspondence is jointly estimated. It can be done by iteratively updatingobject location and region information obtained from previous frames.♫

In the previous study most of them have concentrated towards Object detection (Ben Ayed et al., 2015; Najva and Bijoy, 2016; Ramya and Rajeswari, 2016; Risha and Kumar, 2016; Shen et al., 2013; Soundrapandiyan and Mouli, 2015; Viswanath et al., 2015) ,Object tracking (Bagherpour et al., 2012; Coşkun and Ünal, 2016; Foytik et al., 2011; Lee et al., 2012; Poschmann et al., 2014; Weng et al., 2013; Yilmaz et al., 2006; Zhang et al., 2016) and Object recognition (Chakravarthy et al., 2015; Elhariri et al., 2015; Gang et al., 2010; Ha and Ko, 2015; Nair et al., 2011) for tracking the object using video sequences. These are discussed as follows. The basic flow diagram of an object tracking shown in figure 1.

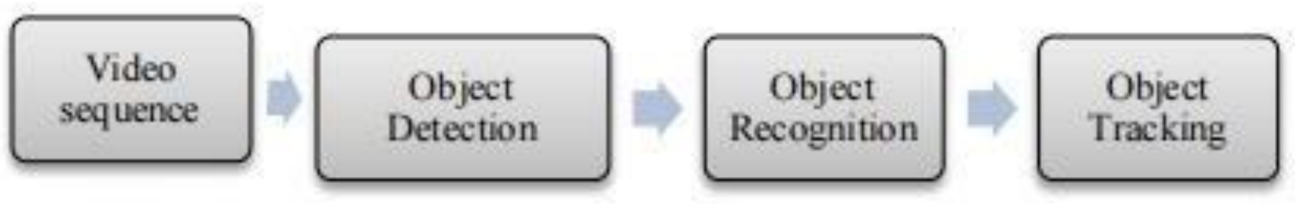


Figure 1 The Basic flow diagram of Object tracking

Studies related to object detection
The detection of an object in video sequence plays a significant role in many applications. Specifically as video surveillance applications (Amandeep and Goyal, 2015). The different types of object detection are shown in figure 2.

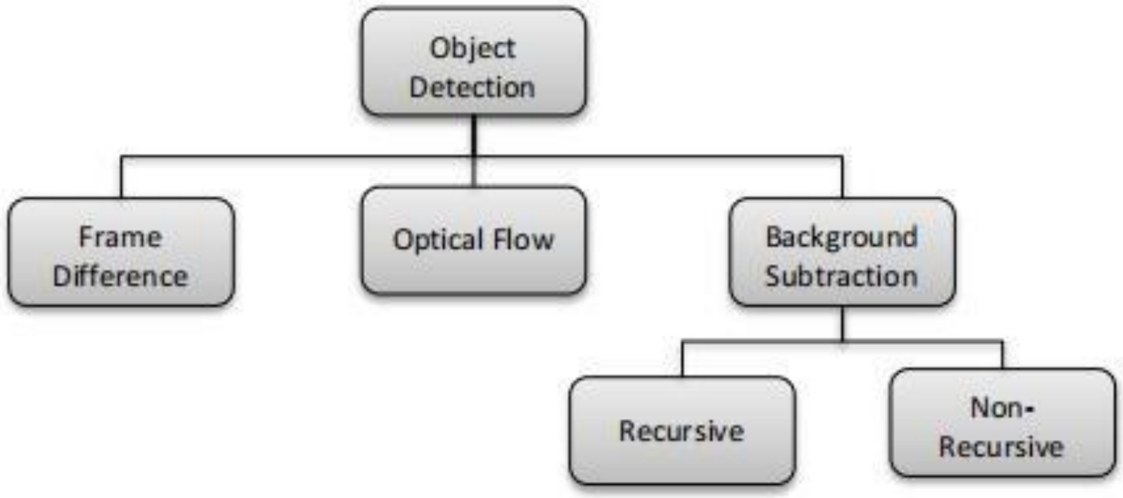


Figure 2 Types of object detection method

Chapter 3

Software Requirements Specification

3.1 Software need to be installed on system:

- **Python 3** or higher.
- **OpenCV Library** in Python for Machine Learning and image processing purposes.
- **NumPy Library** in Python in order to calculate the mathematical operations during running processes.
- **iMutils Library**: A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.
- **An Python IDE** like Visual Studio Code, Atom or PyCharm for running the project.

