SMART ALARMING SYSTEM USING OBJECT DETECTION

A PROJECT WORK REPORT Submitted to

Jawaharlal Nehru Technological University Hyderabad

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted by

B. Aman Rai Saxena	**16E11A1218**
G. Varun Dev	**16E11A1217**
M. Nikhil	**16E11A1209**
O. Chaitanya	**16E11A1235**

Under the Supervision of

Mrs. G. Rashmi
Assistant Professor

Department of Information Technology

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana.

APRIL 2020

DEPARTMENT OF INFORMATION TECHNOLOGY

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana

Certificate

*This is to certify that the project work entitled “***Smart Alarming System Object Detection”** *is the bonafide work done.*

By

B. Aman Rai Saxena	**16E11A1218**
G. Varun Dev	**16E11A1217**
M. Nikhil	**16E11A1209**
O. Chaitanya	**16E11A1235**

in the Department of Information Technology, **BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY,** *Ibrahimpatnam is submitted to* **Jawaharlal Nehru Technological University, Hyderabad** *in partial fulfillment of the requirements for the award of* **B. Tech** *degree in* **Information Technology** *during* ***2016–2020*.**

Guide:

Mrs. G.Rashmi
Assistant Professor
Dept of IT,
Bharat Institute of Engineering and Technology,
Ibrahimpatnam– 501 510, Hyderabad.

Head of the Department:

Mr. Madana Mohan
Professor
Dept of CSE,
Bharat Institute of Engineering and Technology,
Ibrahimpatnam– 501 510, Hyderabad.

Viva-Voice held on…………………………………………

Internal Examiner

External Examiner

DEPARTMENT OF INFORMATION TECHNOLOGY

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana

vision of the Institution

To achieve the autonomous & university status and spread universal education by inculcating discipline, character and knowledge into the young minds and mould them into enlightened citizens.

Mission of theInstitution

Our mission is to impart education, in a conducive ambience, as comprehensive as possible, with the support of all the modern technologies and make the students acquire the ability and passion to work wisely, creatively and effectively for the betterment of our society.

vision of ITdepartment

Serving the high quality educational needs of local and rural students within the core areas of Computer Science and Engineering and Information Technology through a rigorous curriculum of theory, research and collaboration with other disciplines that is distinguished by its impact on academia, industry and society.

Mission of IT department

The Mission of the department of Information Technology and Engineering is

- To work closely with industry and research organizations to provide high quality computer education in both the theoretical and applications of Computer Science and Engineering.
- The department encourages original thinking, fosters research and development, evolve innovative applications of technology.

DEPARTMENT OF INFORMATION TECHNOLOGY

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Information Technology and Engineering program provides students with an in depth education in the conceptual foundations of computer science and in complex hardware and software systems. It allows them to explore the connections between computer science and a variety of other disciplines in engineering and outside. Combined with a strong education in mathematics, science, and the liberal arts it prepares students to be leaders in computer science practice, applications to other disciplines and research.

Program Educational Objective 1: (PEO1)

The graduates of Information Technology and Engineering will have successful career in technology or managerial functions.

Program Educational Objective 2: (PEO2)

The graduates of the program will have solid technical and professional foundation to continue higher studies.

Program Educational Objective 3: (PEO3)

The graduates of the program will have skills to develop products, offer services and create new knowledge.

Program Educational Objective 4: (PEO4)

The graduates of the program will have fundamental awareness of Industry processes, tools and technologies.

DEPARTMENT OF INFORMATION TECHNOLOGY

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana.

PROGRAM OUTCOMES (POs)

PO1:	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2:	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3:	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4:	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5:	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6:	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7:	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8:	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9:	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10:	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11:	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12:	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DEPARTMENT OF INFORMATION TECHNOLOGY

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1:	**Foundation of mathematical concepts:** To use mathematical methodologies to crack problem using suitable mathematical analysis, data structure and suitable algorithm.
PSO2:	**Foundation of Computer System:** The ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems.
PSO3:	**Foundations of Software development:** The ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of software design process. Familiarity and practical proficiency with a broad area of programming concepts and provide new ideas and innovations towards research.

DEPARTMENT OF INFORMATION TECHNOLOGY

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programs: CSE, ECE, EEE & Mechanical)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpatnam-501 510, Hyderabad, Telangana

QUALITY OF THE PROJECT

I. Consideration to Factors

Factors ***(Environment, Safety, Ethics, Cost)***	**Type of Project** ***(Application, Product, Research, Review, etc.)***	**Standards**
Security	Application	arXiv journal

II. POs and PSOs addressed through the project with justification

S. No.	**POs and PSOs addressed**	**Justification**
1.	**PO1 Engineering Knowledge:**	Applying Knowledge of Engineering Fundamentals(Python)
2.	**PO2 Problem Analysis:**	Identifying proper methods to solve Engineering problems reaching conclusions using Machine Learning
3.	**PO3 Design of Solution:**	Designing solutions for security personnel and easing their work.
4.	**PO4 Conduct investigation on complex problems:**	Security surveillance is an important aspect to identify an intruder.
5.	**PO5 Modern Tool Usage:**	We have used python language with appropriate algorithms.

6.	**PO6 The Engineer and Society:**	By Engineering Knowledge we have resolved some issues with the project.
7.	**PO7 Environment and Sustainability:**	By the impact of professional engineering we have imposed social development.
8.	**PO8 Ethics:**	We have applied ethical principles and responsibilities of engineering practice.
9.	**PO9 Individual and Team Work:**	We have distributed the work according to our strengths and equally worked for the project.
10.	**PO10 Communication:**	Communicated effectively through reports, presentations, documentation by receiving clear instructions.
11.	**PO11 Project Management and Finance:**	We managed the project through Engineering management principles and developed the project with very minimal resources.
12.	**PO12 Life Long Learning:**	We have ability to engage in independent and life-long learning.
13.	**PSO1 Foundation of Mathematical Concept:**	Used YOLO algorithm.
14.	**PSO3 Foundation of Software Development:**	For developing our project we used operating system Windows 10 and Python as the programming language.

DECLARATION

We hereby declare that this Project Work is titled *"Smart Alarming System Using Object Detection"* is a genuine project work carried out by us, in **B.Tech (Information Technology)** degree course of **Jawaharlal Nehru Technology University Hyderabad, Hyderabad** and has not been submitted to any other course or university for the award of my degree by us.

	Candidate Name(s)	Roll Number	Signature
1.	B. Aman Rai Saxena	16E11A1218	
2.	G. Varun Dev	16E1121217	
3.	M. Nikhil	16E11A1209	
4.	O. Chaitanya	16E11A1235	

Date:

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We avail this opportunity to express our deep sense of gratitude and hearty thanks to Sri CH. Venugopal Reddy, Secretary & Correspondent of BIET, for providing congenial atmosphere and encouragement.

We would like to thank Prof. G. Kumaraswamy Rao, Director, Former Director & O.S. of DLRL Ministry of Defence, Dr. B. Prasada Rao, I.P.S.(Retd.), Director of Training & Placements, Industry Interface, Former Principal Secretary to Govt. of AP, DGP of ACB, Commissioner of Police, Hyderabad, Dr. Merugu Lakshminarayana, Professor of ECE, BIET(SCIENTIST 'H' (Retd.) & Chair(2016 & 2017), IEEE Hyderabad), Dr. V. Rambabu, Principal for having provided all the facilities and support.

We would like to thank *HOD Mr. Madana Mohan, Professor; Academic I/C Ms.Y.Sailaja, Assistant Professor,* for their expert guidance and encouragement at various levels of our Project.

We are thankful to our *guide Ms. G. Rashmi, Assistant Professor, Dept. of Information Technology* for her sustained inspiring Guidance and cooperation throughout the process of this project. His wise counsel and suggestions were invaluable.

We are thankful to our Project Coordinator *Dr. Mohammed Aquil Mirza, Dept. of* Computer Science and Engineering for his support and cooperation throughout the process of this project.

We express our deep sense of gratitude and thanks to all the Teaching and Non-Teaching Staff of our college who stood with us during the project and helped us to make it a successful venture.

We place highest regards to our Parents, our Friends and Well-wishers who helped a lot in making the report of this project.

ABSTRACT

Object Detection is being widely used in the industry right now. It is the method of detection and shaping real-world objects. Even though there exist many detection methods, the accuracy, rapidity, and efficiency of detection are not good enough. 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 thebounding boxes using convolutional network and the class probabilities for these boxes and detects the image faster as compared to other algorithms. This algorithm performs efficient object detection while not compromising on the performance. Using this algorithm we are implementing intrusion detection system so that if an object get's detected at a secured area the user or the security personnel can be informed, we also have an alarm system that alerts the user with the voice (name of the object detected).we have used google text to speech conversion API to make the software speak the name of the object detected.

Keywords: Convolutional Neural Network, Fast-Convolutional Neural Network, Bounding Boxes, YOLO, Google text to speech.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Term	Description
SSD	Single shot Multi-box detector
YOLO	You Only Look Once
COCO	Common Object in Context
BS	Background Subtraction
VGG	Visual Geometry Group
NMS	Non Maximum Suppression
ILSVRC	ImageNet Largescale Visual Recognition competition
SGD	Stochastic Gradient descent.

1.INTRODUCTION

1.1 INTRODUCTION

In artificial vision, the neural convolution networks are distinguished in the classification of images. In this paper, a YOLO based algorithm is implemented for detection and tracking in python environment. Object detection involves detecting region of interest of object from given class of image. Different methods are –Frame differencing, Optical flow, Background subtraction. This is a method of detecting and locating an object which is in motion with the help of a camera. Detection and tracking algorithms are described by extracting the features of image and video for security applications. Object detection is a domain that has benefited immensely from the recent developments in deep learning. Recent years have seen people develop many algorithms for object detection, some of which include YOLO, SSD, Mask RCNN and Retina Net. SSD is a popular object detection algorithm that was developed in Google Inc. It is based on the VGG-16 architecture. YOLO is simple and easier to implement. A set of default boxes is made to pass over several feature maps in a convolutional manner. If an object detected is one among the object classifiers during prediction, then a score is generated. The object shape is adjusted to match the localization box. For each box, shape offsets and confidence level are predicted. During training, default boxes are matched to the ground truth boxes. The model loss is computed as a weighted sum of confidence loss and localization loss. Measure of the deviation of the predicted box from the ground truth box is localization loss. Confidence is a measure of in which manner confidence the system is that a predicted object is the actual object.

Security is a big concern in today’s world. Hiring someone to protect ourselves while we are busy is costly and many people cannot afford it, so to protect people there is police. Byusing object detection we alarm the user of any intrusion in their home, this can be done by installing a camera at their home, and running this project on that live video, if there is any activity of intrusion the users can inform police and be safe.

1.2 EXISTING SYSTEM

In Existing system there was a need of a security person to monitor the intruder activity at a places such as banks, borders and safe houses..etc, the person would monitor and inform the police or his boss about any possible intrusion or threat, this would be a costly affair as there are many cameras as borders and security places, that more number of people are required to monitor and inform the particular person.

1.3 PROPOSED SYSTEM

In this project, YOLO algorithm is implemented for detection and tracking of objects, the objects are tracked and if there is an intrusion in the live camera then the user can be informed as an alarm raises after the detection of the object to alert the user when an intrusion is noticed, to keep the cost of equipment low we are using YOLO algorithm this can run on less memory and give great results.

1.4 SYSTEM REQUIREMENTS:

1.4.1 HARDWARE REQUIREMENTS:

➢ Processor	: Intel i3 and above
➢ RAM	: 4GB and Higher
➢ HARD DISK	: 500 GB Minimum

1.4.2 SOFTWARE REQUIREMENTS:

➢ Programming Language / Platform	: Python
➢ IDE	: PYCHARM/JUPYTER
➢ Library	: OPENCV

1.5 APPLICATIONS

- Pedestrian Tracking System in Real World.
- Real Time Object Detection.
- Implemented in Security Surveillance.
- Can be used in Military

2.RELATED WORK

A method for detecting objects in images using a single deep neural network. Our approach, named YOLO, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. We present a class of efficient models called Mobile Nets for mobile and embedded vision applications. Mobile Nets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks. We introduce two simple global hyper-parameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. The YOLO object detector divides an input image into an *SxS* grid where each cell in the grid predicts only a single object. If there exist multiple, small objects in a single cell then YOLO will be unable to detect them, ultimately leading to missed object detections. Objects are detected from video sequence by making use of object detection algorithms like Gaussian Mixture Model, Haar algorithm, Histogram of oriented gradients and Local binary patterns. In this project objects are tracked based on color, motion of single and multiple objects (vehicles) are detected and counted in multiple frames. Further single algorithm may be designed for object tracking by considering shape, color, texture, object of interest, motion of object in multi direction.

