An industrial oriented mini project report

on

REAL-TIME OBJECT DETECTION AND RECOGNITION USING DEEP LEARNING

Submitted by

S.SAI MAHESH	**19W91A05K9**
V.RAHUL	**19W91A05N4**
P.V.KRISHNA CHAITANYA	**20W95A0521**
S.PRANEETH	**19W91A05L0**

Under the Esteemed Guidance of

Mrs.D.Pushpa,

Assistant Professor

TO

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfilment of the requirements for award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALLAREDDY INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

(Sponsored by Malla Reddy Educational society)

(Affiliated to JNTU, Hyderabad)

Maisammaguda, Dhulapally post,

Secunderabad-500014.

2022-2023

MALLA REDDY
INSTITUTE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution - UGC, Govt. of India)
(Sponsored by Malla Reddy Educational Society)
Approved by AICTE, New Delhi, Recognized Under 2(f) & 12(B)
Affiliated to JNTU, Hyderabad, Accredited by NBA & NAAC with 'A' Grade

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

This is to certify that this is the bonafide certificate of an industrial oriented mini project report titled "**REAL TIME OBJECT DETECTION AND RECOGNITION USING MACHINE LEARNING "** is submitted by **S.SAI MAHESH(19W91A05K9),V.RAHUL(19W91A05N4),P.V.KRISHNACHAITANYA(20W95A0521),S.PRANEETH (19W91A05L0)** of B. Tech in the partial fulfilment of the requirements for the degree of **Bachelor of Technology in Computer Science and Engineering**, Dept. of Computer Science & Engineering and this has not been submitted for the award of any other degree of this institution.

Internal Guide Sign **Head of the Department sign**

External Examiner sign

DECLARATION

we hereby declare that the Mini Project report entitled **"REAL TIME OBJECT DETECTION AND RECOGNITION USING DEEP LEARNING"** submitted to Malla Reddy Institute of Engineering and Technology(Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH), for the award of the degree of Bachelor of Technology in Computer Science & Engineering is a result of original industrial oriented mini project done by us.

It is further declared that the mini project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

S.SAI MAHESH	**19W91A05K9**
V.RAHUL	**19W91A05N4**
P.V. KRISHNA CHAITANYA	**20W95A0521**
S. PRANEETH	**19W91A05L0**

ACKNOWLEDGEMENT

First and foremost, I am grateful to the Principal **Dr. M. ASHOK**, for providing me with
all the resources in the college to make my project a success. I thank him for his valuable suggestions at the time of seminars which encouraged me to give my best in the project.

I would like to express my gratitude to **Dr. ANANTHA RAMAN G R**, Head of the Department, Department of Computer Science and Engineering for his support and valuable suggestions during the dissertation work.

I offer my sincere gratitude to my project - coordinator **Dr. MD.HASAN** and internal guide **Mrs.D.PUSHPA, Assistant Professor,** of Computer Science and Engineering department who has supported me throughout this project with their patience and valuable suggestions.

I would also like to thank all the supporting staff of the Dept. of CSE and all other departments who have been helpful directly or indirectly in making the project a success. I am extremely grateful to my parents for their blessings and prayers for my completion of project that gave me strength to do my project.

S.SAI MAHESH	**19W91A05K9**
V.RAHUL	**19W91A05N4**
P.V. KRISHNA CHAITANYA	**20W95A0521**
S. PRANEETH	**19W91A05L0**

CONTENTS

ABSTRACT

Object Detection is the process of finding and recognizing real-world object instances such as car, bike, TV, flowers, and humans out of an images or videos. An object detection technique lets you understand the details of an image or a video as it allows for the recognition, localization, and detection of multiple objects within an image. Object detection is a core problem in computer vision. Detection starts 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.

LIST OF FIGURES

LIST OF SCREENS

CHAPTER-1
INTRODUCTION

The motive of object detection is to recognize and locate all known objects in a scene. Preferably in 3D space, recovering pose of objects in 3D is very important for robotic control systems. Imparting intelligence to machines and making robots more and more autonomous and independent has been a sustaining technological dream for the mankind. It is our dream to let the robots take on tedious, boring, or dangerous work so that we can commit our time to more creative tasks. Unfortunately, the intelligent part seems to be still lagging behind. In real life, to achieve this goal, besides hardware development, we need the software that can enable robot the intelligence to do the work and act independently. One of the crucial components regarding this is vision, apart from other types of intelligences such as learning and cognitive thinking. A robot cannot be too intelligent if it cannot see and adapt to a dynamic environment.

The searching or recognition process in real time scenario is very difficult. So far, no effective solution has been found for this problem. Despite a lot of research in this area, the methods developed so far are not efficient, require long training time, are not suitable for real time application, and are not scalable to large number of classes. Object detection is relatively simpler if the machine is looking for detecting one particular object. However, recognizing all the objects inherently requires the skill to differentiate one object from the other, though they may be of same type. Such problem is very difficult for machines, if they do not know about the various possibilities of objects.

1.1 Motivation:

Blind people do lead a normal life with their own style of doing things. But, they definitely face troubles due to inaccessible infrastructure and social challenges. The biggest challenge for a blind person, especially the one with the complete loss of vision, is to navigate around places. Obviously, blind people roam easily around their house without any help because they know the position of everything in the house. Blind people have a tough time finding objects around them. . So we decided to make a REAL TIME OBJECT DETECTION System. We are interested in this project after we went through few papers in this area. As a result we are highly motivated to develop a system that recognizes objects in the real time environment.

1.2INTRODUCTION TO OBJECT DETECTION

Object Detection is the process of finding and recognizing real-world object instances such as car, bike, TV, flowers, and humans out of an images or videos. An object detection technique lets you understand the details of an image or a video as it allows for the recognition, localization, and detection of multiple objects within an image.
It is usually utilized in applications like image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).Object Detection is done through many ways:

- Feature Based Object Detection
- Viola Jones Object Detection
- SVM Classifications with HOG Features
- Deep Learning Object Detection

Object detection from a video in video surveillance applications is the major task these days. Object detection technique is used to identify required objects in video sequences and to cluster pixels of these objects.

The detection of an object in video sequence plays a major role in several applications specifically as video surveillance applications. Object detection in a video stream can be done by processes like pre-processing, segmentation, foreground and background extraction, feature extraction. 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 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.

1.3 DIGITAL IMAGE PROCESSING

Computerized picture preparing is a range portrayed by the requirement for broad test work to build up the practicality of proposed answers for a given issue. A critical trademark hidden the plan of picture preparing frameworks is the huge level of testing and experimentation that Typically is required before touching base at a satisfactory arrangement. This trademark infors that the capacity to plan approaches and rapidly model hopeful arrangements by and large assumes a noteworthy part in diminishing the cost and time required to land at a suitable framework execution.

1.4 WHAT IS DIP?

A picture might be characterized as a two-dimensional capacity f(x, y), where x, y are spatial directions, and the adequacy off at any combine of directions (x, y) is known as the power or dark level of the picture by then. Whenever x, y and the abundance estimation of are all limited discrete amounts, we call the picture a computerized picture. The field of DIP alludes to preparing advanced picture by methods for computerized PC. Advanced picture is made out of a limited number of components, each of which has a specific area and esteem. The components are called pixels. Vision is the most progressive of our sensor, so it is not amazing that picture play the absolute most imperative part in human observation. Be that as it may, dissimilar to people, who are constrained to the visual band of the EM range imaging machines cover practically the whole EM range, going from gamma to radio waves. They can work likewise on pictures produced by sources that people are not acclimated to partner with picture.