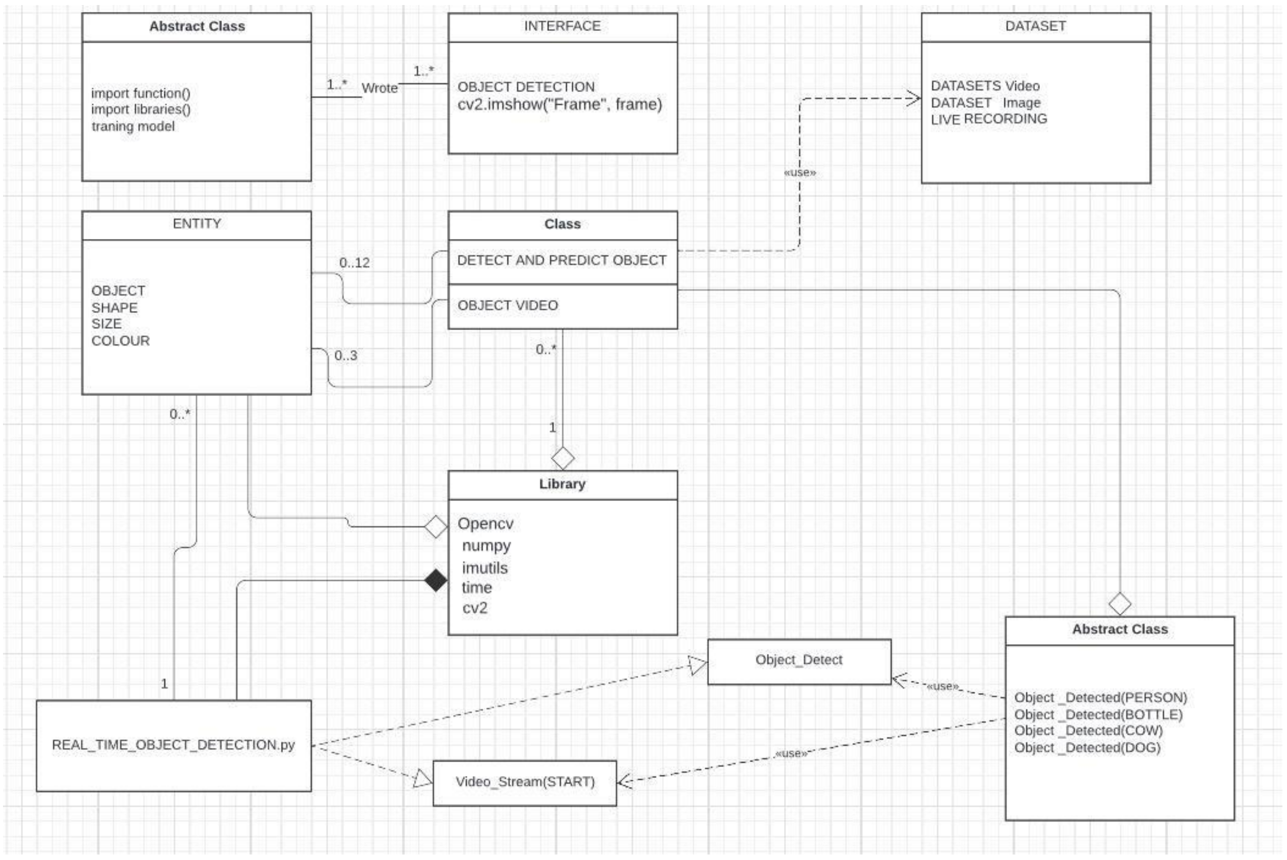


Figure3.1 : UML Diagram

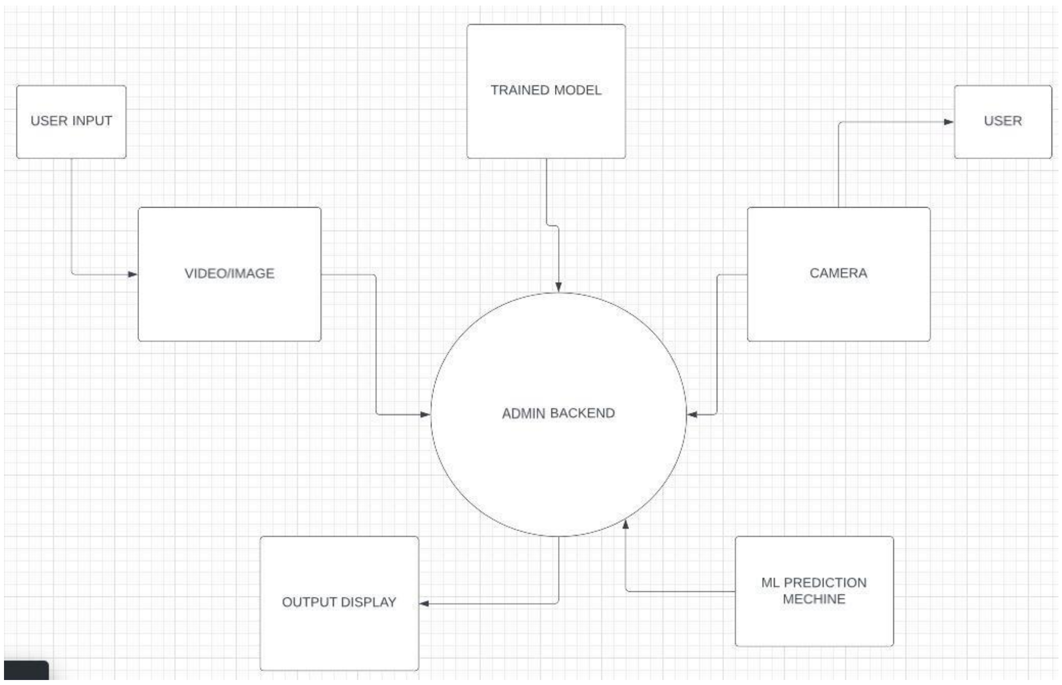


Figure3.2 : Component Diagram

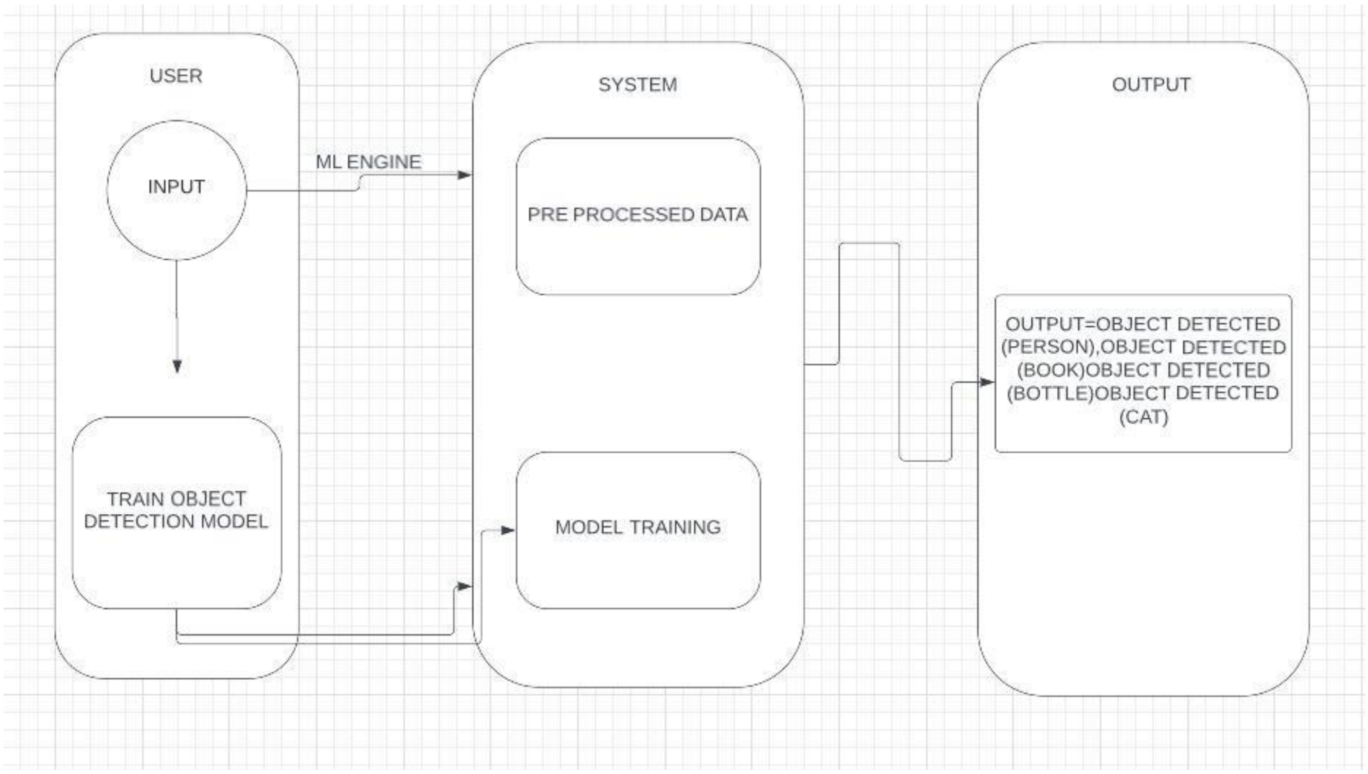

USER
SYSTEM
OUTPUT
INPUT
ML ENGINE
PRE PROCESSED DATA
OUTPUT=OBJECT DETECTED (PERSON),OBJECT DETECTED (BOOK)OBJECT DETECTED (BOTTLE)OBJECT DETECTED (CAT)
TRAIN OBJECT DETECTION MODEL
MODEL TRAINING

Figure3.3 : Activity Diagram

Chapter 4

Requirement Analysis

4.1 Requirement Specification

- **OpenCV 4.6** needs to compile at least 1GB RAM with 2GB swap memory, because less than this configuration OpenCV will not be able to compile on that system to generate object code.
- **NumPy** requires Python 3 to be installed on any PC.
- **iMutils** requires Python 3 to be installed on any PC.

4.2 Functional Specification

1. R1. System must have an unbiased dataset.
2. R2. Dataset must have over 1500+ images labelled in order to detect the Object.
3. R3. The dataset must not re-use the same images in training and testing phases.

1. R4. The system must be correctly able to load the object detection model.
2. R5. The system must be able to detect the object in images or video streams.

4.3 Non-Functional Specification

Product Operation:

1. R1. The Object should be localized by detecting the object datapoints and the background must be ignored.
2. R2. The system will be implemented in Python script with an accuracy of the model of over 90%.
3. R3. The object must not been move out of camera’s sight in order to get correct results.
4. R4. The background must not be too bright or too dark while detecting the object.

Chapter 5

System Design

5.1 Proposed Solution

A feasible approach has been proposed that consists of first detecting the type of object by learning from the COCO Dataset.

Convolutional neural networks are used to develop a model which consists of multiple layers to classify the given objects into any of the defined classes. These objects are detected by making use of higher resolution feature maps and are possible because of the recent advancement in deep learning with image processing. Mobilenet SSD is an object detection model that computes the output bounding box and class of an object from an input image. This Single Shot Detector (SSD) object detection model uses Mobilenet as the backbone and can achieve fast object detection optimized for mobile devices.

5.2 Design Approach

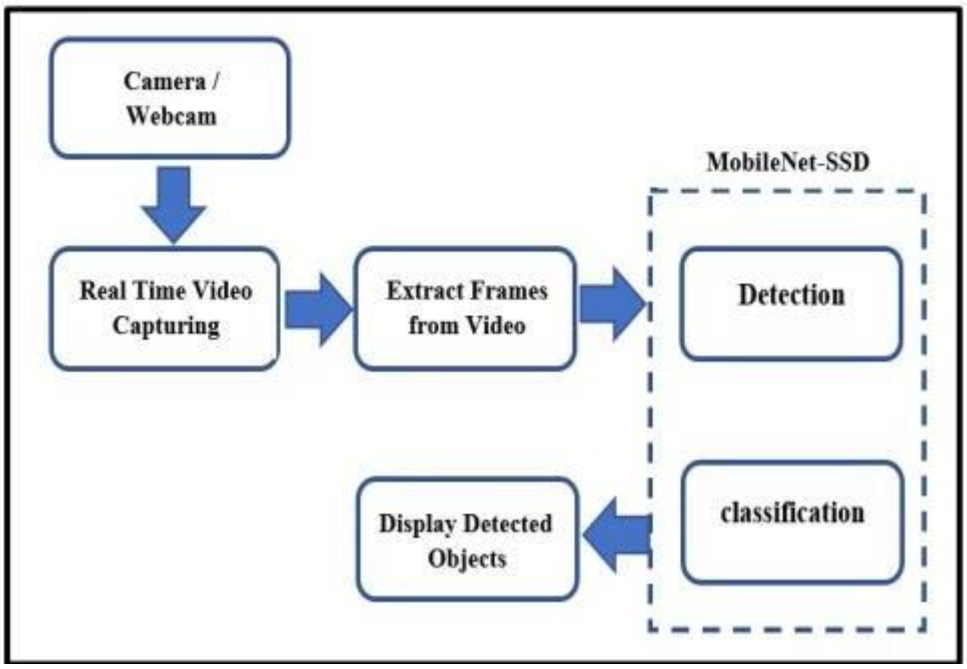


FIG 5.1

Data Visualization

In the first step, let us visualize the total number of images in our dataset that we are using, Approximately there are 200K labelled images are present in our dataset based on COCO model.

Data Augmentation

In the next step, we augment our dataset to include more number of images for our training. In thisstep of data augmentation, we rotate and flip each of the images in our dataset.

Splitting the data

In this step, we split our data into the training set which will contain the images on which the YOLO model will be trained using COCO dataset and the test set with the images on which our model will be tested.

Using the Trained dataset to Label the Output

This step is the main step where we fit our images in the training set and the test set to ourSequential model using the COCO dataset and compare the input image with the labelled dataset in the system..

Labelling the Information

After building the model, we label the probabilities for our results such as different objects like humans, Animals and over 80 other objects.

5.3 Design Tools used:

Introduction

OpenCV: OpenCV stands for Open-Source Computer Vision. It's an Open Source BSD licensed librarythat includes hundreds of advanced Computer Vision algorithms that are optimized to use hardware acceleration. OpenCV is commonly used for machine learning, image processing, image manipulation, and much more. OpenCV has a modular structure. There are shared and static libraries and a CV Namespace.

OpenCV is used in our application to easily load bitmap files that contain landscaping pictures and perform a blend operation between two pictures so that one picture can be seen in the background of another picture. This image manipulation is easily performed in a few lines of code using OpenCV versus other methods.