3.LITERARURE SURVEY

Literature Survey:

You Only Look Once: Unified, Real-Time Object Detection, by Joseph Redmon. Their prior work is on detecting objects using a regression algorithm. To get high accuracy and good predictions they have proposed YOLO algorithm in this paper. Understanding of Object Detection Based on CNN Family and YOLO, by Juan Du. In this paper, they generally explained about the object detection families like CNN, R-CNN and compared their efficiency and introduced YOLO algorithm to increase the efficiency. Learning to Localize Objects with Structured Output Regression. This paper is about Object Localization. In this, they used the Bounding box method for localization of the objects to overcome the drawbacks of the sliding window method.

Reference 1:

Title: “YOLO based Detection and Classification of Objects in video records”
Author: Arka Prava Jana.
Description: In this objects are detected from videos by making use of object detection algorithms like Gaussian Mixture Model, Haar algorithm, Histogram of oriented gradients and Local binary patterns.

Reference 2:

Title: “Mobile Nets: Efficient Convolutional Neural Networks for Mobile Vision Applications”
Author: Andrew G. Howard
Description: Here they present a class of efficient models called Mobile Nets for mobile and embedded vision applications. Mobile Nets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks.

Reference 3:

Title: "Object detection with deep learning and OpenCV",

Author: Adrian Rosebrock

Description: This comprises of YOLO object detector which divides an input image into a *SxS* grid where each cell in the grid predicts only a single object. If there are multiple, small objects in a single cell then YOLO will be unable to detect them, ultimately leading to missed object detections.

Reference 4:

Title: "SSD: single shot multi box detector"

Author: Wei Liu and Alexander C. Berg,

Description: A method for detecting objects in images using a single deep neural network. This approach, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. This algorithm has a less accuracy compared to yolo.

Reference 5:

Title: "Object Tracking Algorithms for video surveillance applications"

Author: Mohammed Leesan, H. V. Ravish Aradhya.

Description: In this paper objects are tracked based on color, motion of single and multiple objects (vehicles) are detected and counted in multiple frames. Further single algorithm may be designed for object tracking by considering shape, color, texture, object of interest, motion of object in multi direction.

4.MOTIVATION

Some of the popular object detection algorithms are Region-based Convolutional Neural Networks (RCNN), Faster RCNN, Single Shot Detector (SSD) and You Only Look Once (YOLO). Object detection involves detecting region of interest of object from given class of image. Different methods are –Frame differencing, Optical flow, Background subtraction. Detection and tracking algorithms are described by extracting the features of image and video for security applications. The main intention of this project was to find and implement an application on security by using object detection .In this process we have compared different algorithms and found out YOLO as the suitable one as it can even run on systems of less memory(>=500) ,so we have choose this algorithm to make this accessible on normal less performance systems as majority of object detection algorithms use more memory to run.

5.OBJECTIVES

The aim of real time object detection and tracking —Due to object detection's close relationship with video analysis and image understanding, it has attracted much research attention in recent years, YOLO have shown results with considerable confidence level. Main Objective of YOLO algorithm to detect various objects in real time video sequence and track them in real time. The main objective of this project is to provide security at homes and places in less cost using the object detection technology. This will eliminate the extra cost of security members to manage the live surveillance.

6.PROBLEM STATEMENT

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box. We are further adding alarm that speaks out the name of the object detected and alerts the security personnel. This also solves the problem of continuous monitoring of system display of the captured video (live), which is a straining acti

7.DESIGN METHODOLOGY AND IMPLEMENTATION

7.1 IMPLEMENTATION

7.1.1FRAME DIFFERENCING:

Frames are captured from camera at regular intervals of time. Difference is estimated from the consecutive frames. Optical Flow This technique estimates and calculates the optical flow field with algorithm used for optical flow. A local mean algorithm is used then to enhance it. To filter noise a self-adaptive algorithm takes place. It contains a wide adaptation to the number and size of the objects and helpful in avoiding time consuming and complicated preprocessing methods.

7.1.2BACKGROUD SUBTRACTION:

Background subtraction (BS) method is a rapid method of localizing objects in motion from a video captured by a stationary camera. This forms the primary step of a multi-stage vision system. This type of process separates out background from the foreground object in sequence in images.

7.1.3OBJECT TRACKING:

It is done in video sequences like security cameras and CCTV surveillance feed; the objective is to track the path followed, speed of an object. The rate of real time detection can be increased by employing object tracking and running classification in few frames captured in a fixed interval of time. Object detection can run on a slow frame rates looking for objects to lock onto and once those objects are detected and locked, then object tracking, can run in faster frame speed.

7.1.4OBJECT ALARMING:

After the detection of object the detected frame name is fetched from the dataset and is sent to Google Text to Speech API commonly known as the gTTS API, after sending object name the voice of the object detected will be spelled out loudly so the user can know of any intruder in his place . gTTS is a very easy to use tool which converts the text on the bounding box to speech and raises an alarm. The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more. The speech can be delivered in any one of the two available audio speeds, fast or slow.

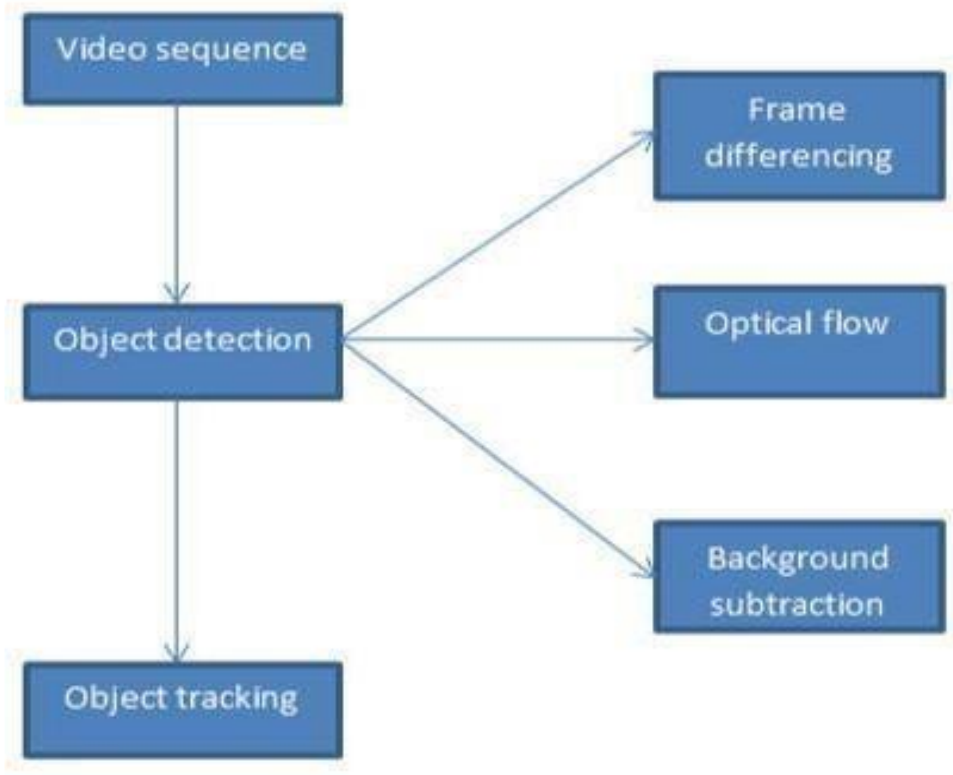
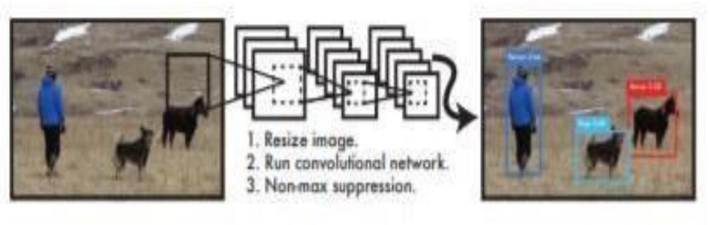


Fig 1: Architecture for real time object detection.

7.2 DESIGN METHODOLOGY:

YOLO Algorithm:

There are manyobject detection algorithms such as RCNN, SSD, Faster-RCNN, but YOLO uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes. with YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.



The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Fig2: YOLO detection system

Unified Detection:

We unify the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real time speeds while maintaining high average precision. Our system divides the input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) ∗ IOU truth prediction . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth. Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally, the confidence prediction represents the IOU between the predicted box and any ground truth box. Each grid cell also predicts C conditional class probabilities, Pr(Classification |Object). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B. At test time we multiply the conditional class probabilities and the individual box confidence predictions, which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

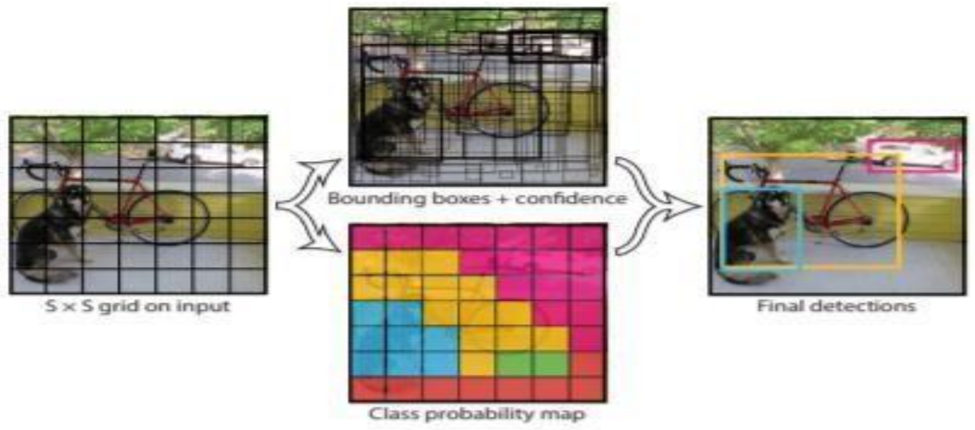
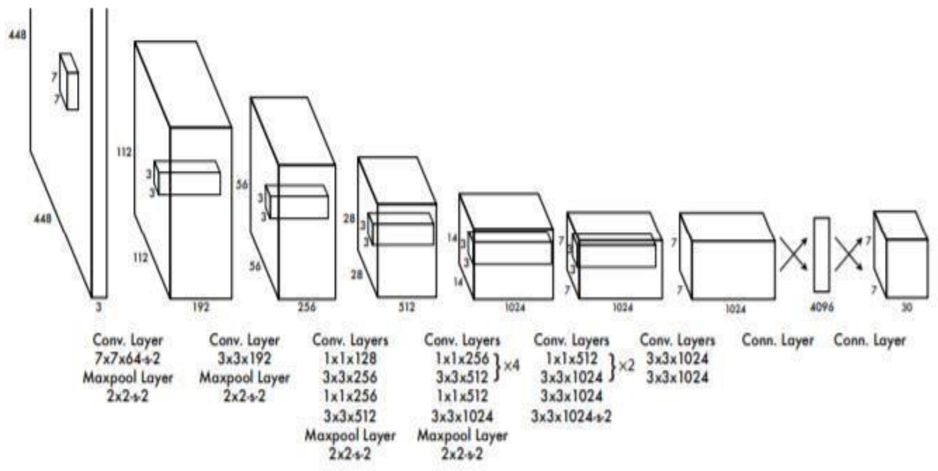


Fig3:Detection model

For evaluating YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Our final prediction is a 7 × 7 × 30 tensor.