There is no broad understanding among creators in regards to where picture handling stops and other related territories, for example, picture examination and PC vision begin. Now and then a qualification is made by characterizing picture handling as a teach in which both the info and yield at a procedure are pictures. This is constraining and to some degree manufactured limit. The range of picture investigation is in the middle of picture preparing and PC vision.

There are no obvious limits in the continuum from picture handling toward one side to finish vision at the other. In any case, one helpful worldview is to consider three sorts of mechanized procedures in this continuum: low, mid and abnormal state forms. Low-level process includes primitive operations, for example, picture preparing to decrease commotion differentiate upgrade and picture honing. A low- level process is described by the way that both its sources of info and yields are pictures.

Mid-level process on pictures includes assignments, for example, division, depiction of that Question diminish them to a frame reasonable for PC handling and characterization of individual articles. A mid-level process is portrayed by the way that its sources of info by and large are pictures however its yields are properties removed from those pictures. At long last more Elevated amount handling includes "Understanding an outlet of perceived items, as in picture examination and at the farthest end of the continuum playing out the intellectual capacities typically connected with human vision. Advanced picture handling, as effectively characterized is utilized effectively in a wide scope of regions of outstanding social and monetary esteem.

1.5 MOBILE NETS

To build lightweight deep neural networks Mobile Nets are used. It is based on a streamlined architecture that uses depth-wise separable convolutions. Mobile Net uses 3×3 depth-wise separable convolutions that uses between 8 times less computation than standard convolution at solely a little reduction accuracy. Applications and use cases including object detection, fine grain classification, face attributes and large scale-localization.

1.6 TENSOR FLOW

Tensor flow is an open source software library for high performance numerical computation. It allows simple deployment of computation across a range of platforms (CPUs, GPUs, TPUs) due to its versatile design also from desktops to clusters of servers to mobile and edge devices. Tensor flow was designed and developed by researchers and engineers from the Google Brain team at intervals
Google's AI organization, it comes with robust support for machine learning and deep learning and the versatile numerical computation core is used across several alternative scientific domains.

To construct, train and deploy Object Detection Models TensorFlow is used that makes it easy and also it provides a collection of Detection Models pre-trained on the COCO dataset, the Kitti dataset, and the Open Images dataset. One among the numerous Detection Models is that the combination of Single Shot Detector (SSDs) and Mobile Nets architecture that is quick, efficient and doesn't need huge computational capability to accomplish the object Detection.

1.7 APPLICATION OF OBJECT DETECTION

The major applications of Object Detection are:

FACIAL RECOGNITION

"Deep Face" is a deep learning facial recognition system developed to identify human faces in a digital image. Designed and developed by a group of researchers in Facebook. Google also has its own facial recognition system in Google Photos, which automatically separates all the photos according to the person in the image.
There are various components involved in Facial Recognition or authors could say it focuses on various aspects like the eyes, nose, mouth and the eyebrows for recognizing a faces.

PEOPLE COUNTING

People counting is also a part of object detection which can be used for various purposes like finding person or a criminal; it is used for analyzing store performance or statistics of crowd during festivals.
This process is considered a difficult one as people move out of the frame quickly.

INDUSTRIAL QUALITY CHECK

Object detection also plays an important role in industrial processes to identify or recognize products. Finding a particular object through visual examination could be a basic task that's involved in multiple industrial processes like sorting, inventory management, machining, quality management, packaging and so on. Inventory management can be terribly tough as things are hard to trace in real time.
Automatic object counting and localization permits improving inventory accuracy.

SELF DRIVING CARS

Self-driving is the future most promising technology to be used, but the working behind can be very complex as it combines a variety of techniques to perceive their surroundings, including radar, laser light, GPS, odometer, and computer vision. Advanced control systems interpret sensory info to allow navigation methods to work, as well as obstacles and it. This is a big step towards Driverless cars as it happens at very fast speed.

SECURITY

Object Detection plays a vital role in the field of Security; it takes part in major fields such as face ID of Apple or the retina scan used in all the sci-fi movies. Government also widely use this application to access the security feed and match it with their existing database to find any criminals or to detecting objects like car number involved in criminal activities. The applications are limitless.

CHAPTER-2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Deep learning has gained a tremendous influence on how the world is adapting to Artificial Intelligence since the past few years. Some of the popular object detection algorithms are Region-based Convolutional Neural Networks (RCNN) and FasterRCNN, Single Shot Detector (SSD).

Disadvantages:

- Detection and tracking speed is slow
- Low accuracy

2.2 PROPOSED SYSTEM

Object detection, a subset of computer vision, is an automated method for locating interesting objects in an image with respect to the background. ... Like other computer vision tasks, deep learning is the state-of-art method to perform object detection, main Objective of (DNN) Deep learning and Mobile Nets to perform efficient implementation of detection and tracking. This algorithm performs efficient object detection while not compromising on the performance. algorithm to detect various objects in real time video sequence and track them in real time. This model showed excellent detection and tracking results on the object trained and can further utilized in specific scenarios to detect, track and respond to the particular targeted objects in the video surveillance.

Advantages:

- Detection and tracking speed is high compare to existing system
- High accuracy compare to existing system

2.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis

the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

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

TECHNICAL FEASIBILITY

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

SOCIAL FEASIBILITY

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

CHAPTER-3
SYSTEM REQUIREMENTS SPECIFICATION

REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

3.1 FUNCTIONAL REQUIREMENTS

Functional requirement should include function performed by a specific screen outline work-flows performed by the system and other business or compliance requirement the system must meet. Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified. The functional specification describes what the system must do, how the system does it is described in the design specification. If a user requirement specification was written, all requirements outlined in the user requirements specifications should be addressed in the functional requirements.

3.2 NON FUNCTIONAL REQUIREMENTS

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy (i.e. how precise are the systems numerical answers.).

3.3 INPUT & OUTPUT DESIGN

INPUT DESIGN

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

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users

and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

1. Convey information about past activities, current status or projections of the
2. Future.
3. Signal important events, opportunities, problems, or warnings.
4. Trigger an action.
5. Confirm an action.

3.4 SYSTEMS REQUIREMENT AND SPECIFICATION

Hardware Requirements

The most common set of requirements defined by any application or software application for virtual computer applications, also known as hardware, Hardware Requirements list is usually accompanied by a hardware compliance list (HCL), especially if there are applications. The HCL list checks hardware devices that are tested, compatible, and sometimes not compatible with a specific application or application. The following sections discuss various aspects of hardware requirements.

- RAM: 4GB and Higher
- Processor: Intel i3 and above
- Hard Disk: 500GB: Minimum

Software Requirements

Software requirements address the definition of software application requirements and pre-requisites that require computer installation to provide System performance. These prerequisites or requirements are not usually included in the software installation package and need to be installed separately before the software can be installed.

- Operating system : Windows 10 Ultimate.
- Coding Language : Python.
- Front-End : tkinter.
- Back-End : Python

CHAPTER- 4

SYSTEM DESIGN

4.1. INTRODUCTION

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

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

This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data

structures, file formats, output formats, and the major modules in the system and their specifications are decided.

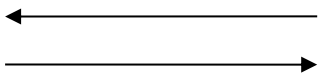
Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

4.1 Data Flow Diagrams:

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

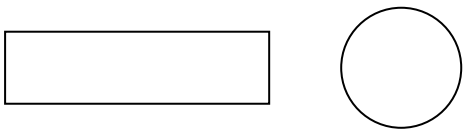
DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.