Python: As a dynamically typed language, Python is flexible. This means there are no hard rules on how to build features, and you'll have more flexibility solving problems using different methods. Furthermore debugging is easier in Python and hence our choice of language.

YOLOv3: YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect an object.

COCO-Dataset: COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

MobileNet-SSD: Our proposed model depends on the MobileNet SSD architecture. One reason why we chose this architecture is on the because that as shown in the paper it gives good object detection accuracy while being quicker than different architectures, for example, YOLO. Especially, this is valid when attempting to detect object in real time in low computing devices as in our system. MobileNet-SSD permits to lessen the detection time by addressing the model utilizing 8-bit integers rather than 32-bit floats.

The input of the model was set to an image with 300 by 300 pixels and the result of the model addressed the position of the bounding box as well as the detection confidences (from 0 to 1) for each identified object. A detection confidence threshold of 0.5 was utilized to decide if the detected object was valid.

5.4 Detailed System Design

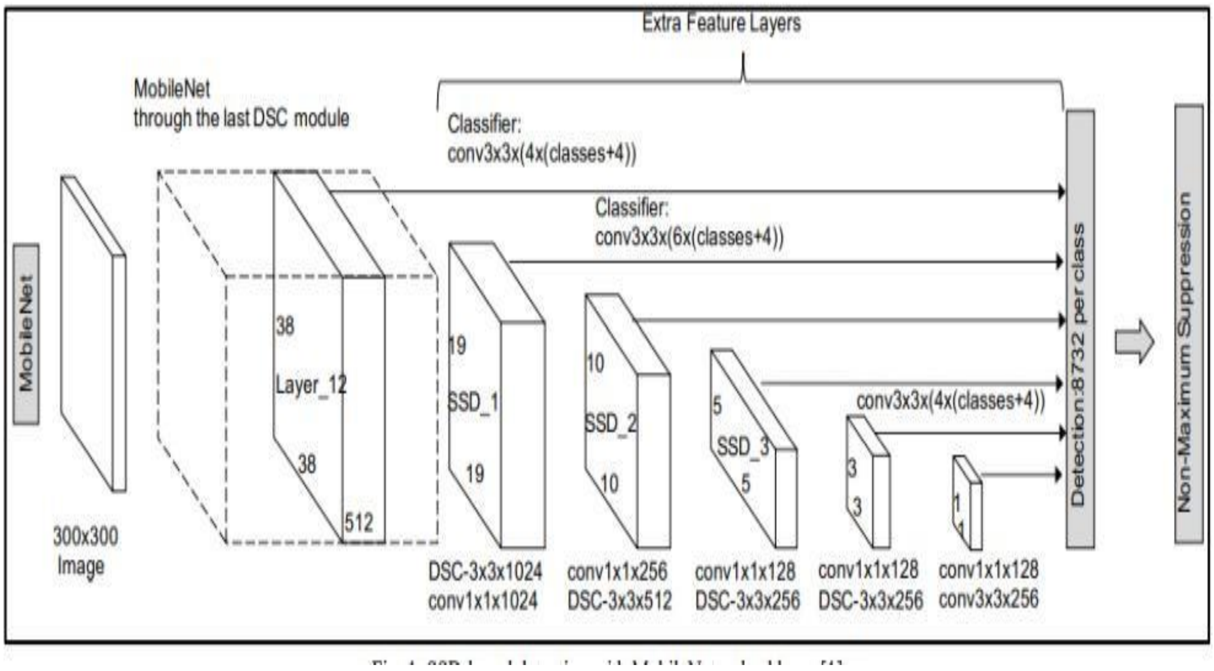


FIG 5.2

Chapter 6

Test Case And Test Results

The data foundation

The model has been trained on the COCO dataset (Common Objects in Context). Just as it sounds like this dataset contains a lot of images with objects we see quite often in everyday life. Specifically it consists of 300k images of 90 different objects such as

- Fruit
- Vehicles
- People
- Etc.

The model

The model we are using is the "Single Shot Multibox Detector (SSD) with MobileNet" located here and takes a little while to download. It's stored using Google's Protocol Buffers format. I know what you're thinking: Why oh why invent yet another structured storage format? In Google's defense this one is pretty cool. Basically it's a language-neutral, platform-neutral, extensible mechanism for serializing structured data — think XML, but smaller, faster, and simpler. You define how you want your data to be structured once, then you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages.

Convolutional neural networks

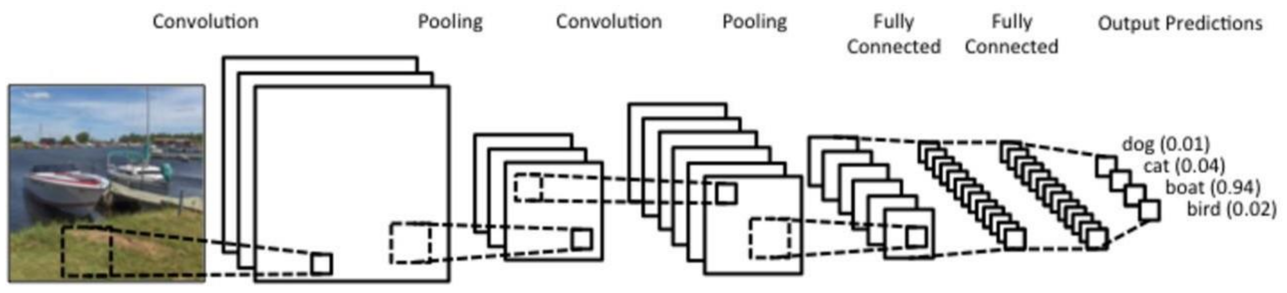


FIG 6.1
A Generic Convolutional Neural Network Architecture with Kernel filters and Pooling layers

Before we have a look at the results let's give a quick introduction to what convolutional neural networks really are and why they are more successful at image analysis than normal multi-layered perceptrons. The whole idea behind using convolutional neural networks is that we need the network to be translation and rotation invariant. This just means that we need to be able to recognize an object no matter where in the image it resides. One way to achieve this is to swipe a patch reacting to certain patterns over the image. Think of it as a filter that lights up when it detects something. Of course we don't know what we are looking for and therefore these filters are learned during training. In general in deep learning we learn lower level features in the initial layers while the later layers captures more elaborate features. This is pretty cool as we can save initially trained early layers and reuse them in other models.

Results

The video below was shot from my cell phone while the modeling team was working. As you can see the model does indeed identify some objects quite successfully. But it also fails to detect many of them. There are many reasons for this. One of them is that this model is optimized for speed and not for performance.

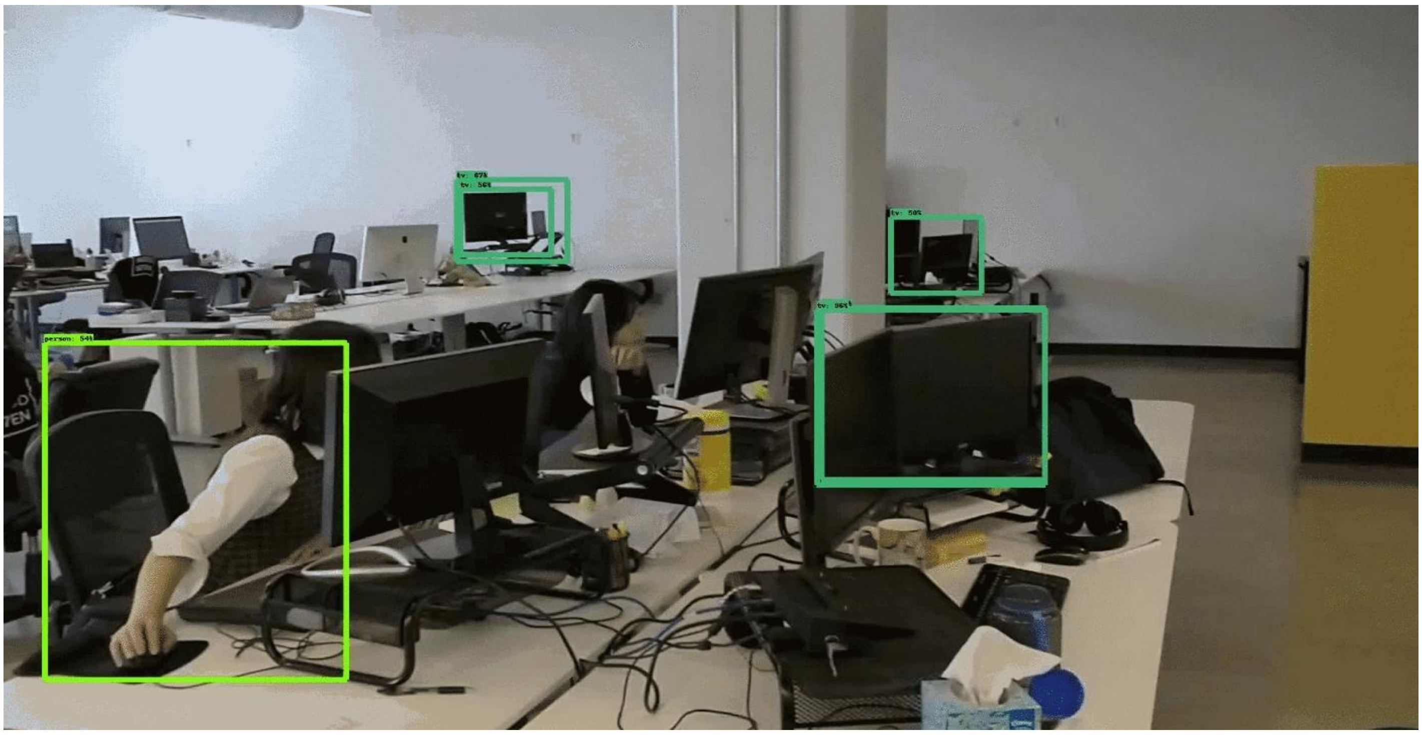


FIG6.2

In this short post It showed you how it's possible to utilize a previously fitted convolutional neural network and classify objects in a retrospective video. However this can be extended into a live object detection from a webcam or a surveillance camera

SAMPLE CASES AND RESULTS

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Object	Correct	99.99%	100%

	Detection	Prediction		
T02	Object Detection	Incorrect Prediction	89.97%	100%
T03	Login	Login Authenticatio n	100%	100%
T04	Image status	Image display status	100%	100%
T05	Live video	Live video feed status	100%	100%
T06	Movemen t	Live video	79.9%	100%
T07	Simmilar	Live video/Image	59.4%	100%
T08	Colour	Live video/Image	77.5%	100%

Chapter 7

Project Planning

7.1 Proposed Project Plan

1. This system is capable of tracking various objects.
2. After training with the YOLO-COCO dataset, it can detect 80 types of object including humanand animals such as cat and dog, objects like bottle, books, tables, chairs, etc.
3. It also can access the webcam and predict the result.