7.2.1 ALGORITHM ARCHITECTURE:

We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset. The initial convolutional layers of the network extract features ,from the image while the fully connected layers predict the output probabilities and coordinates. Our network architecture is inspired by the Google Net model for image classification. Our network has 24 convolutional layers followed by 2 fully connected layers. we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers. The full network is shown in Figure 3. We also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO



448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

Fig4: YOLO Architecture

The final output of our network is the 7 × 7 × 30 tensor of predictions

7.2.2 SYSTEM ARCHITECTURE

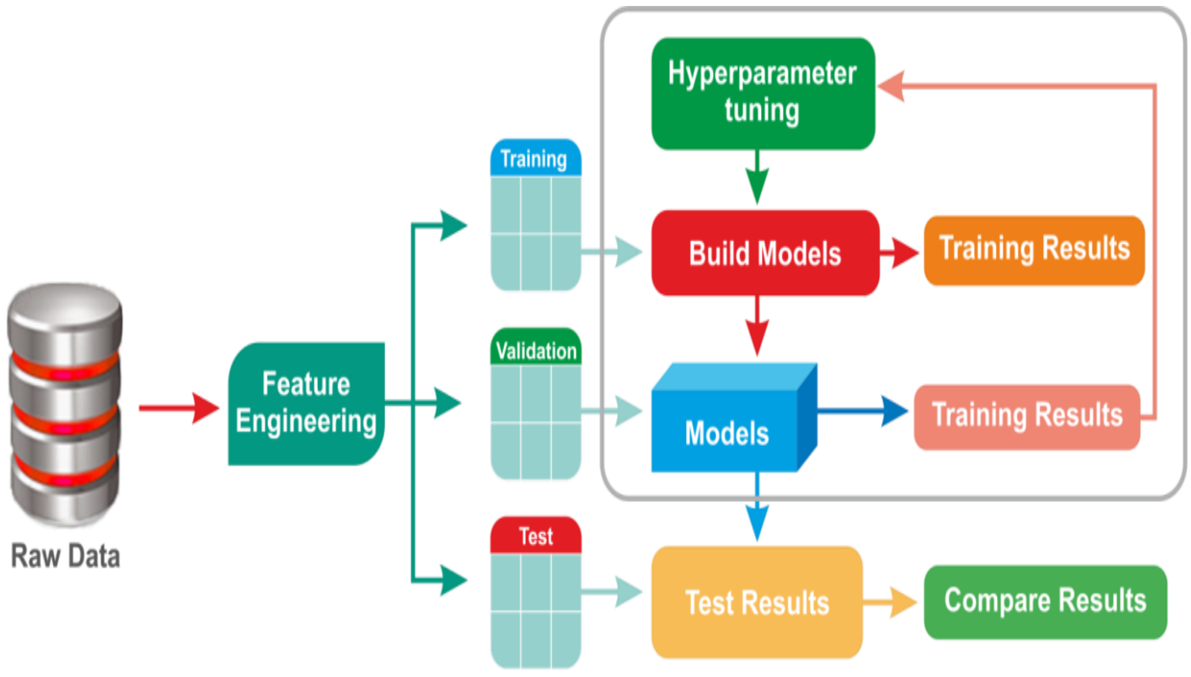


Fig5: System architecture of the project.

The above system architecture shows the actual skeleton of the project, here the model is trained on the dataset and then the data is given to the trained model for further detection of the object based on the names in the dataset.

7.3UML DIAGRAMS

Unified Modeling Language (UML) is a modelling language. The main purpose of UML is to visualize the way a system has been designed. It is a visual language to sketch the behavior and structure of the system. This was adopted by Object Management Group (OMG) as a standard in 1997.

7.3.1Use Case Diagram:

- The purpose of use case diagram is to capture the dynamic aspect of a system. This is used to gather the requirements of a system including internal and externalinfluences.
- The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.
- The UML is a very important part of developing objects Oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

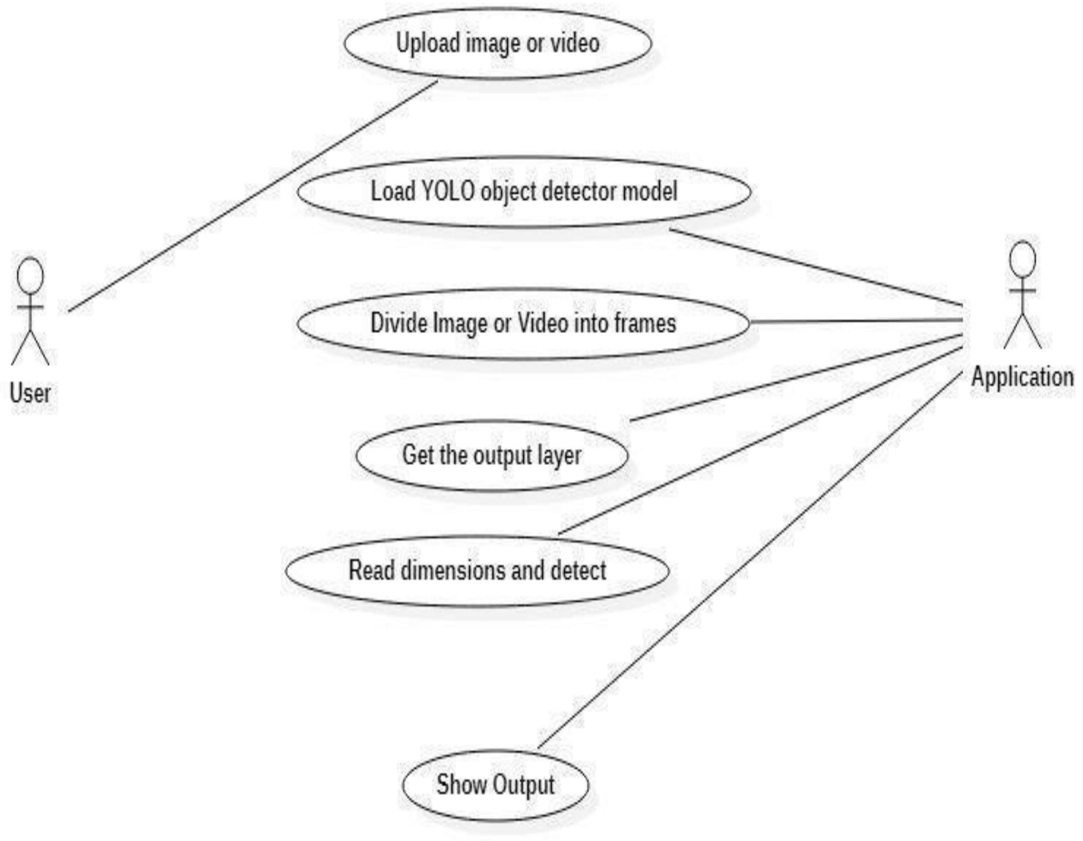

Upload image or video
Load YOLO object detector model
Divide Image or Video into frames
Get the output layer
Read dimensions and detect
Show Output
User
Application

Fig 6: Use-Case diagram for real time object detection and tracking.

7.3.2Sequence Diagram:

- A sequence diagram details the interaction between objects in a sequential order i.e. the order in which these interactions take place.
- This diagrams sometimes known as event diagrams or event scenarios. This helps in understanding how the objects and component interacts to execute the process.
- This has two dimensions which represents time (Vertical) and different objects (Horizontal).

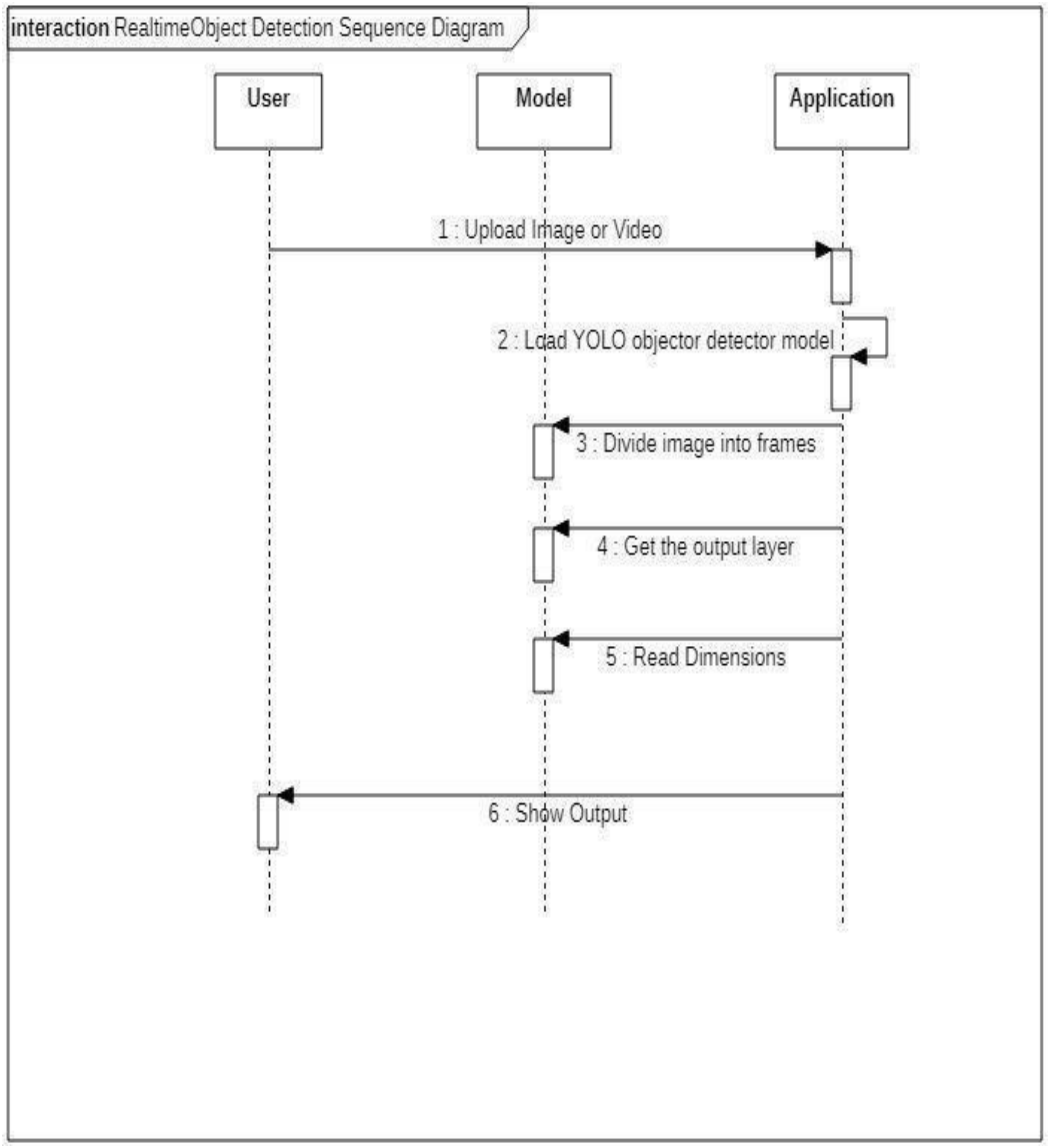
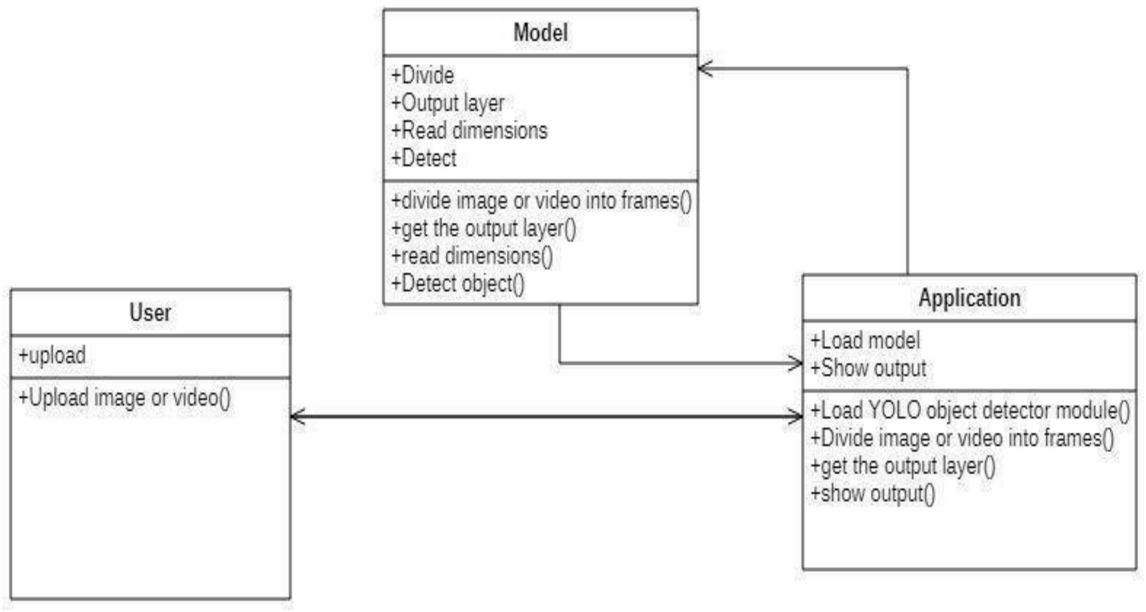

interaction RealtimeObject Detection Sequence Diagram
User
Model
Application
1 : Upload Image or Video
2 : Load YOLO objector detector model
3 : Divide image into frames
4 : Get the output layer
5 : Read Dimensions
6 : Show Output

Fig 7: Sequence diagram for real time object detection and tracking.