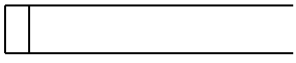


2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.



4. Data Store: Here data are stored or referenced by a process in the System.

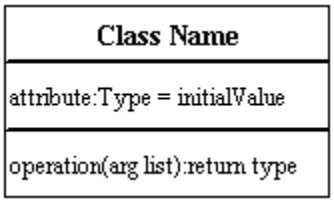


What is a UML Class Diagram?

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

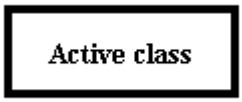
Basic Class Diagram Symbols and Notations

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.



Active Class

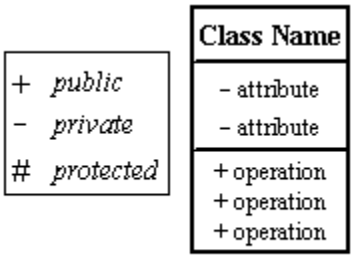
Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



Visibility

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.

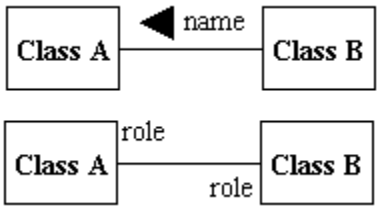
.



Associations

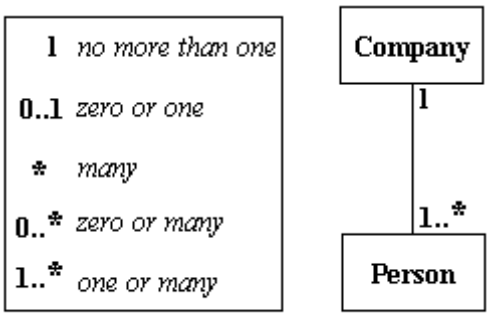
Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

Note: It's uncommon to name both the association and the class roles.



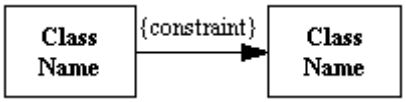
Multiplicity (Cardinality)

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.



Constraint

Place constraints inside curly braces {}.

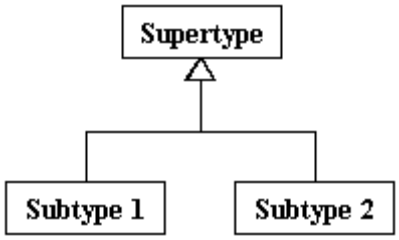


Simple Constraint

Generalization

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another.

For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

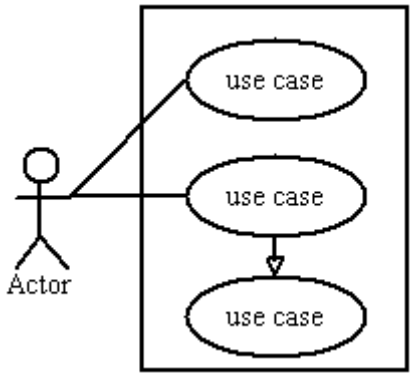
What is a UML Use Case Diagram?

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

Basic Use Case Diagram Symbols and Notations

System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

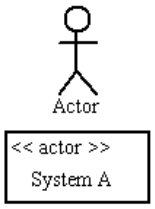


Use Case

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.

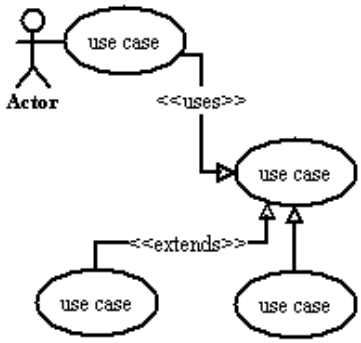
Actors

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



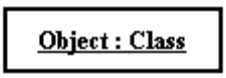
Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

Basic Sequence Diagram Symbols and Notations

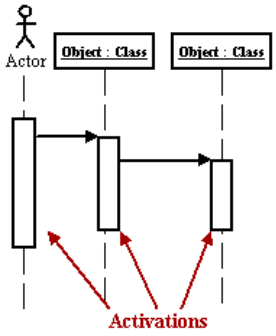
Class roles

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



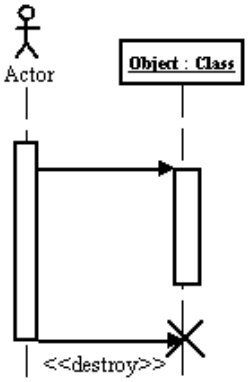
Activation

Activation boxes represent the time an object needs to complete a task.



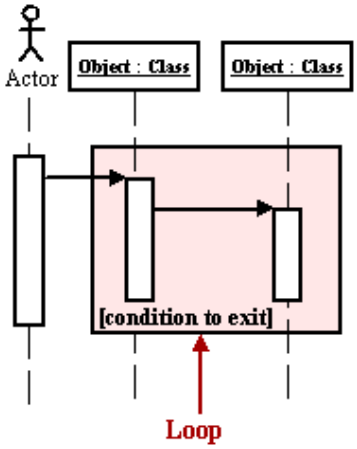
Destroying Objects

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X.



Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].



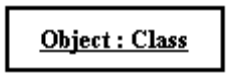
Collaboration Diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

Basic Collaboration Diagram Symbols and Notations

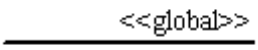
Class roles:

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.



Association roles

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.



Activity Diagram

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

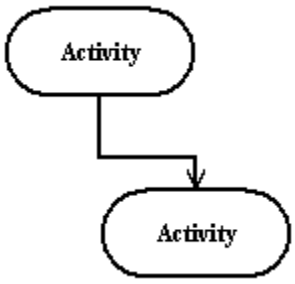
Basic Activity Diagram Symbols and Notations

Action states

Action states represent the non interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

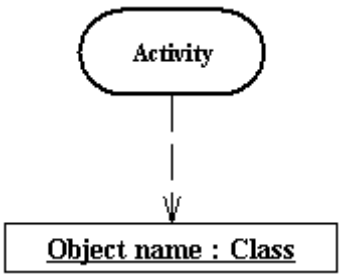
Action Flow

Action flow arrows illustrate the relationships among action states.



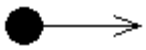
Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



Initial State

A filled circle followed by an arrow represents the initial action state.

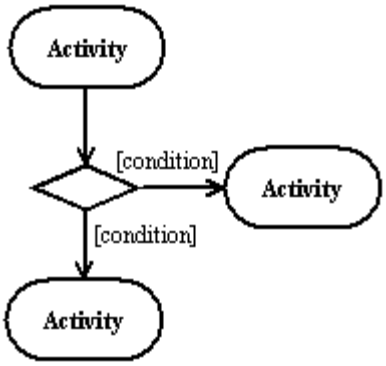


Final State

An arrow pointing to a filled circle nested inside another circle represents the final action state.