7.2 Detailed Project Plan

Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections. We use a totally different approach. We apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Features of this System

- The system is easy to implement in any existing organizational system.
- No need to install any hardware as the system can relate to your existing surveillancesystem only.
- The system can be used easily with any camera or hardware like surveillance cameras.
- You can check the analytics based on the system generated reports.
- Easy to access and control the movements from any device through

applications.

Chapter 8
Implementation

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

Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and probability of the object being present in the cell.

This process greatly lowers the computation as both detection and recognition are handled by cells from the image, but—

It brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions.

YOLO makes use of Non Maximal Suppression to deal with this issue.

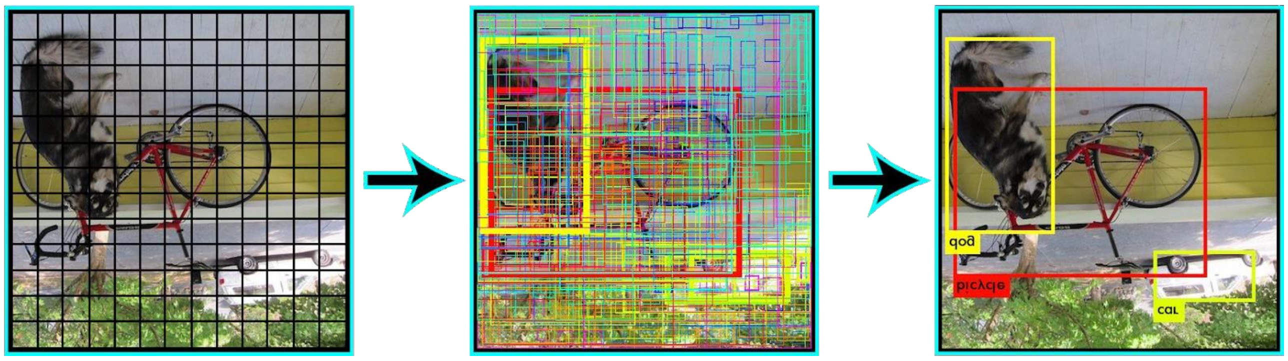


FIG 8.1

In Non Maximal Suppression, YOLO suppresses all bounding boxes that have lower probability scores.

YOLO achieves this by first looking at the probability scores associated with each decision and taking the largest one. Following this, it suppresses the bounding boxes having the largest Intersection over Union with the current high probability bounding box.

This step is repeated till the final bounding boxes are obtained.

THE COCO PART

The pre-trained COCO-SSD model is on the fast-but-less-accurate side of the spectrum of all tensorflow architectures, allowing it to be used in a browser.

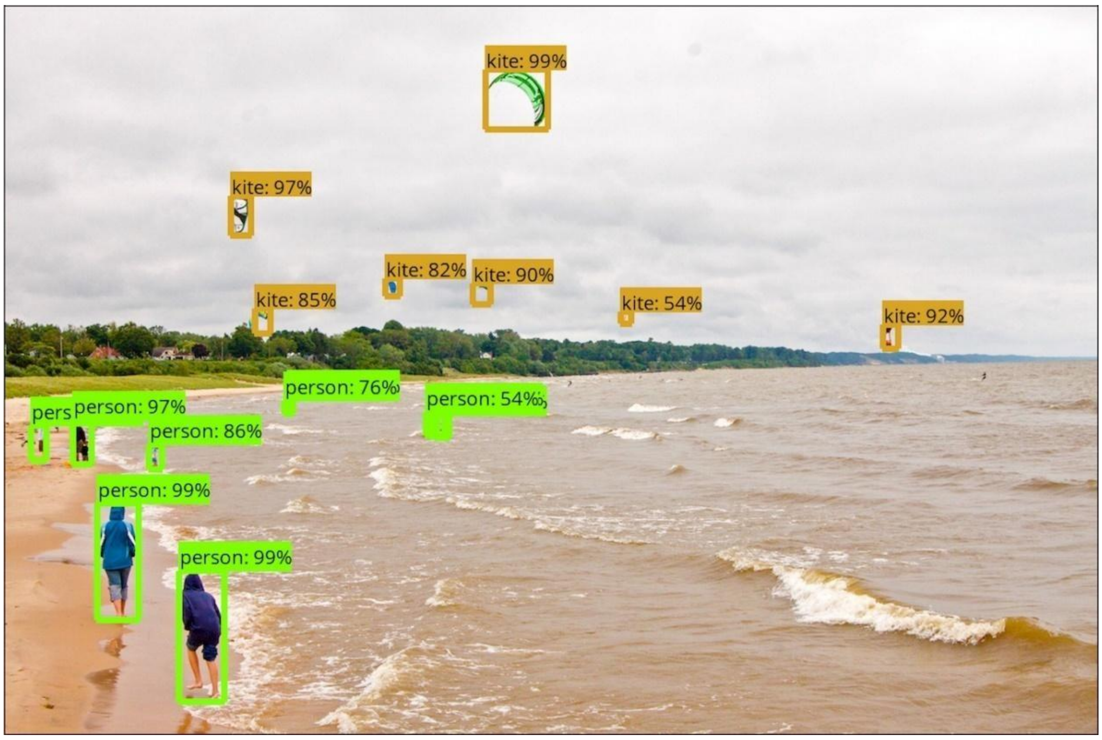


FIG 8.2

COCO refers to the "Common Objects in Context" dataset, the data on which the model was trained on. This collection of images is mostly used for object detection, segmentation, and captioning, and it consists of over 200k labeled images belonging to one of 90 different categories, such as "person," "bus," "zebra," and "tennis racket." SSD, short for Single Shot Detector, is a neural network architecture made of a single feed-forward convolutional neural network that predicts the image's objects labels and their position during the same action. The counterpart of this "single-shot" characteristic is an architecture that uses a "proposal generator," a component whose purpose is to search for regions of interest within an image.

Once the regions of interests have been identified, the second step is to extract the visual features of these regions and determine which objects are present in them, a process known as "feature extraction." COCO-SSD default's feature extractor is lite_mobilenet_v2, an extractor based on the MobileNet architecture. In general, mobileNet is designed for low resources devices, such as mobile, single-board computers, e.g., Raspberry Pi, and even drones.