7.3.3 Class Diagram:

The class diagram describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among the classes.

It explains which class contains information and also describes responsibilities of the system. This is also known as structural diagram.



.

Fig 8: Class diagram for real time object detection and tracking.

7.3.4Activity Diagram:

It is behavioral diagram which reveals the behaviors of a system.it sketches the control flow from initiation point to a finish point showing the several decision paths that exist while the activity is being executed. This doesn’t show any message flow from one activity to another, it is sometimes treated as the flowchart. Despite they look like a flowchart, they are not. In the Unified Modeling Language, Activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.

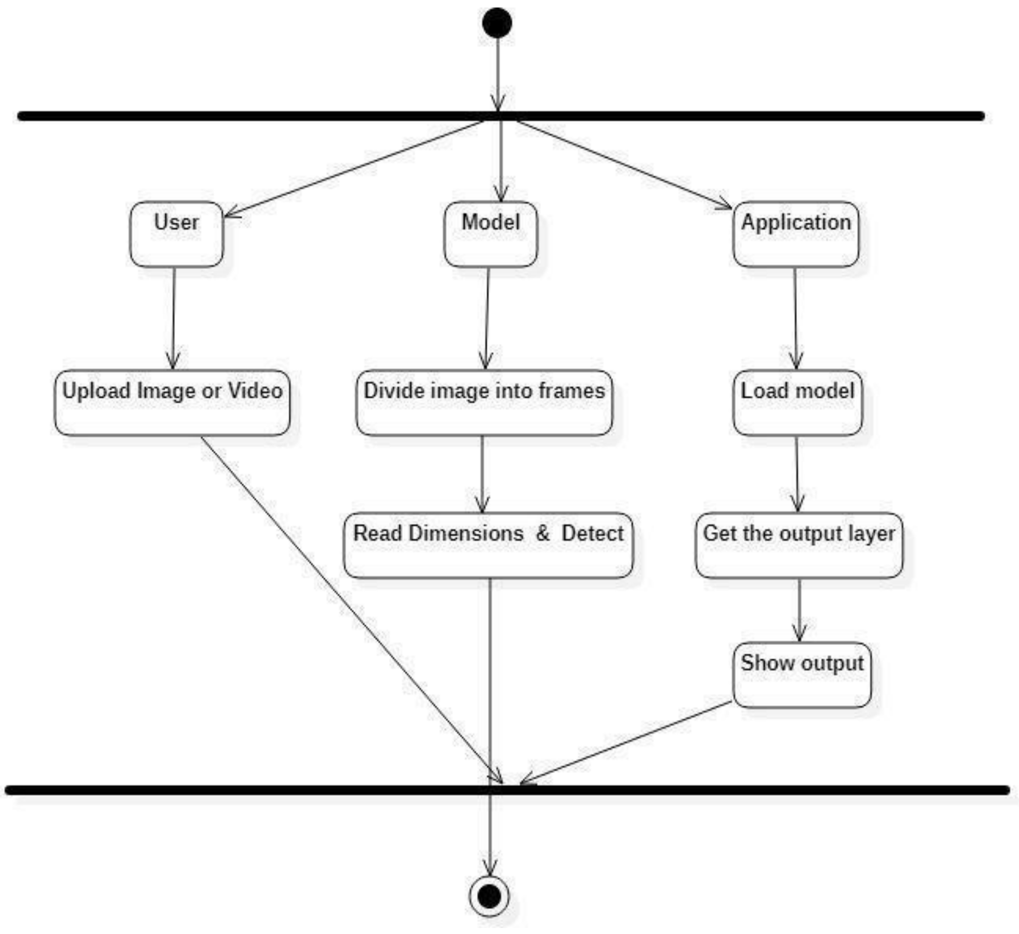


Fig9: Activity diagram for real time object detection and tracking

7.4 a)Training:

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the Google Net models in Caffe's Model Zoo [24]. We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [28]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224 × 224 to 448 × 448. Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1. We use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on. To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λ co-ordination and λno.of.obj to accomplish this. We set λ co-ordination = 5 and λ no.obj = .5.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this, we predict the square root of the bounding box width and height instead of the width and height directly. YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall. During training we optimize the following, multi-part 781 loss function:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

where 1obj ‘i’ denotes if object appears in cell ‘i’ and 1 obj ‘ij’ denotes that the ‘jth’ bounding box predictor in cell ‘i’ is “responsible” for that prediction. Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditionalclass probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is “responsible” for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell). We train the network for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012. When testing on 2012 we also include the VOC 2007 test data for training. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005. Our learning rate schedule is as follows: For the first epochs we slowly raise the learning rate from 10−3 to 10−2. If we start at a high learning rate our model often diverges due to unstable gradients. We continue training with 10−2 for 75 epochs, then 10−3 for 30 epochs, and finally 10−4 for 30 epochs. To avoid over fitting, we use dropout and extensive data augmentation. A dropout layer with rate = .5 after the first connected layer prevents co-adaptation between layers [18]. For data augmentation we introduce random scaling and translations of up to 20% of the original image size. We also randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

Inference:

Just like in training, predicting detections for a test image only requires one network evaluation. On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods. The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2- 3% in map.2.4.

Limitations of YOLO:

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds. Since our model learns to predict bounding boxes fromdata, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down sampling layers from the input image. Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

FRONT-END:

we have a GUI graphical user interface for the execution of the project, so that the execution of the project becomes simple.

GUI:

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

GUI objects include icons, cursors, and buttons. These graphical elements are sometimes enhanced with sounds, or visual effects like transparency and drop shadows.

A GUI is considered to be more user-friendly than a text-based command-line interface, such as MS-DOS, or the shell of Unix-like operating systems.

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

Tkinter - Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.

wxPython − This is an open-source Python interface windows.

JPython – JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries.

In our project we have made use of Tkinter library.

TKINTER LIBRARY:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Geometry Management:

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

The pack() Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.

The grid() Method − This geometry manager organizes widgets in a table-like structure in the parent widget.

The place() Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

ADDING BUTTONS:

Button: To add a button in your application, this widget is used.

The general syntax is:

w=Button(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

Active background: to set the background colour, when button is under the cursor.

Active foreground: to set the foreground colour, when button is under the cursor.

BG: to set he normal background colour.

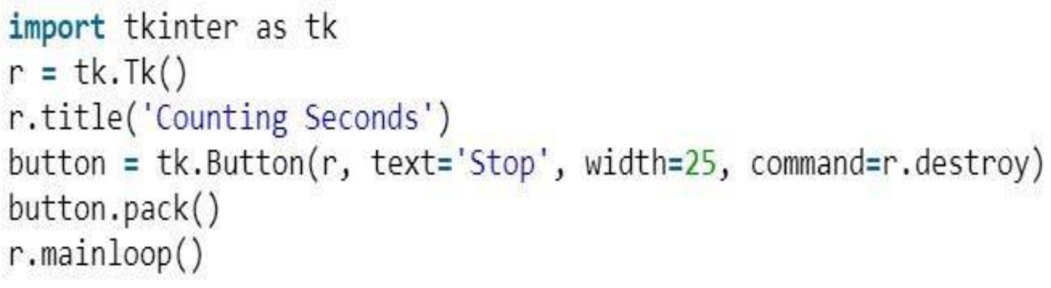
command: to call a function.

font: to set the font on the button label.

image: to set the image on the button.

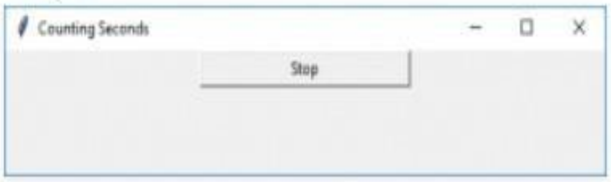
width: to set the width of the button.

height: to set the height of the button.



```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=25, command=r.destroy)
button.pack()
r.mainloop()
```

Output:



Sample code for the button named "stop"

2. **Canvas:** It is used to draw pictures and other complex layout like graphics, text, widgets.

The general syntax is:

w = Canvas(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

bd: to set the border width in pixels**.**

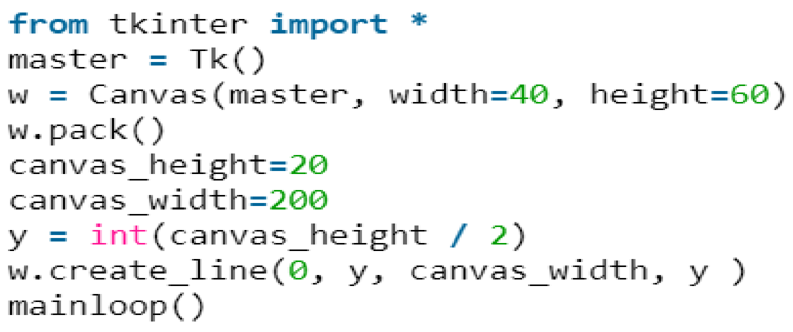
BG: to set the normal background color.

cursor: to set the cursor used in the canvas.

Highlight color: to set the color shown in the focus highlight**.**

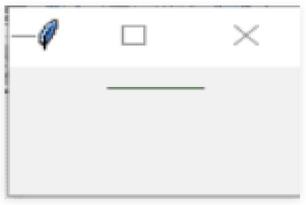
width: to set the width of the widget.

height: to set the height of the widget.



```
from tkinter import *
master = Tk()
w = Canvas(master, width=40, height=60)
w.pack()
canvas_height=20
canvas_width=200
y = int(canvas_height / 2)
w.create_line(0, y, canvas_width, y )
mainloop()
```

Output:



The above image shows the output of canvas

Label: It refers to the display box where you can put any text or image which can be updated any time as per the code.

The general syntax is:

w=Label(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.
BG: to set the normal background color.

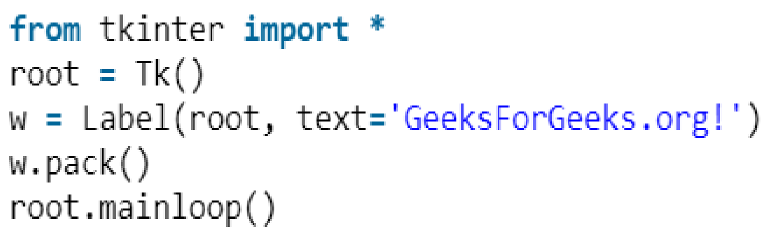
command: to call a function.

font: to set the font on the button label.

image: to set the image on the button.

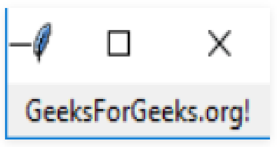
width: to set the width of the button.

Height: to set the height of the button.



```
from tkinter import *
root = Tk()
w = Label(root, text='GeeksForGeeks.org!')
w.pack()
root.mainloop()
```

Output:



```
GeeksForGeeks.org!
```

The above image shoes the output of a label implemented

Convert Text to Speech in Python:

There are several APIs available to convert text to speech in python. One of such APIs is the Google Text to Speech API commonly known as the gTTS API. gTTS is a very easy to use tool which converts the text entered, into audio which can be saved as a mp3 file.