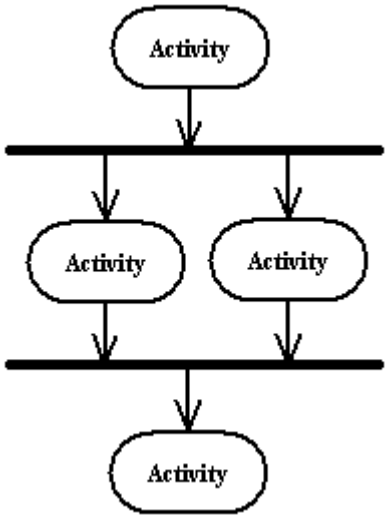
Branching

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



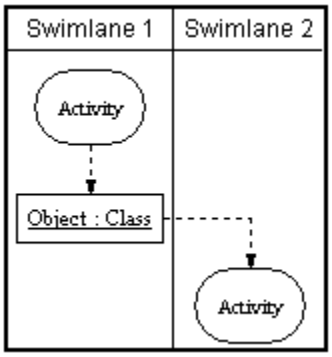
Synchronization

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



Swimlanes

Swimlanes group related activities into one column.



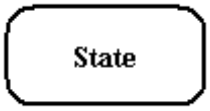
State chart Diagram

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

Basic State chart Diagram Symbols and Notations

States

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



Transition

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.

Initial State

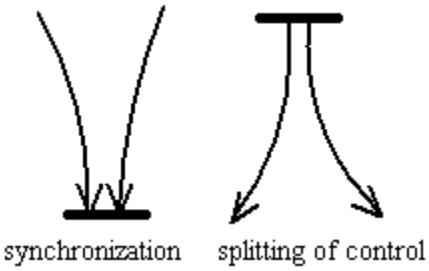
A filled circle followed by an arrow represents the object's initial state.

Final State

An arrow pointing to a filled circle nested inside another circle represents the object's **final state.**

Synchronization and Splitting of Control

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



STATE CHART DIAGRAM:

What is a UML Component Diagram?

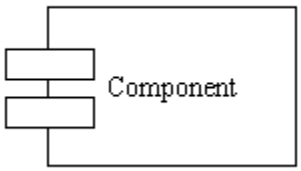
A component diagram describes the organization of the physical components in a system.

Basic Component Diagram Symbols and Notations

Component

A component is a physical building block of the system. It is represented as a rectangle with tabs.

Learn how to resize grouped objects like components.



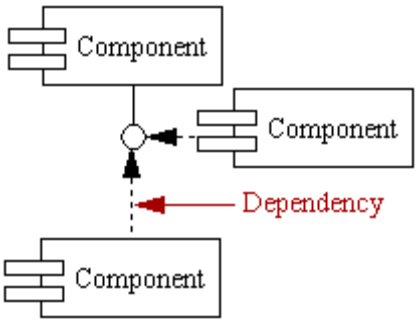
Interface

An interface describes a group of operations used or created by components.

Dependencies

Draw dependencies among components using dashed arrows.

Learn about line styles in Smart Draw.



COMPONENT DIAGRAM:

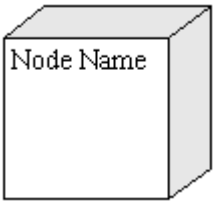
What is a UML Deployment Diagram?

Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

Basic Deployment Diagram Symbols and Notations

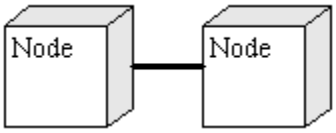
Component

A node is a physical resource that executes code components. Learn how to resize grouped objects like nodes.



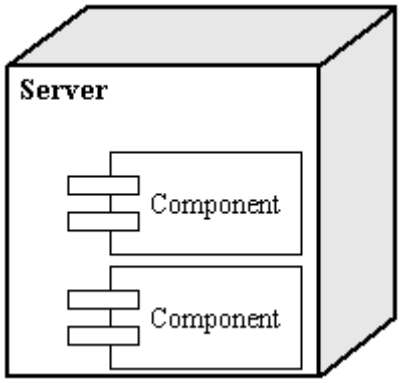
Association

Association refers to a physical connection between nodes, such as Ethernet. Learn how to connect two nodes.



Components and Nodes

Place components inside the node that deploys them.



UML Diagrams Overview

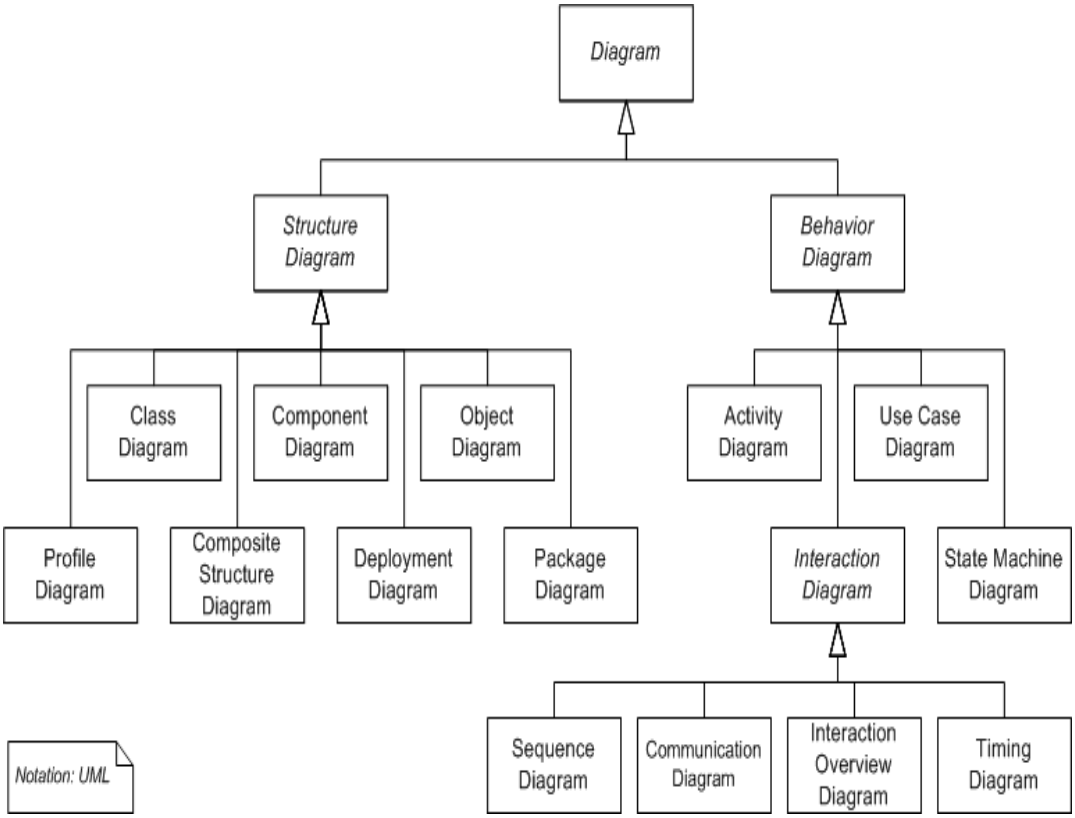


Fig 4.1.1 UML diagram Overview

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling

language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

4.2 UML DIAGRAMS

DATA FLOW DIAGRAM:

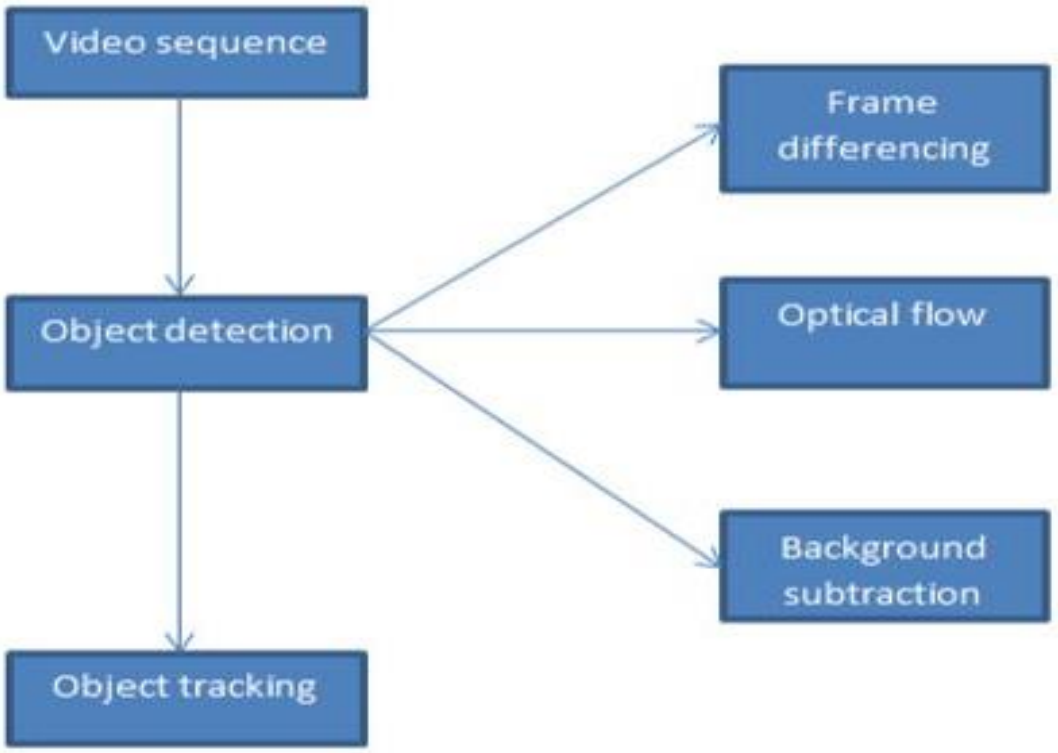


FIG 4.2.1 Data flow diagram for real time detection and Recognition

USE CASE DIAGRAM:

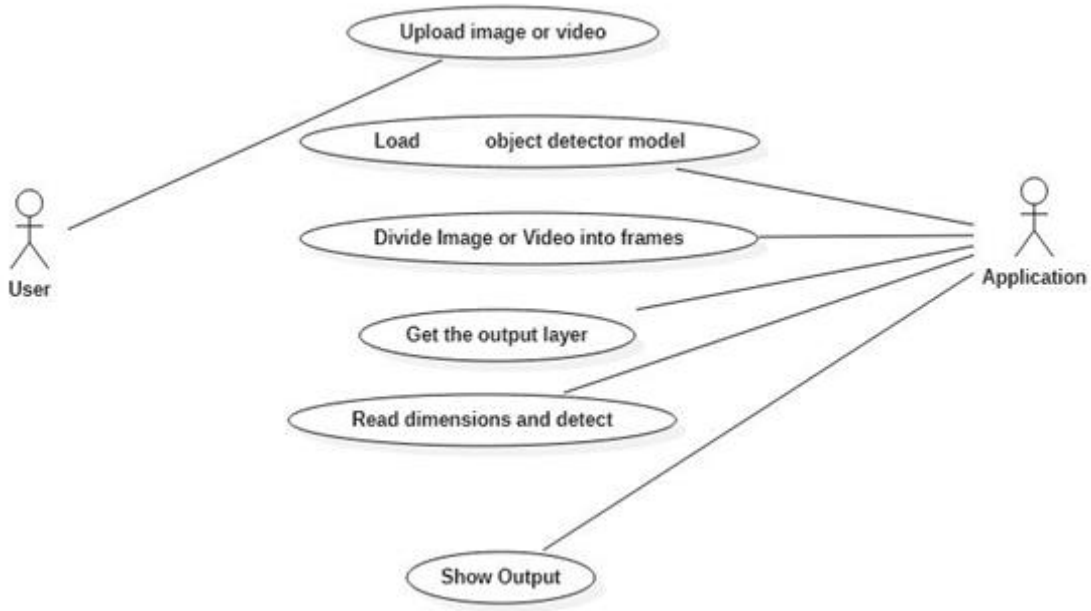


FIG 4.2.2 use case diagram for real time detection and Recognition

CLASS DIAGRAM:

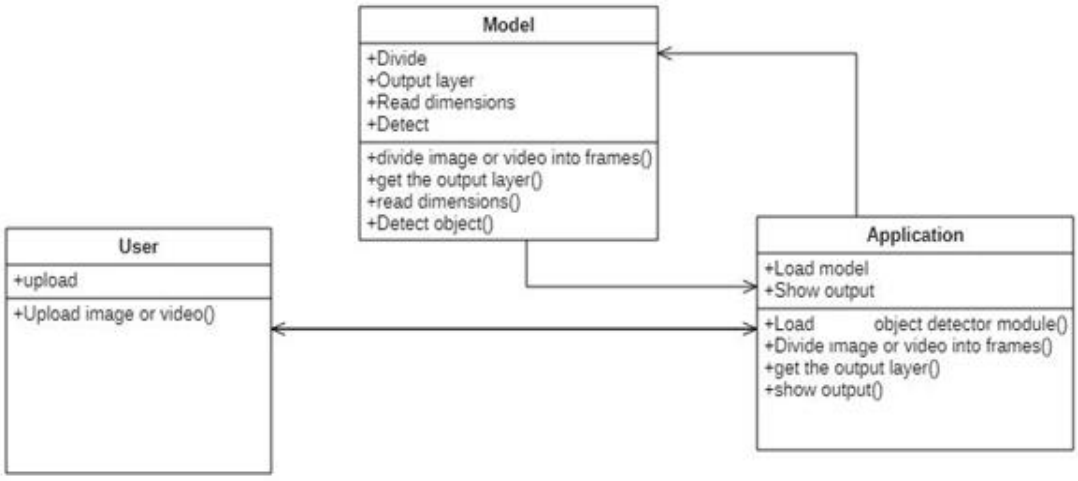


FIG 4.2.3 class diagram for real time detection and Recognition

ACTICITY DIAGRAM:

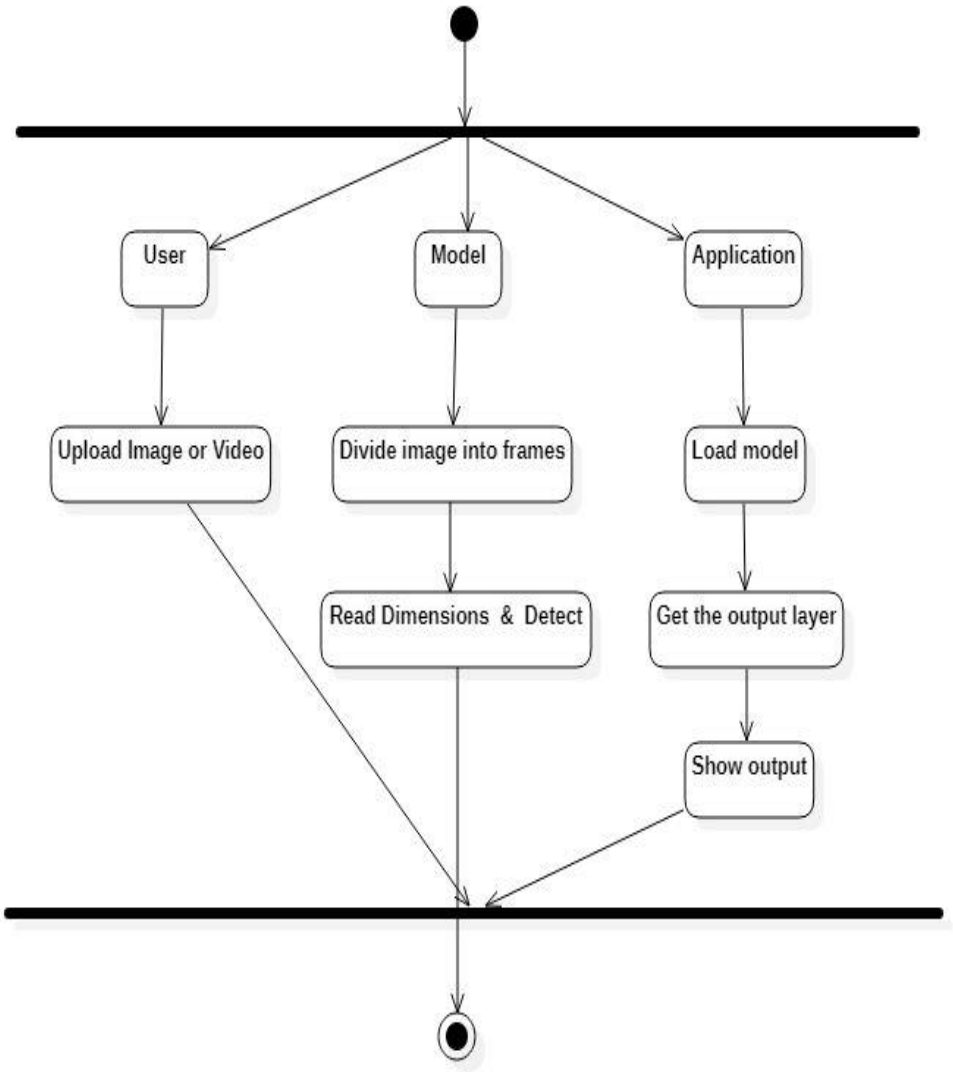


FIG 4.2.4 activity diagram for real time detection and Recognition

SEQUENCE DIAGRAM:

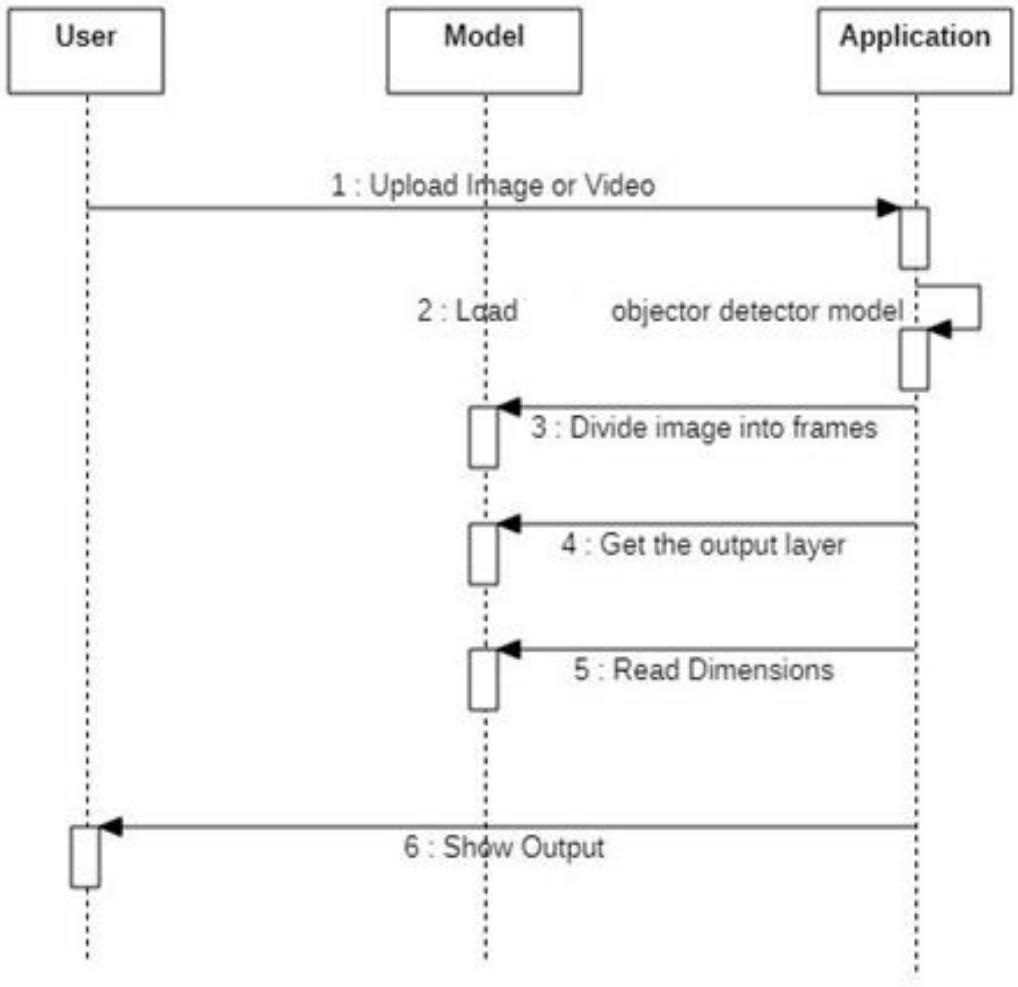


FIG 4.2.5 sequence diagram of real time detection and Recognition

COMPONENT DIAGRAM:

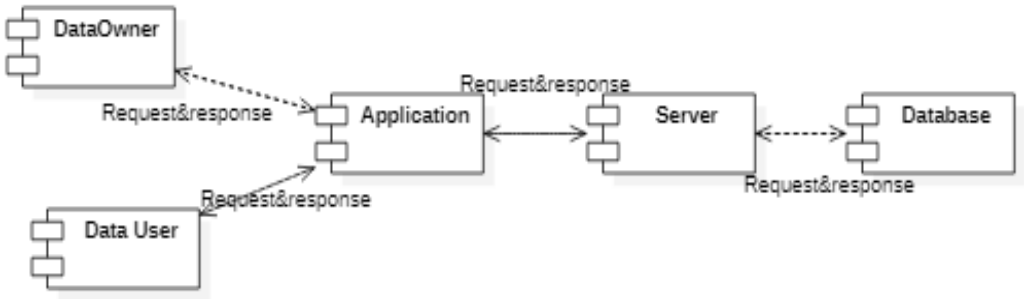


FIG 4.2.6 component diagram of real time detection and Recognition

CHAPTER-5

IMPLEMENTATION

5.1 MODULE & DESCRIPTION

Modules

1. **Object detection**
2. **Object tracking**

Modules description

Object detection

Frame 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.

Background 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. And predict shape of object using opev-CV and DNN.

Object 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.