The Coding Part:

```
# USAGE
# python real_time_object_detection.py

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
'''ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())'''

# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike",
"person",  "sheep","mobile","purse","book",
    "sofa", "train", "tvmonitor","mobile","pen"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load our serialized model from disk
```

```
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe('MobileNetSSD_deploy.prototxt.txt',
'MobileNetSSD_deploy.caffemodel')

# initialize the video stream, allow the cammera sensor to warmup,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
fps = FPS().start()

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # grab the frame dimensions and convert it to a blob
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)

    # pass the blob through the network and obtain the detections
and
    # predictions
    net.setInput(blob)
    detections = net.forward()

    # loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated
with
        # the prediction
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the `confidence`
is
        # greater than the minimum confidence
        if confidence > 0.2:
            # extract the index of the class label from the
            # `detections`, then compute the (x, y)-coordinates of
            # the bounding box for the object
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # draw the prediction on the frame
            label = "{}: {:.2f}%".format(CLASSES[idx],
                confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                COLORS[idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

    # update the FPS counter
    fps.update()

# stop the timer and display FPS information
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Chapter 9

Screenshots of project

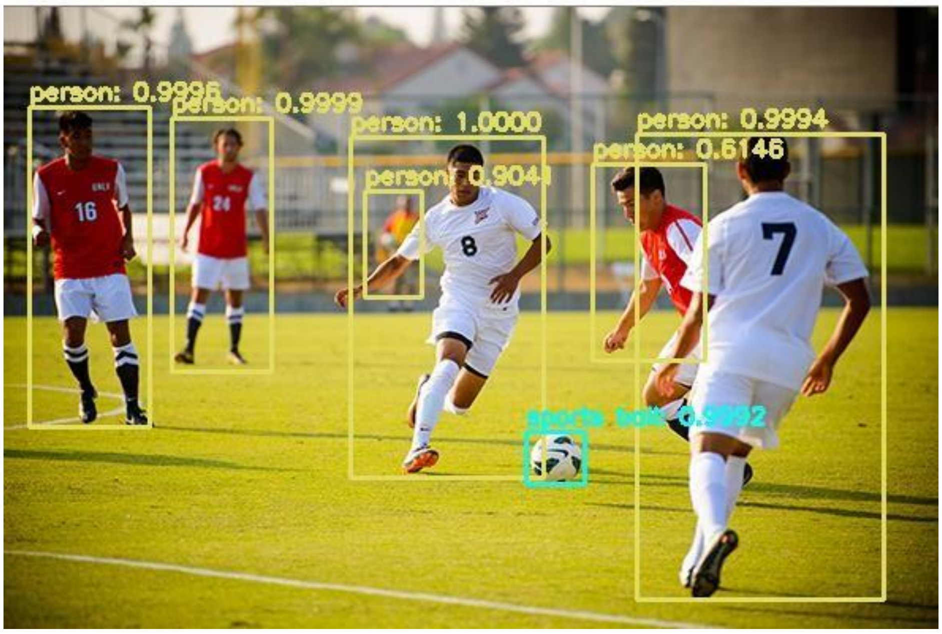
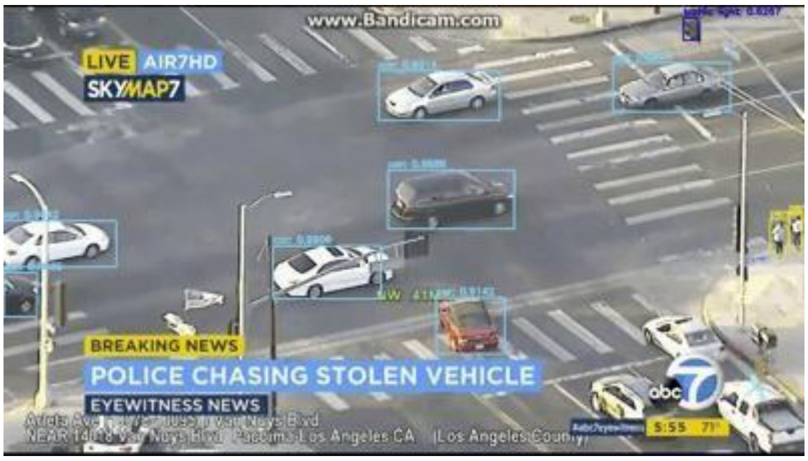


Fig 9.1



YOLO is able to correctly detect each of the players on the pitch, including the soccer ball itself. Notice the person in the background who is detected despite the area being highly blurred and partially obscured.

Fig 9.2

In the video/GIF, you can see not only the vehicles being detected, but people, as well as the traffic lights, are detected too.The YOLO object detector is performing quite well here

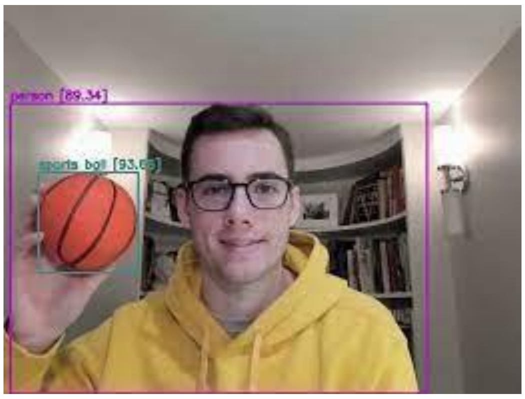


Fig 9.3

In this we use our webcam for testing to see how our model detects object in real time .

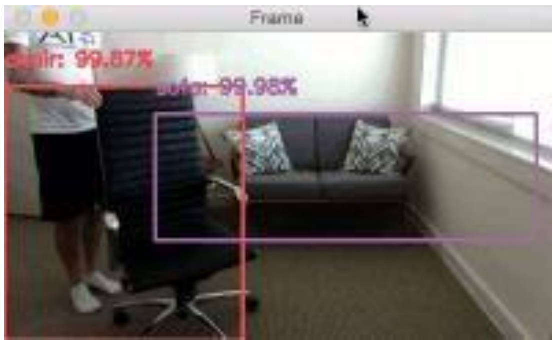


Fig 9.4

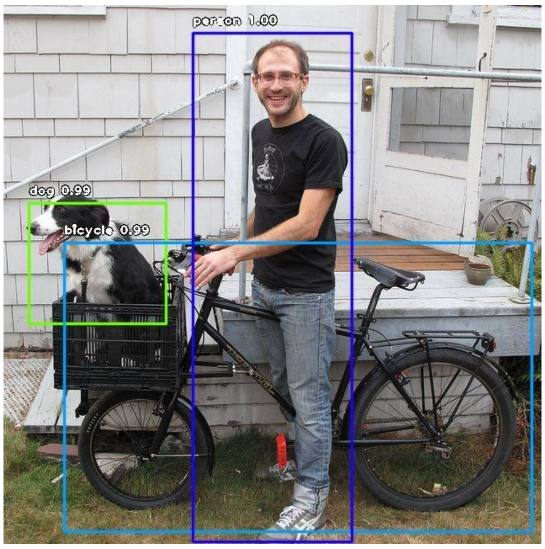


Fig 9.5

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

In this project, we proposed about YOLO algorithm for the purpose of detecting 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 to the entire image in predicting boundaries. And also it predicts fewer false positives in background areas. Comparing to other classifier algorithms this algorithm is much more efficient and fastest algorithm to use in real time.

We also trained this YOLO algorithm with COCO dataset in order to achieve the 90% accuracy. Also COCO dataset contains 200K labelled dataset which can be used to train and detect over 80 objects.

Computer vision (CV) is a fascinating field of study that attempts to automate the process of assigning meaning to digital images or videos. In other words, we are helping computers see and understand the world around us! A number of machine learning (ML) algorithms and techniques can be used to accomplish CV tasks, and as ML becomes faster and more efficient, we can deploy these techniques to embedded systems. This course, offered by a partnership among Edge Impulse, OpenMV, Seeed Studio, and the TinyML Foundation, will give you an understanding of how deep learning with neural networks can be used to classify images and detect objects in images and videos. You will have the opportunity to deploy these machine learning models to embedded systems, which is known as embedded machine learning or TinyML. Familiarity with the Python programming language and basic ML concepts (such as neural networks, training, inference, and evaluation) is advised to understand some topics as well as complete the projects. Some math (reading plots, arithmetic, algebra) is also required for quizzes and projects. If you have not done so already, taking the "Introduction to Embedded Machine Learning" course is recommended. This course covers the concepts and vocabulary necessary to understand how convolutional neural networks (CNNs) operate, and it covers how to use them to classify images and detect objects. The hands-on projects will give you the opportunity to train your own CNNs and deploy them to a microcontroller and/or single board computer.

10.2 FUTURE SCOPE

Most existing algorithms only tackle a small subset of the different tasks necessary for understanding an image and are very expensive computationally. In order to reproduce a fraction of the average person's ability to detect objects, one would have to combine several different algorithms to make a combined system that runs in real time, an enormous challenge with today's hardware.

Indeed, object detection is a key task for most computer and robot vision systems. Although there has been great progress in the last several years, there will be even bigger improvements in the future with the advent of artificial intelligence in conjunction with existing techniques that are now part of many consumer electronics or have been integrated in assistant driving technologies.

We can implement this project in various fields such as Anti- theft Analytics System, House Security Cameras, Anti-Theft Vehicle System by using it in CCTV on roads and many more use cases are there for this project.

Autonomous Control of a Robot

One of the goals suggested at the beginning of the project was to build a helicopter robot, complete with a system control that could be operated by a user or could be implemented as an autonomous control system. One of the modules that would be required for such a task would be a tracker, capable of relaying back to the control center the current location of the robot at any given time. Acknowledging the criticisms given in section 5.3, this tracker could be improved further by perhaps providing a more sophisticated algorithm (such as [HLGT03]) and deploying the system on a faster machine. Then this tracker could possibly be used in such an environment.

Optimisations and Distribution

There are many optimisations that could be made to the code. For example, the threaded nature of the visualizer can cause a delay in the point cloud rendering. This is probably due to the effiffifficiency of the application code. With more time, this could perhaps have been optimised to allow for a smoother rendering process. Another more UI friendly feature that could have been added was a command panel allowing the user to open and close particular parts of the system at will. This however, was not implemented for this project as the author believed it to be unnecessary in demonstrating what the project does. The most important optimisation as far as this author is concerned would be in creating a distributed environment for the code to run. As ROS natively accomodates the use of multiple machine control of the operating system, it is obvious that the next step would be in utilising this to speed up the system. As the tracker is already modular, components could easily be executed on separate systems, allowing for more processing to be done per frame.