The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more. The speech can be delivered in any one of the two available audio speeds, fast or slow. However, as of the latest update, it is not possible to change the voice of the generated audio

gTTS (*Google Text-to-Speech*):

gTTS is a Python library and CLI tool to interface with Google Translate's text-to-speech API.

Features:

Customizable speech-specific sentence tokenizer that allows for unlimited lengths of text to be read, all while keeping proper intonation, abbreviations, decimals and more;

Customizable text pre-processors which can, for example, provide pronunciation corrections;

Automatic retrieval of supported languages.

Parameters

text (string) – The text to be read.

tld (string) – Top-level domain for the Google Translate host, This is useful when google.com might be blocked within a network but a local or different Google host (e.g. google.cn) is not. Default is com.

lang (string, optional) –

The language (IETF language tag) to read the text in.

slow (bool, optional) –

Reads text more slowly. Defaults to False.

lang_check (bool, optional) –

Strictly enforce an existing lang, to catch a language error early. If set to True, a Value error is raised if lang doesn't exist. Setting lang check to False skips Web requests (to validate language) and therefore speeds up instanciation. Default is True.
pre_processor_funcs (list) –

A list of zero or more functions that are called to transform (pre-process) text before tokenizing. Those functions must take a string and return a string

7.4 b) Performance

To explain about performance, we have experimental results on the performance of the YOLO algorithm on PASCAL VOC2007 dataset and comparison of faster R-CNN and YOLO algorithms based on performance. First we compare YOLO with other real-time detection systems on PASCAL VOC 2007. To understand the differences between YOLO and R-CNN variants we explore the errors on VOC 2007 made by YOLO and Fast R-CNN, one of the highest performing versions of R-CNN. Based on the different error profiles we show that YOLO can be used to rescore Fast R-CNN detections and reduce the errors from background false positives, giving a significant performance boost. We also present VOC 2012 results and compare mAP to current state-of-the-art methods. Finally, we show that YOLO generalizes to new domains better than other detectors on two artwork datasets.

Comparison to Other Detection Systems:

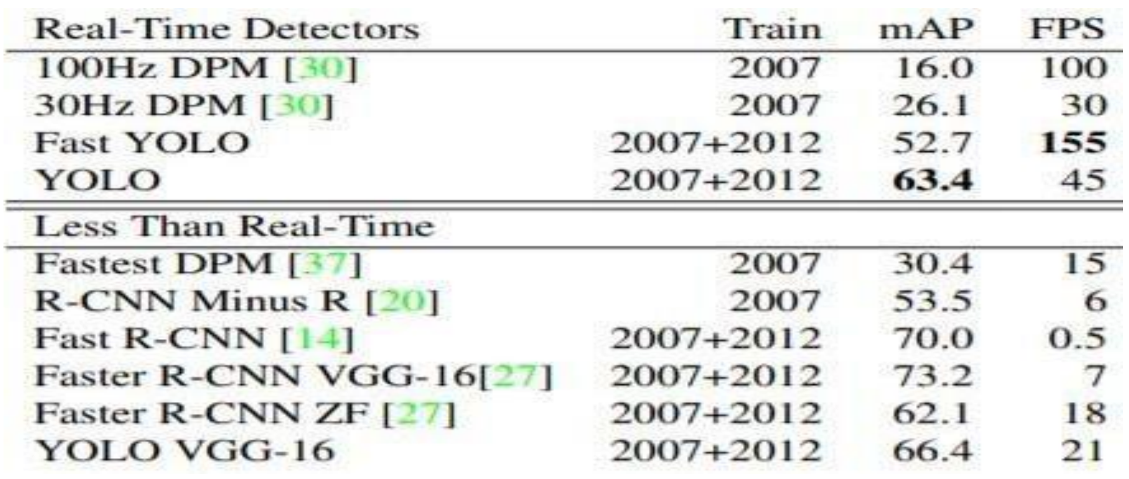
Object detection is a core problem in computer vision. Detection pipelines generally start by extracting a set of robust features from input images. Then, classifiers or localizers are used to identify objects in the feature space. These classifiers or localizers are run either in sliding window fashion over the whole image or on some subset of regions in the image. We compare the YOLO detection system to several top detection frameworks, highlighting key similarities and differences. Deformable parts models. Deformable parts models (DPM) use a sliding window approach to object detection. DPM uses a disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions, etc. Our system replaces all of these disparate parts with a single convolutional neural network. The network performs feature extraction, bounding box prediction, nonmaximal suppression, and contextual reasoning all concurrently. Instead of static features, the network trains the features in-line and optimizes them for the detection task. Our unified architecture leads to a faster, more accurate model than DPM. R-CNN. R-CNN and its variants use region proposals instead of sliding windows to find objects in images. Selective 782 Search generates potential bounding boxes, a convolutional network extracts features, an SVM scores the boxes, a linear model

adjusts the bounding boxes, and non-max suppression eliminates duplicate detections. Each stage of this complex pipeline must be precisely tuned independently and the resulting system is very slow, taking more than 40 seconds per image at test time. YOLO shares some similarities with R-CNN. Each grid cell proposes potential bounding boxes and scores those boxes using convolutional features. However, our system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object. Our system also proposes far fewer bounding boxes, only 98 per image compared to about 2000 from Selective Search. Finally, our system combines these individual components into a single, jointly optimized model. Other Fast Detectors Fast and Faster R-CNN focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search. While they offer speed and accuracy improvements over R-CNN, both still fall short of real-time performance. Many research efforts focus on speeding up the DPM pipeline. They speed up HOG computation, use cascades, and push computation to GPUs. However, only 30Hz DPM actually runs in real-time. Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design. Detectors for single classes like faces or people can be highly optimized since they have to deal with much less variation . YOLO is a general purpose detector that learns to detect a variety of objects simultaneously. Deep Multi Box. Unlike R-CNN, train a convolutional neural network to predict regions of interest instead of using Selective Search. Multi Box can also perform single object detection by replacing the confidence prediction with a single class prediction. However, Multi Box cannot perform general object detection and is still just a piece in a larger detection pipeline, requiring further image patch classification. Both YOLO and Multi Box use a convolutional network to predict bounding boxes in an image but YOLO is a complete detection system. train a convolutional neural network to perform localization and adapt that localizer to perform detection.

Over Feat efficiently performs sliding window detection but it is still a disjoint system. Over Feat optimizes for localization, not detection performance. Like DPM, the localizer only sees local information when making a prediction. Over Feat cannot reason about global context and thus requires significant post-processing to produce coherent detections. Multi Grasp. Our grid approach to bounding box prediction is based on the Multi Grasp system for regression to grasps. However, grasp detection is a much simpler task than object detection. Multi Grasp only needs to predict a single graspable region for an image containing one object. It doesn't have to estimate the size, location, or boundaries of the object or predict it's class, only find a region suitable for grasping. YOLO predicts both bounding boxes and class probabilities for multiple objects of multiple classes in an image.

7.4.1Comparison to Other Real-Time Systems

We compare YOLO to their GPU implementation of DPM which runs either at 30Hz or 100Hz. While the other efforts don't reach the real-time milestone, we also compare their relative map and speed to examine the accuracy-performance tradeoffs available in object detection systems. Fast YOLO is the fastest object detection method on PASCAL; as far as we know, it is the fastest extant object detector. With 52.7% map, it is more than twice as accurate as prior work on real-time detection. YOLO pushes map to 63.4% while still maintaining real-time performance. We also train YOLO using VGG-16. This model is more accurate but also significantly slower than YOLO. It is useful for comparison to other detection systems that rely on VGG-16 but since it is slower than real-time the rest of the paper focuses on our faster models. Fastest DPM effectively speeds up DPM without sacrificing much map but it still misses real-time performance by a factor of 2. It also is limited by DPM's relatively low accuracy on detection compared to neural network approaches. R-CNN minus R replaces Selective Search with static bounding box proposals .



Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Fig10: comparison of performance and speed of detection system's

While it is much faster than R-CNN, it still falls short of real-time and takes a significant accuracy hit from not having good proposals. Fast R-CNN speeds up the classification stage of R-CNN but it still relies on selective search which can take around 2 seconds per image to generate bounding box proposals. Thus it has high MAP but at 0.5 fps it is still far from real time. The recent Faster R-CNN replaces selective search with a neural network to propose bounding boxes, In our tests, their most accurate model achieves 7 fps while a smaller, less accurate one runs at 18 fps. The VGG-16 version of Faster R-CNN is 10 map higher but is also 6 times slower than YOLO. The Faster R-CNN is only 2.5 times slower than YOLO but is also less accurate.

VOC 2007 Error Analysis:

To further examine the differences between YOLO and state-of-the-art detectors, we look at a detailed breakdown of results on VOC 2007. We compare YOLO to Fast RCNN since Fast R-CNN is one of the highest performing detectors on PASCAL and it's detections are publicly available. For each category at test time we look at the top N predictions for that category. Each prediction is either correct or it is classified based on the type of error :

- Correct: correct class and IOU $> .5$
- Localization: correct class, $.1 < \text{IOU} < .5$
- Similar: class is similar, IOU $> .1$

YOLO struggles to localize objects correctly. Localization errors account for more of YOLO's errors than all other sources combined. Fast R-CNN makes much fewer localization errors but far more background errors. 13.6% of it's top detections are false positives that don't contain any objects. Fast R-CNN is almost 3x more likely to predict background detections than YOLO.

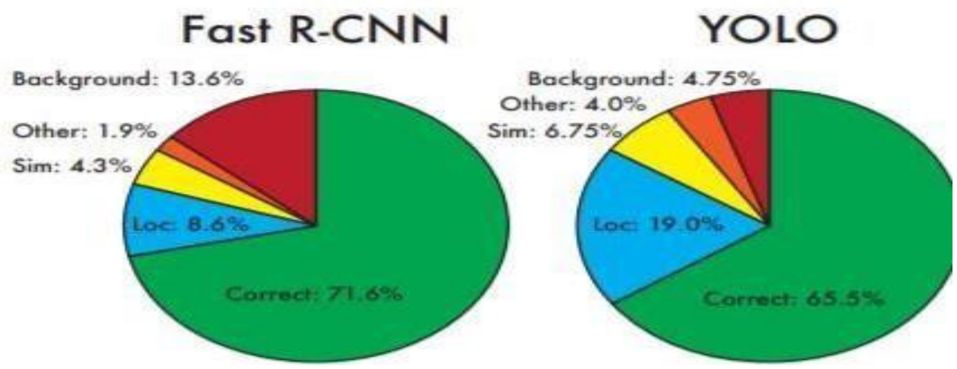
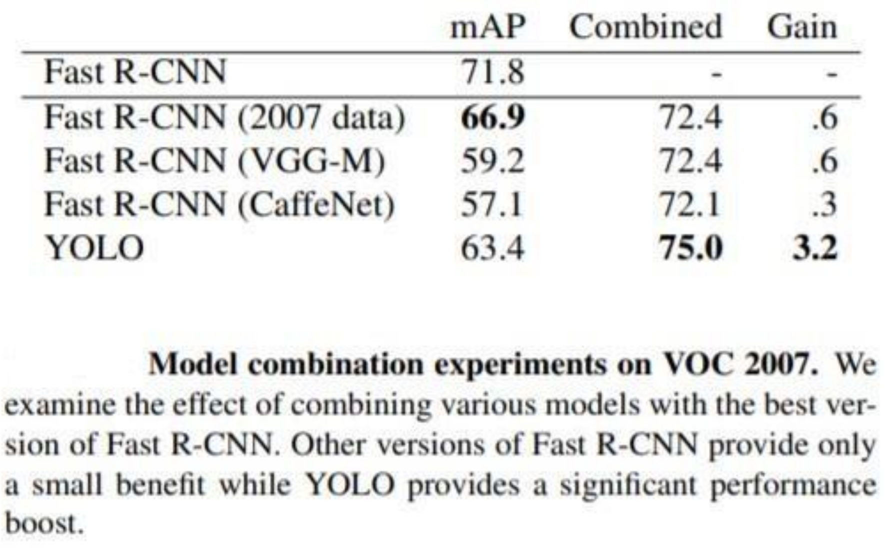