5.2 TECHNOLOGY DESCRIPTION

Introduction of Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py":

C:\Users\Your Name>py

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.
>>>print("Hello, World!")
Hello, World!
Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:
exit()

Virtual Environments and Packages

Introduction

Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you've created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate
(tutorial-env) $ python
Python 3.5.1 (default, May  6 2016, 10:59:36)
  ...
>>> import sys
>>>sys.path
['', '/usr/local/lib/python35.zip', ...,
'~/envs/tutorial-env/lib/python3.5/site-packages']
>>>
```

Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can

browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature:

```
(tutorial-env) $ pip search astronomy

skyfield            - Elegant astronomy for Python
gary                - Galactic astronomy and gravitational dynamics.
novas               - The United States Naval Observatory NOVAS astronomy library
astroobs            - Provides astronomy ephemeris to plan telescope observations
PyAstronomy         - A collection of astronomy related tools for Python.
...
```

pip has a number of subcommands: "search", "install", "uninstall", "freeze", etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package's name:

```
(tutorial-env) $ pip install novas
Collecting novas
  Downloading novas-3.1.1.3.tar.gz (136kB)
Installing collected packages: novas
  Running setup.py install for novas
Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
(tutorial-env) $ pip install requests==2.6.0
Collecting requests==2.6.0
Using cached requests-2.6.0-py2.py3-none-any.whl
Installing collected packages: requests
Successfully installed requests-2.6.0
```

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

```
(tutorial-env) $ pip install --upgrade requests
Collecting requests
Installing collected packages: requests
Found existing installation: requests 2.6.0
```

Uninstalling requests-2.6.0:
Successfully uninstalled requests-2.6.0
Successfully installed requests-2.7.0
pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

```
(tutorial-env) $ pip show requests
---
Metadata-Version: 2.0
Name: requests
Version: 2.7.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: me@kennethreitz.com
License: Apache 2.0
Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages
Requires:
```

pip list will display all of the packages installed in the virtual environment:

```
(tutorial-env) $ pip list
novas (3.1.1.3)
numpy (1.9.2)
pip (7.0.3)
requests (2.7.0)
setuptools (16.0)
```

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

```
(tutorial-env) $ pip freeze > requirements.txt
(tutorial-env) $ cat requirements.txt
novas==3.1.1.3
numpy==1.9.2
```

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

```
(tutorial-env) $ pip install -r requirements.txt
Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))
  ...
Collecting numpy==1.9.2 (from -r requirements.txt (line 2))
  ...
Collecting requests==2.7.0 (from -r requirements.txt (line 3))
  ...
Installing collected packages: novas, numpy, requests
  Running setup.py install for novas
Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0
```

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

Using the Python Interpreter

Invoking the Interpreter. The Python interpreter is usually installed as /usr/local/bin/python3.8 on those machines where it is available; putting /usr/local/bin in your Unix shell's search path makes it possible to start it by typing the command:

python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn't work, you can exit the interpreter by typing the following command: quit().

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$ python3.8
Python 3.8 (default, Sep 16 2015, 09:25:04)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>>
>>>the_world_is_flat = True
>>>ifthe_world_is_flat:
...     print("Be careful not to fall off!")
...
Be careful not to fall off!
```

For more on interactive mode, see Interactive Mode.

The Interpreter and Its Environment

Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

```
# -*- coding: encoding -*-
```

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

```
# -*- coding: cp1252 -*-
```

One exception to the first line rule is when the source code starts with a UNIX “shebang” line. In this case, the encoding declaration should be added as the second line of the file. For example:

```
#!/usr/bin/env python3
# -*- coding: cp1252 -*-
```

Introduction to Artificial Intelligence

"The science and engineering of making intelligent machines, especially intelligent computer programs". -John McCarthy-

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

Trending AI Articles:

1. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data

2. Data Science Simplified Part 1: Principles and Process

3. Getting Started with Building Realtime API Infrastructure

4. AI & NLP Workshop

Applications of AI

- **Gaming** − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess, river crossing, N-queens problems and etc.

- **Natural Language Processing** − Interact with the computer that understands natural language spoken by humans.
- **Expert Systems** − Machine or software provide explanation and advice to the users.
- **Vision Systems** − Systems understand, explain, and describe visual input on the computer.
- **Speech Recognition** − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.
- **Handwriting Recognition** − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.
- **Intelligent Robots** − Robots are able to perform the instructions given by a human.

Major Goals

- Knowledge reasoning
- Planning
- Machine Learning
- Natural Language Processing
- Computer Vision
- Robotics

MACHINE LEARNING

Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of

this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

In this tutorial, we'll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. Correlation is a measure of association between two variables that are not designated as either dependent or independent. Regression at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

Approaches to machine learning are continuously being developed. For our purposes, we'll go through a few of the popular approaches that are being used in machine learning at the time of writing.

K-Nearest Neighbor

The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the k in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of k = 1, the object is assigned to the class of the single nearest neighbor. Let's look at an example of k-nearest neighbor. In the diagram below, there are blue diamond objects and orange star objects. These belong to two separate classes: the diamond class and the star class.

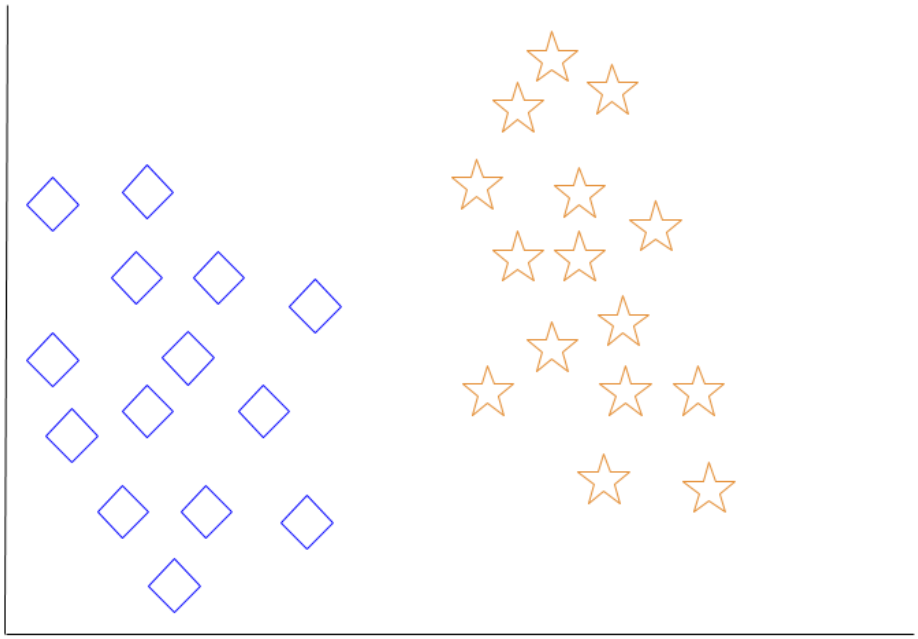


FIG 5.2.1 k-nearest neighbor -1

When a new object is added to the space — in this case a green heart — we will want the machine learning algorithm to classify the heart to a certain class.