This processing could range from adding an extra fifiltering mechanism to the optical tracker, rendering a faster point cloud, and producing more relevant data than the location of the object (velocity etc.). Of course by lessons learned from the roboearth detector, the system could be modified to also include a model of the object being detected. A single machine could be used to load the model alone, allowing another machine to process the system itself, providing even better performance. This would enable the tracker to fifilter out unwanted objects and also provide a fast tracking paradigm with pose estimation thanks to the model.

Closing Comments

This project applied modern sensor technology to the tracking problem. With the help of rich API's, a system that does indeed perform as is stated in this report was built. This report detailed the process in taking the project from formal specififfication through to design and implementation. The strengths and flflaws of the system were discussed and analysed, and a comparison was made with similar system. This report has hopefully presented to the user with a best understanding possible of the tracking system developed. Accompanying the report is the project implementation along with a generated documentation directory. This directory contains instructions for installa tion of the system as well as details on the code implementation. It is the author's wish that the reader may attempt to install the system where possible, to get a direct interpretation of the system during execution.

Point Tracking

Moving object can be represented by point in an image structure. An identification of point in a moving object is done by threshold value. Point tracking is difficult when occlusion will be appeared. Some of the points tracking algorithms described in [38] are: Kalman Filter, Particle Filter and Multiple Hypothesis tracking.

Kernel Tracking

Kernel tracking tracks the moving objects which are represented by growing object region from one frame to another. The geometric object representation is common in real time. But the restriction here is that, the part of the moving object defined may be left outside of the region or some background object may be covered by the region. The various kernel tracking methods are [39]: Simple template matching, Dual Tree Complex Wavelet Transform, Layering based tracking, Support Vector Machine and Color Histogram or Mean Shift Method.

Tracking objects

An item/object detection framework is additionally utilized in tracking the objects, for instance tracking a ball during a match in the football world cup, tracking the swing of a cricket bat, tracking an individual in a video.Object tracking has an assortment of uses, some of which are surveillance and security, traffic checking, video correspondence, robot vision and activity.

People Counting

Object detection can be additionally utilized for People counting.It is utilized for dissecting store execution or group measurements during festivals. These will, in general, be progressively troublesome as individuals move out of the frame rapidly (likewise in light of the fact that individuals are non-inflexible objects).

Automated CCTV surveillance

Surveillance is a necessary piece of security and watch. Ongoing advances in computer vision innovation need to prompt the improvement of different programmed surveillance systems. Be that as it may, their viability is influenced by numerous factors and they are not totally dependable. This examination researched the capability of an automated surveillance system to diminish the CCTV administrator outstanding task at hand in both discovery and following exercises.

Typically CCTV is running inevitably, so we need a huge size of the memory framework to store the recorded video. By utilizing an object discovery framework we can mechanize CCTV so that in the event that a few items are detected, at that point the record is going to begin. Utilizing this we can diminish the over and over account a similar picture outlines, which expands memory effectiveness. We can diminish the memory prerequisite by utilizing this object detection system.

Person Detection

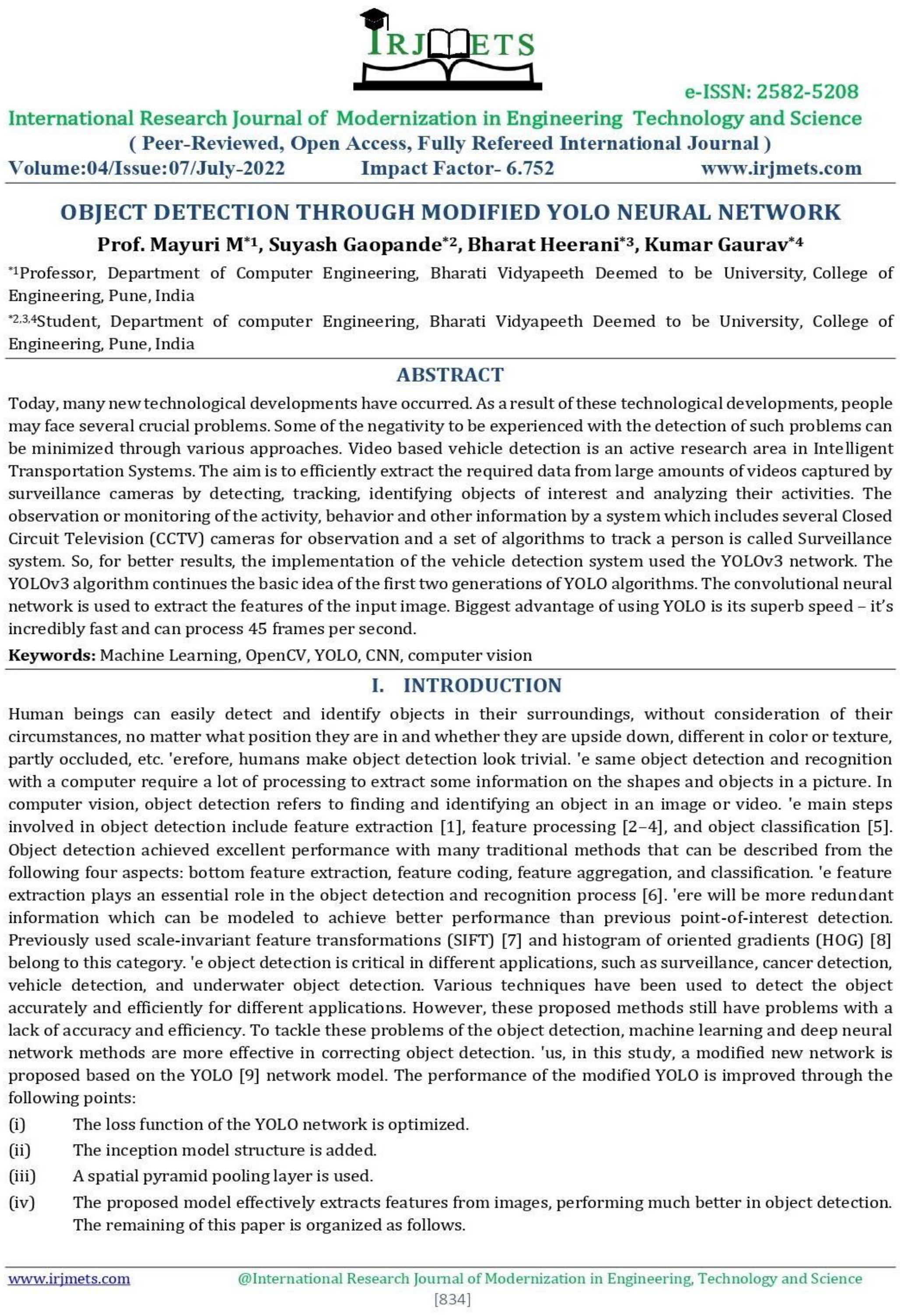
Person detection is necessary and critical work in any intelligent video surveillance framework, as it gives the essential data to semantic comprehension of the video recordings. It has a conspicuous augmentation to automotive applications because of the potential for improving security frameworks. Person detection is undertakings of Computer vision frameworks for finding and following individuals. Person detection is the task of finding all examples of individuals present in a picture, and it has been most broadly achieved via looking through all areas in the picture, at all potential scales, and contrasting a little region at every area with known layouts or examples of individuals. Person detection is commonly viewed as the initial procedure in a video surveillance pipeline and can take care of into more significant level thinking modules, for example, action recognition and dynamic scene analysis.

Vehicle Detection

Vehicle Detection is one of the most important part in our daily life. As the world is moving faster and the numbers of cars are keep on increasing day by day, Vehicle detection is very important. By using Vehicle Detection technique we can detect the number plate of a speeding car or accident affected car. This also enables for security of the society and decreasing the number of crimes done by car. By using Vehicle Detection Technology Pixel Solutionz have successfully detected the speed of the vehicle and we have also detected the number plate of the car using Optical Character Recognition (OCR). By detecting the Number plate, Pixel Solutionz managed to measure the speed of the vehicle and for and oil company we have successfully developed a Safety Alert System with collision detection warning alert.

Chapter 11

Research Publication



e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:07/July-2022 Impact Factor- 6.752 www.irjmets.com

OBJECT DETECTION THROUGH MODIFIED YOLO NEURAL NETWORK

Prof. Mayuri M*[1], Suyash Gaopande*[2], Bharat Heerani*[3], Kumar Gaurav*[4]

*[1]Professor, Department of Computer Engineering, Bharati Vidyapeeth Deemed to be University, College of Engineering, Pune, India

*[2,3,4]Student, Department of computer Engineering, Bharati Vidyapeeth Deemed to be University, College of Engineering, Pune, India

ABSTRACT

Today, many new technological developments have occurred. As a result of these technological developments, people may face several crucial problems. Some of the negativity to be experienced with the detection of such problems can be minimized through various approaches. Video based vehicle detection is an active research area in Intelligent Transportation Systems. The aim is to efficiently extract the required data from large amounts of videos captured by surveillance cameras by detecting, tracking, identifying objects of interest and analyzing their activities. The observation or monitoring of the activity, behavior and other information by a system which includes several Closed Circuit Television (CCTV) cameras for observation and a set of algorithms to track a person is called Surveillance system. So, for better results, the implementation of the vehicle detection system used the YOLOv3 network. The YOLOv3 algorithm continues the basic idea of the first two generations of YOLO algorithms. The convolutional neural network is used to extract the features of the input image. Biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second.