Fig11: Error Analysis

Combining Fast R-CNN and YOLO:

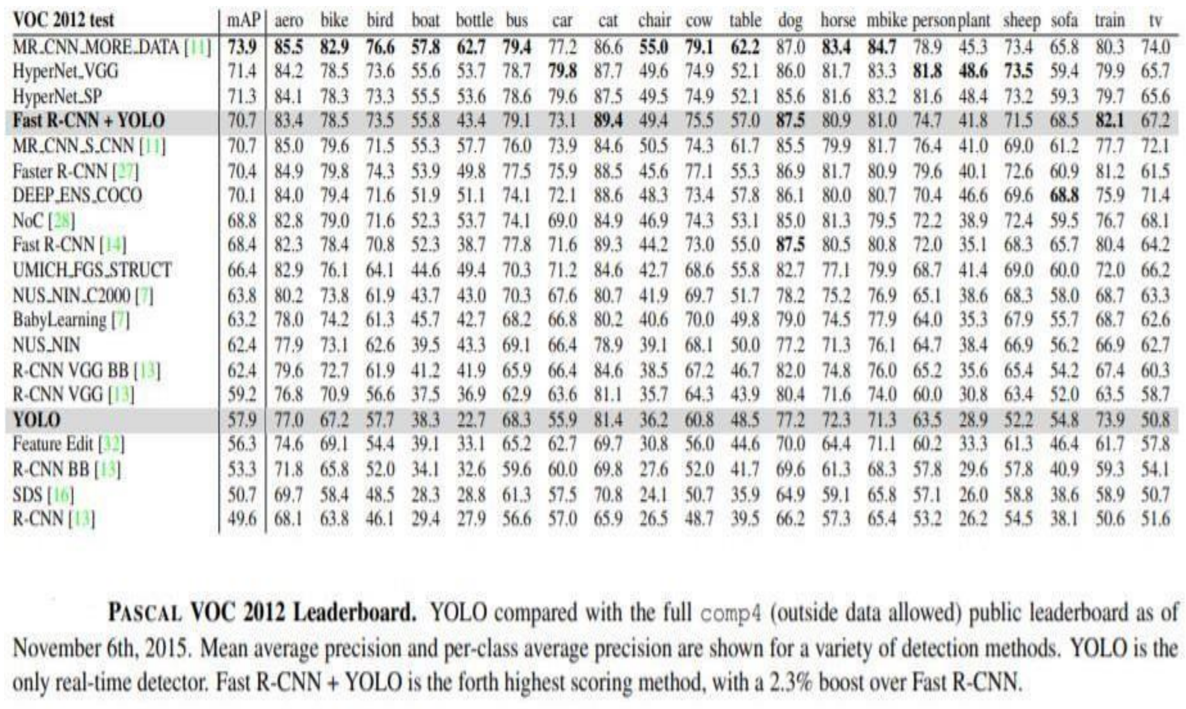
YOLO makes far fewer background mistakes than Fast R-CNN. By using YOLO to eliminate background detections from Fast R-CNN we get a significant boost in performance. For every bounding box that R-CNN predicts we check to see if YOLO predicts a similar box. If it does, we give that prediction a boost based on the probability predicted by YOLO and the overlap between the two boxes. The best Fast R-CNN model achieves a mAP of 71.8% on the VOC 2007 test set. When combined with YOLO, its



	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

Fig12: Model combination experiments on VOC 2007



VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [27]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [28]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

PASCAL VOC 2012 Leaderboard. YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

Fig13: Pascal VOC 2012 Leaderboard

S map increases by 3.2% to 75.0%. We also tried combining the top Fast R-CNN model with several other versions of Fast R-CNN. Those ensembles produced small increases in map between .3 and .6%, see Table 2 for details. The boost from YOLO is not simply a byproduct of model ensembling since there is little benefit from combining different versions of Fast R-CNN. Rather, it is precisely because YOLO makes different kinds of mistakes at test time that it is so effective at boosting Fast R-CNN's performance. Unfortunately, this combination doesn't benefit from the speed of YOLO since we run each model seperately and then combine the results. However, since YOLO is so fast it doesn't add any significant computational time compared to Fast R-CNN.

VOC 2012 Results:

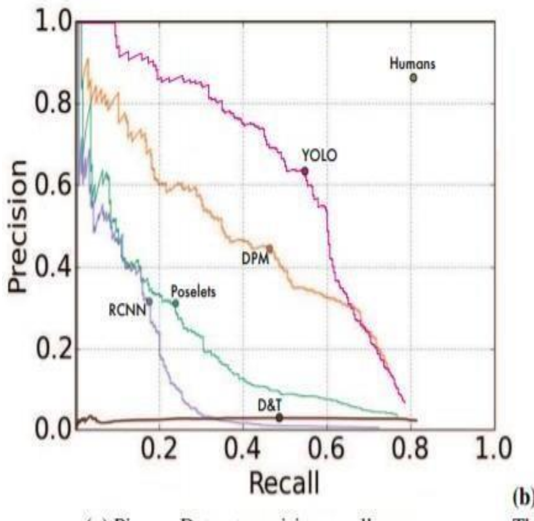
On the VOC 2012 test set, YOLO scores 57.9% map. This is lower than the current state of the art, closer to the original R-CNN using VGG-16, see Table 3. Our system struggles with small objects compared to its closest competitors. On categories like bottle, sheep, and tv/monitor YOLO scores 8-10% lower than R-CNN or Feature Edit. However, on other categories like cat and train YOLO achieves higher performance. Our combined Fast R-CNN + YOLO model is one of the highest performing detection methods. Fast R-CNN gets a 2.3% improvement from the combination with YOLO, boosting it 5 spots up on the public leaderboard.

Generalizability:

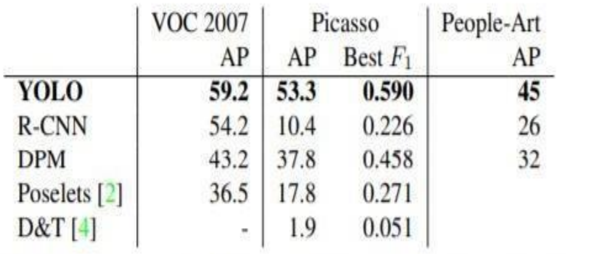
Person Detection in Artwork Academic datasets for object detection draw the training and testing data from the same distribution. In real-world applications it is hard to predict all possible use cases and the test data can diverge from what the system has seen before. We compare YOLO to other detection systems on the Picasso Dataset and the People-Art Dataset two datasets for testing person detection on artwork. Figure 5 shows comparative performance between YOLO and other detection methods. For reference, we give VOC 2007 detection AP on person where all models are trained only on VOC 2007 data. On Picasso models are trained on VOC 2012 while on People-Art they are trained on VOC 2010. R-CNN has high AP on VOC 2007. However, R-CNN drops off considerably when applied to art work. R-CNN uses Selective Search for bounding box proposals which is tuned for natural images. The classifier step in R-CNN only sees small regions and needs good proposals. DPM maintains its AP well when applied to artwork. Prior work theorizes that DPM performs well because it has strong spatial models of the shape and layout of objects. Though DPM doesn't degrade as much as R-CNN, it starts from a lower AP. YOLO has good performance on VOC 2007 and its AP degrades less than other methods when applied to artwork. Like DPM, YOLO models the size and shape of objects, as well as relationships between objects and where objects commonly appear. Artwork and natural images are very different on a pixel level but they are similar in terms of the size and shape of objects, thus YOLO can still predict good bounding boxes and detections.

Real-Time Detection In The Wild:

YOLO is a fast, accurate object detector, making it ideal for computer vision applications. We connect YOLO to a webcam and verify that it maintains real-time performance, including the time to fetch images from the camera and display the detections. The resulting system is interactive and engaging. While YOLO processes images individually, when attached to a webcam it functions like a tracking system, detecing objects as they move around and change in appearance.



(a) Picasso Dataset precision-recall curves.



	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Generalization results on Picasso and People-Art datasets.

Fig14: generalization results on datasets

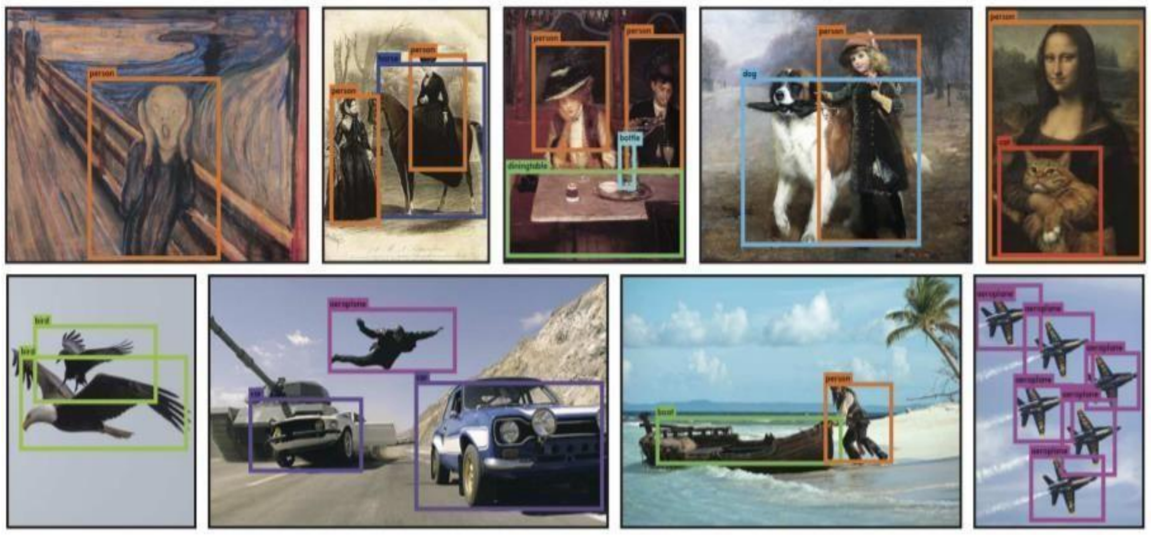


Fig15: Sample images of object detection

8.TESTING

8.1 TEST CASES:

Tested	Test name	Inputs	Expected output	Actual Output	status
1	giving test image or video	image or video	input taken	successfully taken	success

Table 1: Test Case 1

Tested	Test name	Inputs	Expected output	Actual Output	status
2	loading model	invoking YOLO model	model loaded successfully	Model Loaded	success

Table 2: Test Case 2

Tested	**Test name**	**Inputs**	**Expected output**	**Actual Output**	**status**
3	object detection	frames divided by model	object detected	successfully detected	success

Table 3: Test Case 3

Tested	**Test name**	**Inputs**	**Expected output**	**Actual Output**	**status**
4	displaying output	detected object in image	image with object name in box	successfully displayed	success

Table 4: Test Case 4

9.RESULTS OF THE PROJECT

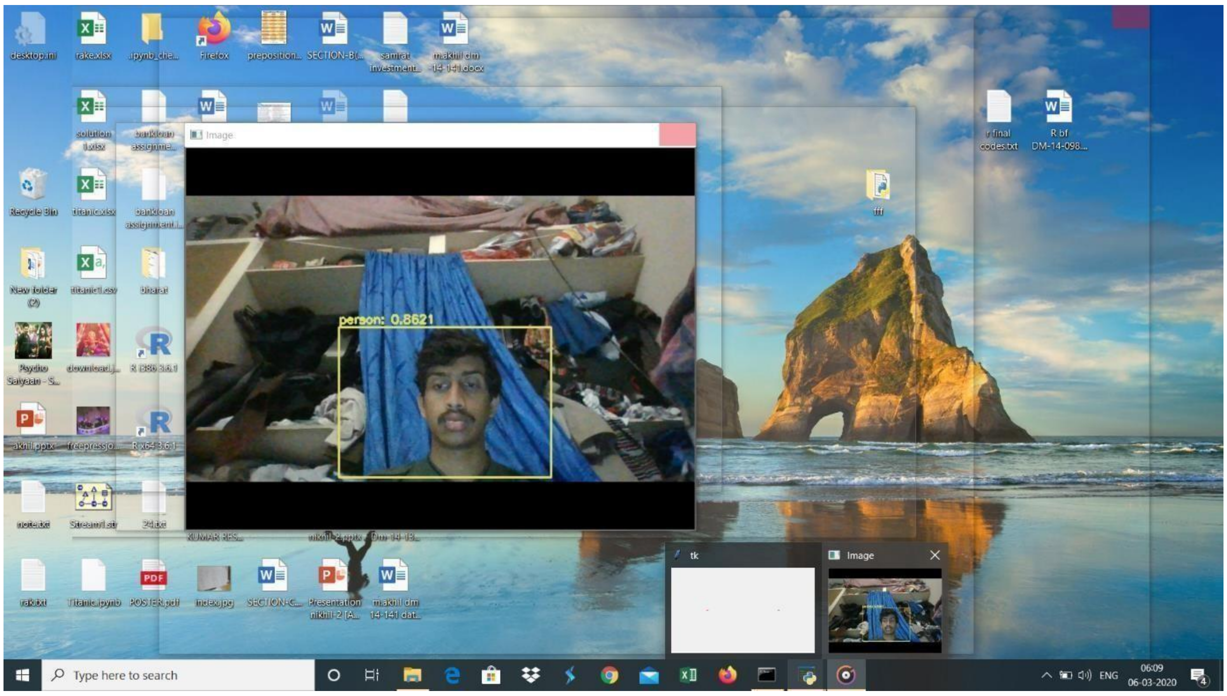


Fig 16: Detection of the object**.**

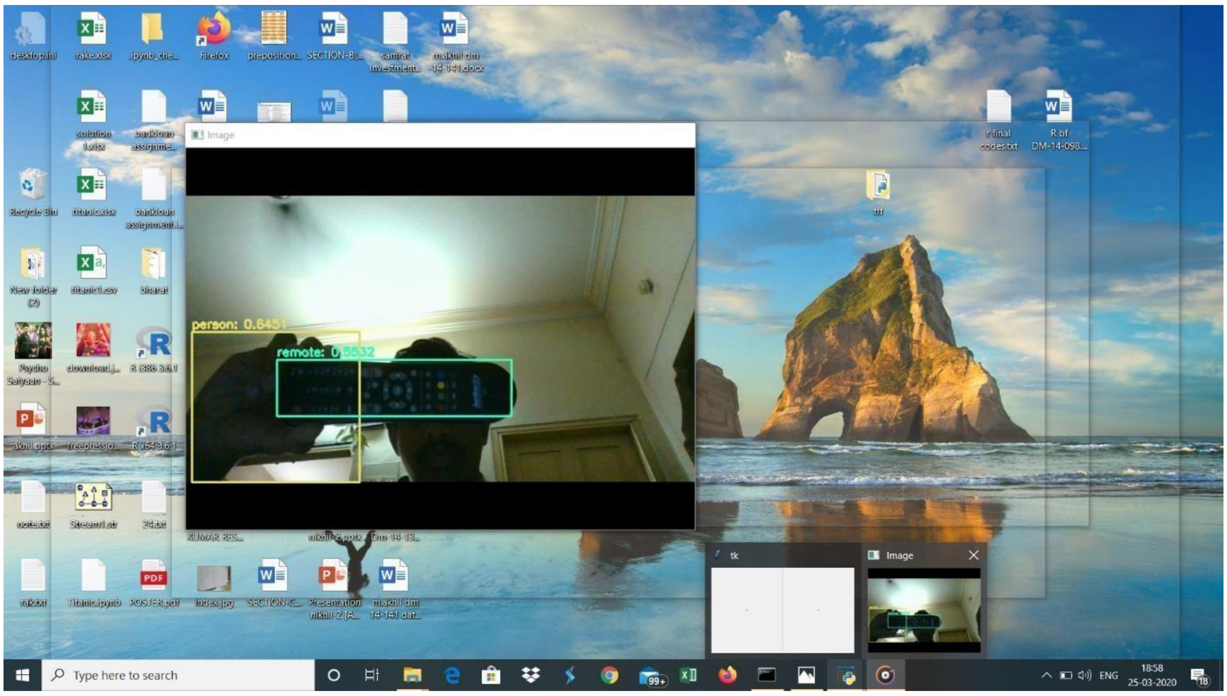


Fig 17: Detection of the object person and remote.

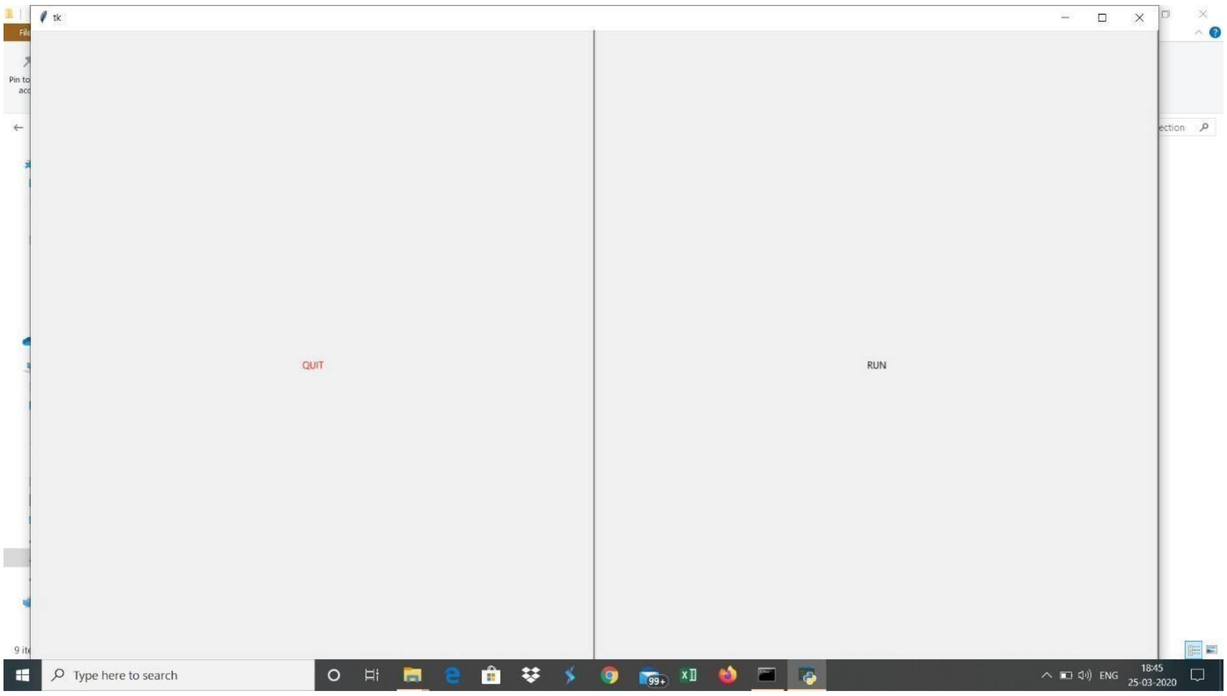
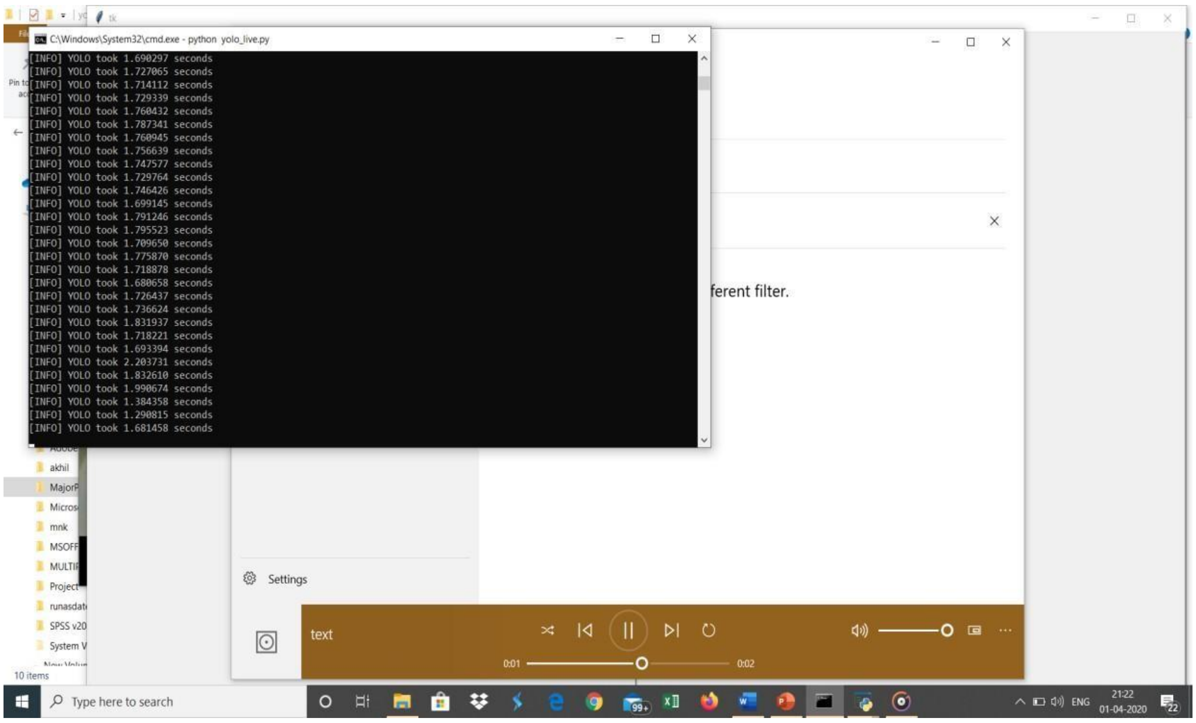


Fig 18: Implementation of GUI using Tkinter library in python.


C:\Windows\System32\cmd.exe - python yolo_live.py

```
[INFO] YOLO took 1.690297 seconds
[INFO] YOLO took 1.727065 seconds
[INFO] YOLO took 1.714112 seconds
[INFO] YOLO took 1.729339 seconds
[INFO] YOLO took 1.760432 seconds
[INFO] YOLO took 1.787341 seconds
[INFO] YOLO took 1.760945 seconds
[INFO] YOLO took 1.756639 seconds
[INFO] YOLO took 1.747577 seconds
[INFO] YOLO took 1.729764 seconds
[INFO] YOLO took 1.746426 seconds
[INFO] YOLO took 1.699145 seconds
[INFO] YOLO took 1.791246 seconds
[INFO] YOLO took 1.795523 seconds
[INFO] YOLO took 1.709650 seconds
[INFO] YOLO took 1.775870 seconds
[INFO] YOLO took 1.718878 seconds
[INFO] YOLO took 1.680658 seconds
[INFO] YOLO took 1.726437 seconds
[INFO] YOLO took 1.736624 seconds
[INFO] YOLO took 1.831937 seconds
[INFO] YOLO took 1.718221 seconds
[INFO] YOLO took 1.693394 seconds
[INFO] YOLO took 2.203731 seconds
[INFO] YOLO took 1.832610 seconds
[INFO] YOLO took 1.990674 seconds
[INFO] YOLO took 1.384358 seconds
[INFO] YOLO took 1.290815 seconds
[INFO] YOLO took 1.681458 seconds
```

Fig 19: time to detect objects (value in command prompt)