Keywords: Machine Learning, OpenCV, YOLO, CNN, computer vision

I. INTRODUCTION

Human beings can easily detect and identify objects in their surroundings, without consideration of their circumstances, no matter what position they are in and whether they are upside down, different in color or texture, partly occluded, etc. 'erefore, humans make object detection look trivial. 'e same object detection and recognition with a computer require a lot of processing to extract some information on the shapes and objects in a picture. In computer vision, object detection refers to finding and identifying an object in an image or video. 'e main steps involved in object detection include feature extraction [1], feature processing [2–4], and object classification [5]. Object detection achieved excellent performance with many traditional methods that can be described from the following four aspects: bottom feature extraction, feature coding, feature aggregation, and classification. 'e feature extraction plays an essential role in the object detection and recognition process [6]. 'ere will be more redundant information which can be modeled to achieve better performance than previous point-of-interest detection. Previously used scale-invariant feature transformations (SIFT) [7] and histogram of oriented gradients (HOG) [8] belong to this category. 'e object detection is critical in different applications, such as surveillance, cancer detection, vehicle detection, and underwater object detection. Various techniques have been used to detect the object accurately and efficiently for different applications. However, these proposed methods still have problems with a lack of accuracy and efficiency. To tackle these problems of the object detection, machine learning and deep neural network methods are more effective in correcting object detection. 'us, in this study, a modified new network is proposed based on the YOLO [9] network model. The performance of the modified YOLO is improved through the following points:

(i) The loss function of the YOLO network is optimized.

(ii) The inception model structure is added.

(iii) A spatial pyramid pooling layer is used.

(iv) The proposed model effectively extracts features from images, performing much better in object detection. The remaining of this paper is organized as follows.

IRJMETS
e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:07/July-2022 Impact Factor- 6.752 www.irjmets.com

II. TECHNOLOGY CLASSIFIERS

- YOLO: - You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a map of 57.9% on COCO test-dev.
- OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.
- Computer Vision: - Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

III. PROPOSED APPRAOCH

A. Proposed Architecture

Architecture describes how the application is going to function. Project Architecture of Object Detection describes how the user's request is taken as input and how the output is delivered. The detailed architecture is shown in Fig 1- Project Architecture of Object Detection

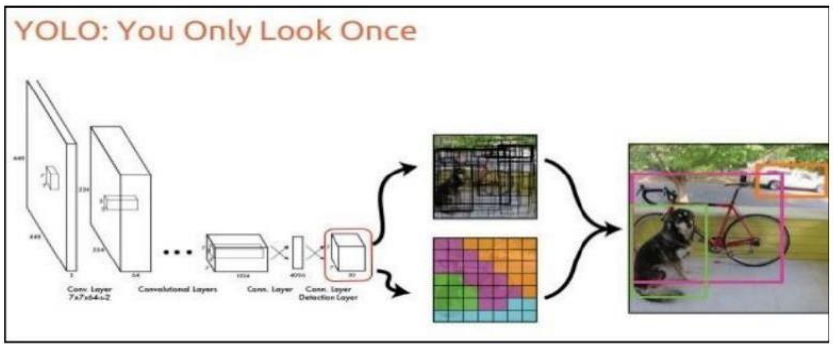


Fig 1. Project Architecture of Object Detection

B. Modules Description

Two main Modules used in the product are User and YOLO USER module perform two functionalitiesRun program: This module allows the user to run the program by giving an input video or image file. Attain Feedback: This module allows the user to receive the output in the form of video or image file saved in certain destination. YOLO module perform four functionalities- Object Registration: This module allows the object to be registered with the model. Object Detection: This module allows the product to detect the object based on the training received. Frame Analysis: This module allows the model to analyze the capture frame. Generate Prediction: This module allows the module to generate a text prediction of what the object is and where the object is present Workflow is stated with use case diagram. In the use case diagram, we basically have two actors namely: the User, System with YOLO. The user has the following methods, receive instructions and give requests. The User perform action to enroll the input file and retrieve the output file .System with YOLO will generate the output file by using the COCO dataset which is defined inside. The detailed usecase is shown in Fig 2- Use Case Diagram for User and System for VLX–A Classifieds Platform.

IRJMETS
e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:07/July-2022 Impact Factor- 6.752 www.irjmets.com

Working of ORB

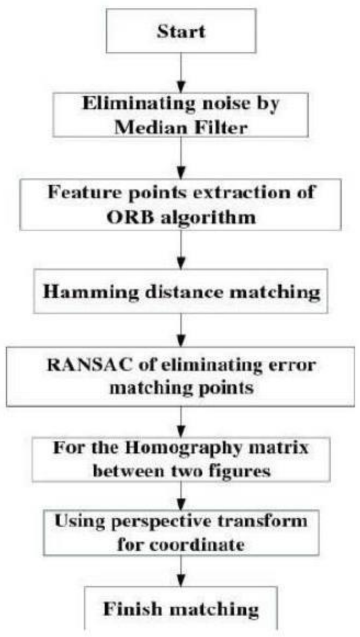

Start
Eliminating noise by Median Filter
Feature points extraction of ORB algorithm
Hamming distance matching
RANSAC of eliminating error matching points
For the Homography matrix between two figures
Using perspective transform for coordinate
Finish matching

Fig-5. Working of ORB

V. IMPLEMENTATION

For realistic execution we are using Operating System Multi Platform (Windows 7 & above, Linux GCC),and Backend as Python 3.6 & above. Dataset as COCO (Common Objects In Context)and Machine Learning Model YOLO V3 (You Only Look Once). Using this application, we can detect the objects and specify them. In order to use the application the user has to run the application and can upload a video file or image file to the program by giving the file path. It is designed to detect many objects with the specification. It can easily find out common objects such as chairs, remotes, bus etc. The application gives a glimpse, of where the object is located with the accuracy. The core features of this project are it provides feedback in the form of video file or image file, it detects most of the common objects in context. The other features include detection of each image is reported with some form of pose information. For example, for face detection in a face detector system compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face.

A. Input File Location

Input video files storage location is shown in Fig-6

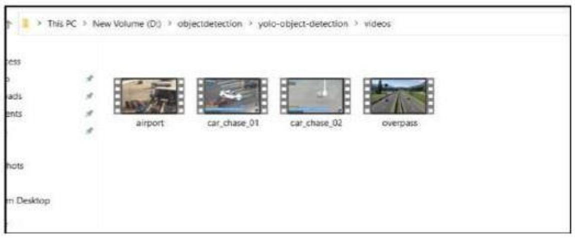

This PC > New Volume (D:) > objectdetection > yolo-object-detection > videos
airport
car_chase_01
car_chase_02
overpass

Fig-6. Input file location

IRJMETS
e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:07/July-2022 Impact Factor- 6.752 www.irjmets.com

B. Output File Location

The output video files are stored after detecting the objects is shown in Fig-7

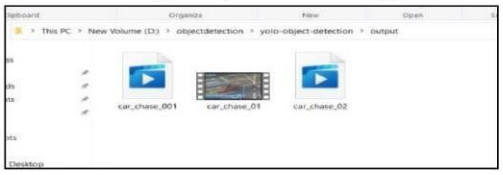

Clipboard
Organize
New
Open
This PC > New Volume (D:) > objectdetection > yolo-object-detection > output
car_chase_001
car_chase_01
car_chase_02
Desktop

Fig-7. Output file location

C. Video Frame During Object Detection

The video frame looks when system detect the required object is shown in Fig-8

Fig-8. Video frame during object detection

D. Object Detection

Video Frame looks when it recognizes an object is shown in Fig-9

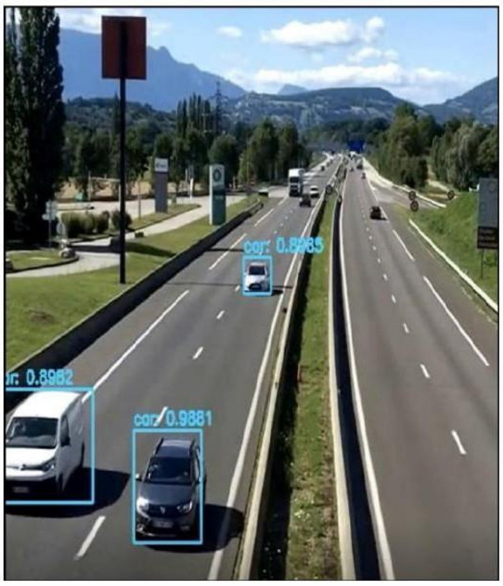
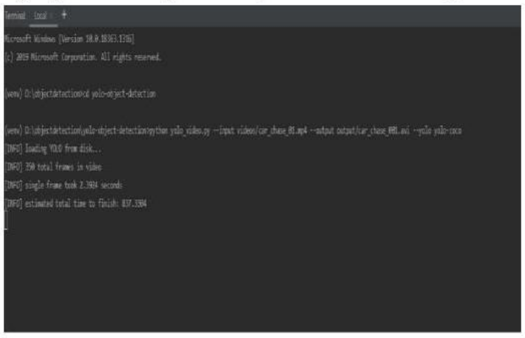


Fig-9. Object detection

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:07/July-2022 Impact Factor- 6.752 www.irjmets.com

E. Output Generating Command

The command used to run the program and to get the output file is shown in Fig-10