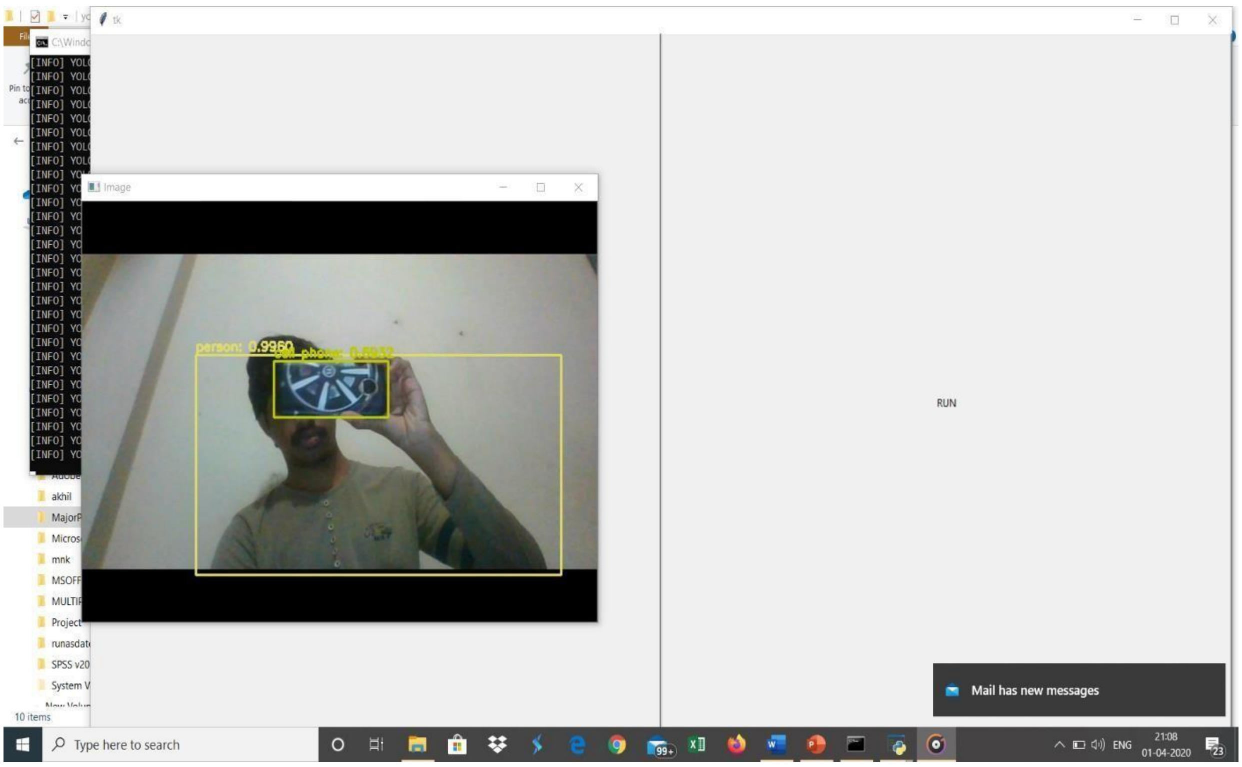
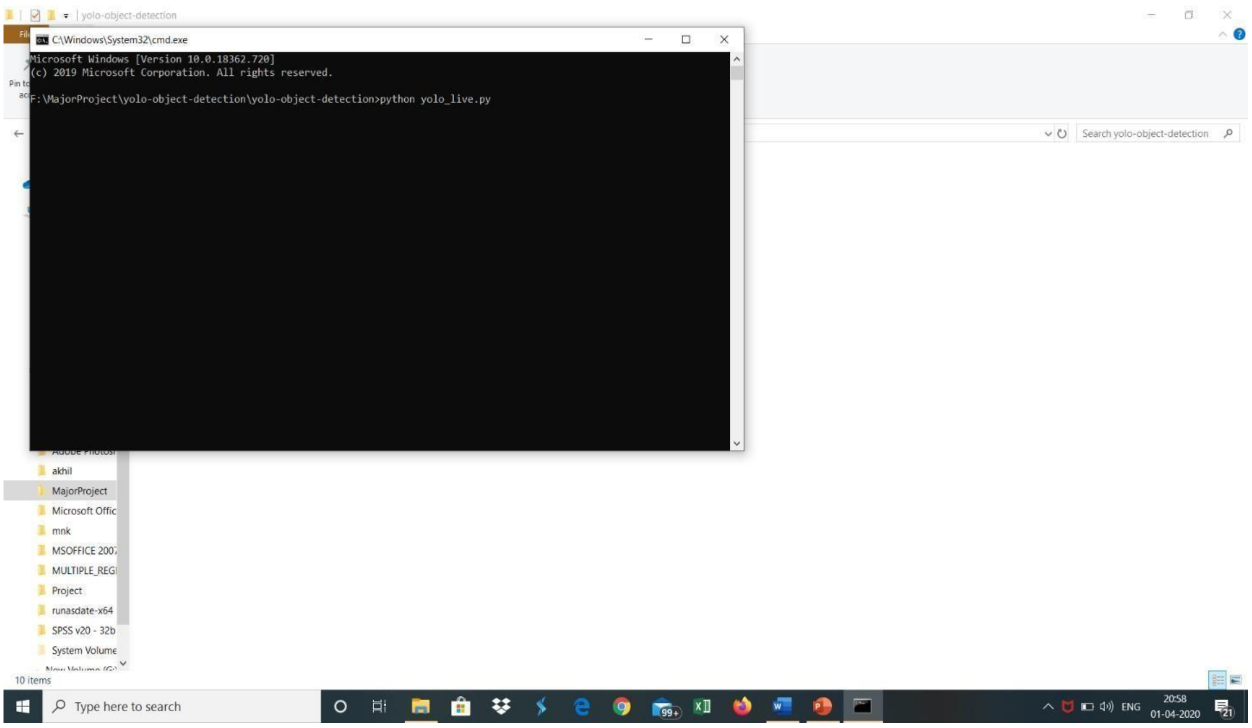


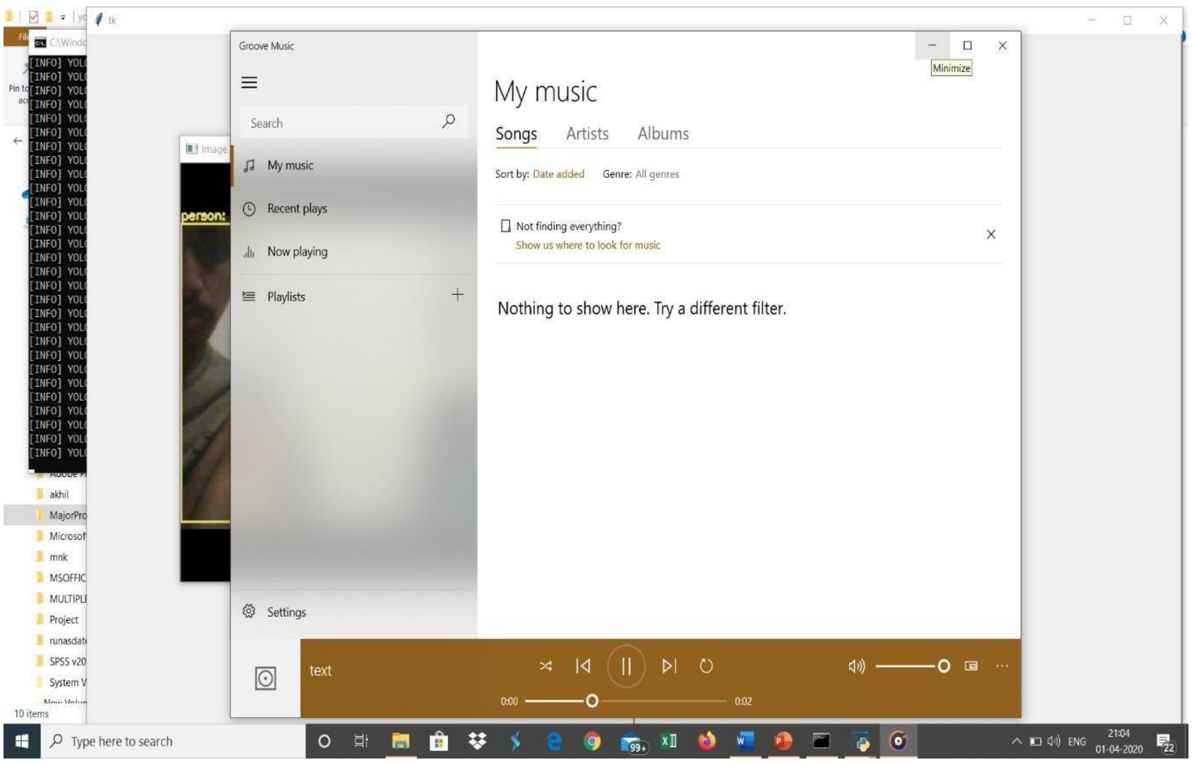
Fig20: Figure showing detection of phone and person



```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

F:\MajorProject\yolo-object-detection\yolo-object-detection>python yolo_live.py
```

Fig21: Figure is showing the execution of the project through command prompt



```
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
[INFO] YOL
```

Nothing to show here. Try a different filter.

Fig21:figure of gTTS alarming the user

The above image is of the google text to speech alarming the user by taking the name of The object detected.

10.CONCLUSION

YOLO is a fast and accurate object detector algorithm. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection. In this project we have successfully used YOLO algorithm of object detection to help security users identify intruders using an alarm(name of the object).This will help people from theft and robbery by helping them to know about any intruder activity happening at their home so that they can inform security personnel, this can be used in military also as it can detect objects and alarm the army of any intruders at the borders, another use of this application can be made on bank's CCTV cameras (as this can detect the harmful objects(knives..etc)) and the police can be informed.

11.REFERENCES

[1] Wei Liu and Alexander C. Berg, “SSD: Single Shot Multi Box Detector”, Google Inc., Dec 2016.

[2] Andrew G. Howard, and Hartwig Adam, “Mobile Nets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, Google Inc., 17 Apr 2017.

[3] Justin Lai, Sydney Maples, “Ammunition Detection: Developing a Real Time Gun Detection Classifier”, Stanford University, Feb 2017

[4] Shreyamsh Kamate, “UAV: Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles”, Texas A&M University, 2015.

[5] Adrian Rosebrock, “Object detection with deep learning and OpenCV”, pyimagesearch.

[6] Mohana and H. V. R. Aradhya, "Elegant and efficient algorithms for real time object detection, counting and classification for video surveillance applications from single fixed camera," 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), Bangalore, 2016, pp. 1-7.

[7] Akshay Mangawati, Mohana, Mohammed Leesan, H. V. Ravish Aradhya, “Object Tracking Algorithms for video surveillance applications” International conference on communication and signal processing (ICCSP), India, 2018, pp. 0676-0680.

[8] Apoorva Raghunandan, Mohana, Pakala Raghav and H. V. Ravish Aradhya, “Object Detection Algorithms for video surveillance applications” International conference on communication and signal processing (ICCSP), India, 2018, pp. 0570-0575.

[9] Manjunath Jogin, Mohana, “Feature extraction using Convolution Neural Networks (CNN) and Deep Learning” 2018 IEEE International Conference On Recent Trends In Electronics Information Communication Technology,(RTEICT) 2018, India.

[10] Arka Prava Jana, Abhiraj Biswas, Mohana, “YOLO based Detection and Classificationof Objects in video records” 2018 IEEE International Conference On Recent Trends In Electronics Information Communication Technology,(RTEICT) 2018, India.

[11] Analogy. Wikipedia, Mar 2018.

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, andA. Zisserman.” The pascal visual object classes (voc) challenge.International journal of computer vision”, 88(2):303–

338, 2010.

[13] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg.Dssd: “Deconvolutional single shot detector”. arXiv preprintarXiv:1701.06659, 2017.

[14] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox,and A. Farhadi. Iqa:“Visual question answering in interactiveenvironments.” arXiv preprint arXiv:1712.03316, 2017.
[15] K. He, X. Zhang, S. Ren, and J. Sun.” Deep residual learninfor image recognition”. In Proceedings of the IEEE conferenceon computer vision and pattern recognition, pages 770–778, 2016.

[16] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara,A. Fathi, I. Fischer, Z.Wojna, Y. Song, S. Guadarrama, et al.”Speed/accuracy trade-offs for modern convolutional object detectors.”

[17] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija,A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit,S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik,D. Cai, Z. Feng, D. Narayanan, and K. Murphy.” Openimages:A public dataset for large-scale multi-label andmulti-class image classification.” Dataset fromhttps://github.com/openimages, 2017.

[18] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017.

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar.Focal loss for denseobject detection. arXiv preprintarXiv:1708.02002, 2017.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan,P. Doll´ar, and C. L. Zitnick.” Microsoft coco: Common objects in context. In European conference on computer vision,”