```
Terminal  Local  +
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

(venv) D:\objectdetection>cd yolo-object-detection

(venv) D:\objectdetection\yolo-object-detection>python yolo_video.py --input videos/car_chase_01.mp4 --output output/car_chase_001.avi --yolo yolo-coco
[INFO] loading YOLO from disk...
[INFO] 350 total frames in video
[INFO] single frame took 2.3934 seconds
[INFO] estimated total time to finish: 837.3384
```

Fig-10. Output Generating command

VI. CONCLUSION & FUTURE ENHANCEMENT

Approach helps in increasing the accuracy and speed and achieves the desired results. By using method, we are able to detect object more precisely and identify the objects individually with exact location of an object in the picture in x, y axis. implementations of the YOLO algorithm on the web using Darknet is one open-source neural network framework. Darknet was written in the C Language, which makes it really fast and provides for making computations on a GPU, essential for real-time predictions. The object detection system can be applied in the area of surveillance system, face recognition, fault detection, character recognition etc. In future we can add a faster model that runs on the GPU and use a camera that provides a 360 field of view and allows analysis completely around the person. We can also include a Global Positioning System and allow the person to detect the objects instantly without any delay in frames and seconds.

ACKNOWLEDGMENT

This study, as well as the research that went into it, would not have been feasible without Prof. Mayuri Molawade's invaluable assistance. From our initial interaction with the issue through the final copy of this paper, his excitement, expertise, and meticulous attention to detail have been an inspiration and keptus on track. Prof. Mayuri Molawade has provided the essential knowledge from his book, which he has been collecting for over two decades. We also appreciate the informative feedback provided by anonymous peer reviewers at Books & Texts. The generosity and competence of one and all have enriched the study in uncountable ways and rescued us from many errors, but we are alone responsible for those that remain.

We are grateful to the BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY). These factors enabled us to extend our book research much beyond our expectations. The team members' creative and technical labor resulted in the final design of the intricate transcription tables in the Appendices, and the paper's language and layout benefitted greatly. Finally,it is with great pleasure that we thank the effort of this incredible team, which has sacrificed countless Friday evenings and Sunday afternoons to read every version of this document and the replies it has elicited with a unique blend of compassionand critique.

VII. REFERENCES

[1] A. Tiwari, A. Kumar, and G. M. Saraswat, "Feature extraction for object recognition and image classification," International Journal of Engineering Research & Technology (IJERT), vol. 2, pp. 2278–0181, 2013.

[2] J. Yan, Z. Lei, L. Wen, and S. Z. Li, "'e fastest deformable part model for object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2497–2504, New York, NY, USA, 2014.

[3] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000

References

[1] P. Chakravorty, "What Is a Signal? [Lecture Notes]," in IEEE Signal Processing Magazine, vol. 35, no. 5, pp. 175-177, Sept. 2018, doi: 10.1109/MSP.2018.2832195.
[2] M. Rouse, “image” [Online]. Available at: https://whatis.techtarget.com/definition/image. [Accessed: 01/10/20]
[3] Merriam-Webster “image” [Online]. Available at: https://www.merriam-webster.com/dic tionary/image, [Assessed: 01/10/20]
[4] Wikipedia, “Image” [Online]. Available at: https://en.wikipedia.org/wiki/Image, [Assessed: 01/10/20]
[5] The Editors of Encyclopedia Britannica, “Image-processing” [Online], Available at: https://www.britannica.com/technology/image-processing. [Assessed: 01/10/20]
[6] Wikipedia, “Encoding images” [Online], Available at: https://www.bbc.co.uk/bitesize/guides/zqyrq6f/revision/3. [Assessed: 02/10/20]
[7] Object (image processing) [Online], Available at: https://en.wikipedia.org/wiki/Object_(im age_processing) , [Assessed: 02/10/20]
[8] P. Ganesh, Object Detection: Simplified [Online], Available at: https://towardsdatasci ence.com/object-detection-simplified-e07aa3830954, [Assessed: 02/10/20]
[9] Tensorflow, Available at: https://www.tensorflow.org/lite/models/object_detection/overview, [Assessed: 03/10/20]
[10] Wikipedia, “Object detection” Available at: ‘https://en.wikipedia.org/wiki/Object_detection, [Assessed: 02/10/20]
[11] Fritz, “Object Detection Guide”, Available at: https://www.fritz.ai/object-detection/, [As sessed: 03/10/20]
[12] Wikipedia, “Outline of object recognition”, Available at: https://en.wikipedia.org/wiki/Out line_of_object_recognition , [Assessed: 02/10/20]
[13] Object Recognition, Available at: cse.usf.edu/~r1k/MachineVisionBook/Machine Vision.files/MachineVision_Chapter15.pdf, [Assessed: 03/10/20]
[14] N. Pinto, D. D. Cox, and J.J. DiCarlo, “Why is Real-World Visual Object Recognition Hard?” (2008) *PLoS* Comput Biol 4(1): e27 .
[15] A. Gulli and P. Sujit, “Deep Learning with Keras” (2017), Available at: https://1lib.us/book/3411804/7ea47a?id=3411804. [Assessed: 03/10/20]
[16] “The Definitive Glossary of Higher Mathematical Jargon - Algorithm". Available at: https://mathvault.ca/math-glossary/ [Assessed: 03/10/20]
[17] “Definition of ALGORITHM". Merriam-Webster Online Dictionary. Available at: https://www.merriam-webster.com/dictionary/algorithm [Assessed: 04/10/20]
[18] Y. Gavrilova, “Artificial Intelligence vs. Machine Learning vs. Deep Learning: Essentials” Available at https://serokell.io/blog/ai-ml-dl-difference [Assessed: 04/10/20]
[19] J. Brownlee, “A Gentle Introduction to Object Recognition with Deep Learning” (2018) Available at: https://machinelearningmastery.com/object-recognition-with-deep-learn ing/ [Assessed: 01/10/20]60
[20] A. Kamal, “YOLO, YOLOv2 and YOLOv3: All You want to know”(2019) Available at: https://medium.com/@amrokamal_47691/yolo-yolov2-and-yolov3-all-you-want-to know-7e3e92dc4899 [Assessed: 04/10/20]
[21] You Only Look Once: Unified, Real-Time Object Detection, 2015. Available at: https://arxiv.org/abs/1506.02640, [Assessed: 04/10/20]
[22] A. Rosebrock., “Intersection over Union (IoU) for object detection” (2016) Available at:

https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-de
tection/, [Assessed: 04/10/20]
[23] I. Tan, “Measuring Labelling Quality with IOU and F1 Score”, Available at: https://me
dium.com/supahands-techblog/measuring-labelling-quality-with-iou-and-f1-score-
1717e29e492f , [Assessed: 01/10/20]
[24] StackOverflow, “Intersection Over Union (IoU) ground truth in YOLO”, Available at:
https://stackoverflow.com/questions/61758075/intersection-over-union-iou-ground-truth
in-yolo, [Assessed: 02/10/20]
[25] J. Redmon & A. Farhadi, (University of Washington), “YOLO9000: Better, Faster,
Stronger” Available at: https://pjreddie.com/media/files/papers/YOLO9000.pdf, [As
sessed: 02/10/20]
[26] “YOLO v2 – Object Detection” Available at: https://www.geeksforgeeks.org/yolo-v2-ob
ject-detection/ [Assessed: 04/10/20]
[27] A. Aggarwal, “YOLO Explained” *Available at:* https://medium.com/analytics-vidhya/yolo
explained-5b6f4564f31 [Assessed: 04/10/20]
[28] K. Mahesh Babu, M.V. Raghunadh, *Vehicle number plate detection and recognition using
bounding box method*, May 2016, pp 106–110
[29] L. Cai, F. Jiang, W. Zhou, and K. Li, Design and Application of An Attractiveness Index for
Urban Hotspots Based on GPS Trajectory Data, (Fellow, IEEE), pg 4
[30] Wikipedia, “Minimum bounding box” Available at: https://en.wikipedia.org/wiki/Mini
mum_bounding_box, [Assessed: 04/10/20]
[31] Dive into Deep Learning, “13.3. Object Detection and Bounding Boxes” Available at:
https://d2l.ai/chapter_computer-vision/bounding-box.html , [Assessed: 04/10/20]
[32] J. Redmon & A. Farhadi, YOLOv3: An Incremental Improvement, University of Washing
ton Available at: https://arxiv.org/abs/1804.02767 [Assessed: 05/10/20]
[33] YOLO: You Only Look Once, Available at: jeremyjordan.me/object-detection-one
stage/#yolo, [Assessed: 05/10/20]
[34] Longman Dictionary, Definition of training, Available at: https://www.ldoceonline.com/dic
tionary/training [Assessed: 05/10/20]
[35] Guangrui Liu “Real-Time Object Detection for Autonomous Driving Based on Deep Learn
ing” (2017), Available at: https://tamucc-ir.tdl.org/handle/1969.6/5637 [Assessed:
03/18/21]
[36] A. Abdulkader & C. Vlahija “Real-time vehicle and pedestrian detection, a data-driven rec
ommendation focusing on safety as a perception to autonomous vehicles”, Available at:
http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1479957&dswid=-3676 [As
sessed: 03/18/21]61
[37] Enhancement methods in image processing Available at: https://www.mathworks.com/dis
covery/image-enhancement.html. [Assessed: 03/18/21]
[38] Evaluating a machine learning model. Available at: https://www.jeremyjordan.me/evaluat
ing-a-machine-learning-model/. [Assessed: 03/18/21]
[39] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Available at
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-net
works-the-eli5-way-3bd2b1164a53. [Assessed: 03/18/21]
[40] J. Jokela “Person Counter Using Real-Time Object Detection and a Small Neural Network”
Turku University of Applied Sciences. Available at: https://www.theseus.fi/bit
stream/handle/10024/153489/Jokela_Jussi.pdf?sequence=1&isAllowed=y. [Assessed:
03/18/21]
[41] Manishgupta “YOLO – You Only Look Once”. Available at: https://towardsdatasci
ence.com/yolo-you-only-look-once-3dbdbb608ec4. [Assessed: 03/18/21]

[42] E. Y. Li "Dive Really Deep into YOLO v3: A Beginner's Guide". Available at: https://to
wardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e.
[Assessed: 03/18/21]
[43] F. Zelic & A. Sable "A comprehensive guide to OCR with Tesseract, OpenCV and Py
thon.". Available at: https://nanonets.com/blog/ocr-with-tesseract/. [Assessed: 03/18/21]
[44] Tessdoc. Available at: https://github.com/tesseract-ocr/tessdoc/blob/master/ImproveQual
ity.md. [Assessed: 06/04/21]
[45] J. Goyvaerts "Regular Expressions: The Complete Tutorial". Available at: https://www.reg
ular-expressions.info/print.html. [Assessed: 06/04/21]
[46] M. Erwig & R. Gopinath, "Explanations for Regular Expressions". Available at:
https://web.engr.oregonstate.edu/~erwig/papers/ExplRegExp_FASE12.pdf. [Assessed:
06/04/21]