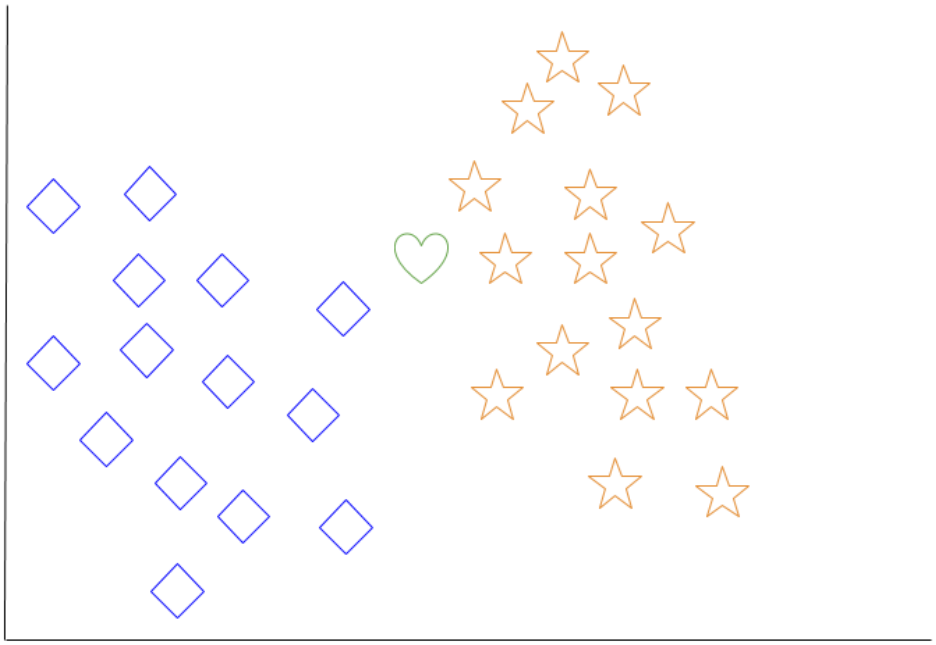


FIG 5.2.2 k-nearest neighbor-2

When we choose k = 3, the algorithm will find the three nearest neighbors of the green heart in order to classify it to either the diamond class or the star class.
In our diagram, the three nearest neighbors of the green heart are one diamond and two stars. Therefore, the algorithm will classify the heart with the star class.

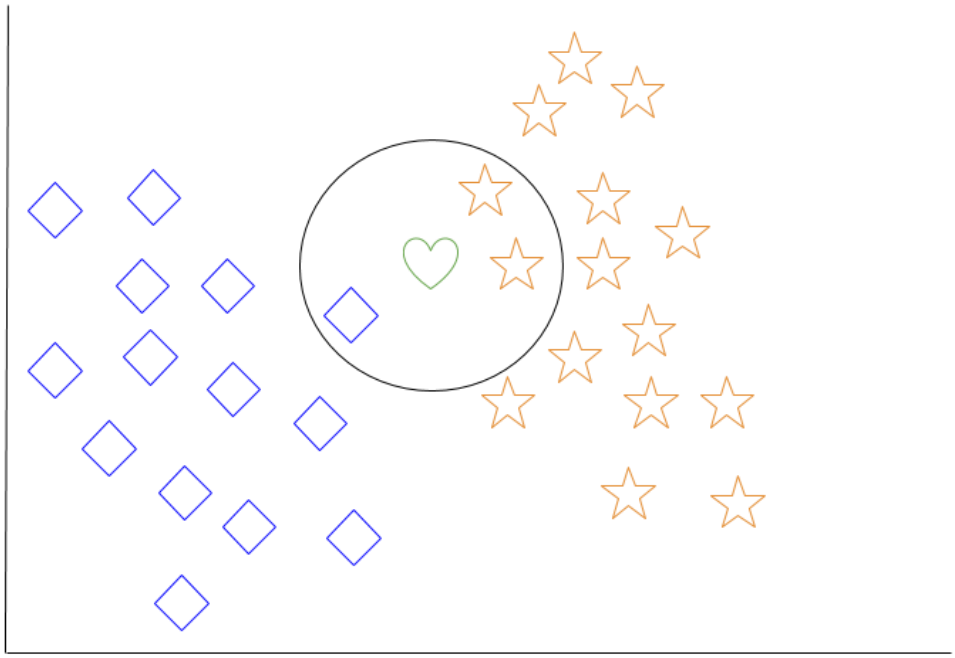


FIG 5.2.3 k-nearest neighbor with k=3

Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of "lazy learning" as generalization beyond the training data does not occur until a query is made to the system.

Decision Tree Learning

For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data's target value.

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data's attributes that are determined through observation are represented by the branches, while the conclusions about the data's target value are represented in the leaves.

When "learning" a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let's look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.

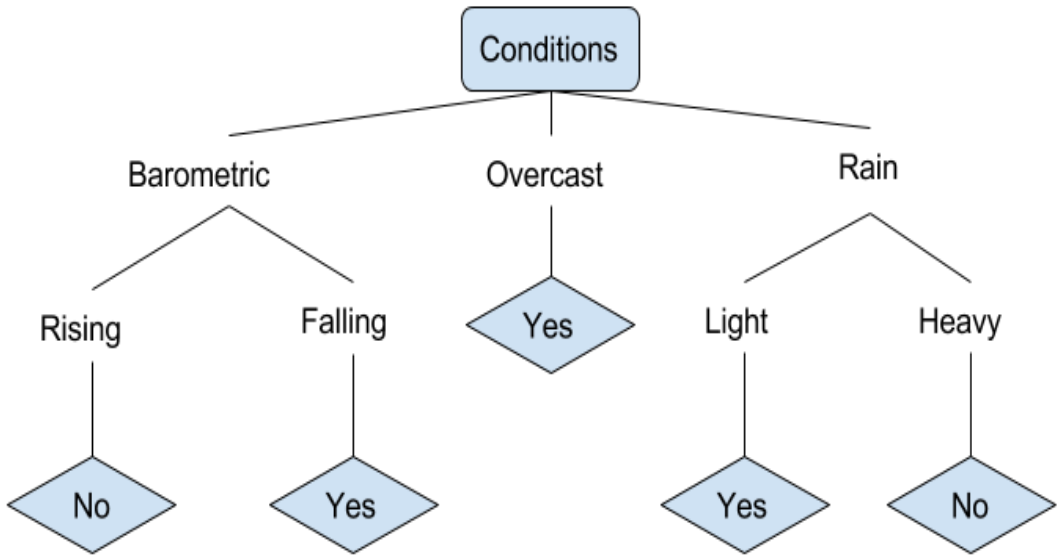


FIG 5.2.4 Condition Flowchart

In the simplified decision tree above, an example is classified by sorting it through the tree to the appropriate leaf node. This then returns the classification associated with the particular leaf, which in this case is either a `Yes` or a `No`. The tree classifies a day's conditions based on whether or not it is suitable for going fishing.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

INTRODUCTION TO DEEP LEARNING

What is deep learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

A formal definition of deep learning is- neurons

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours. The question here is how we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

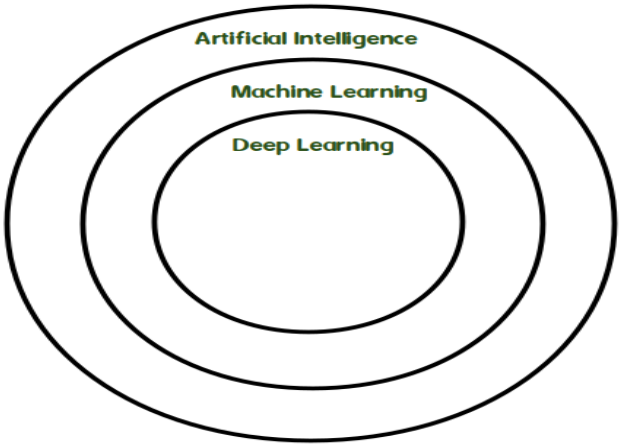


FIG 5.2.5 Deep Learning Overview

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development,

and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

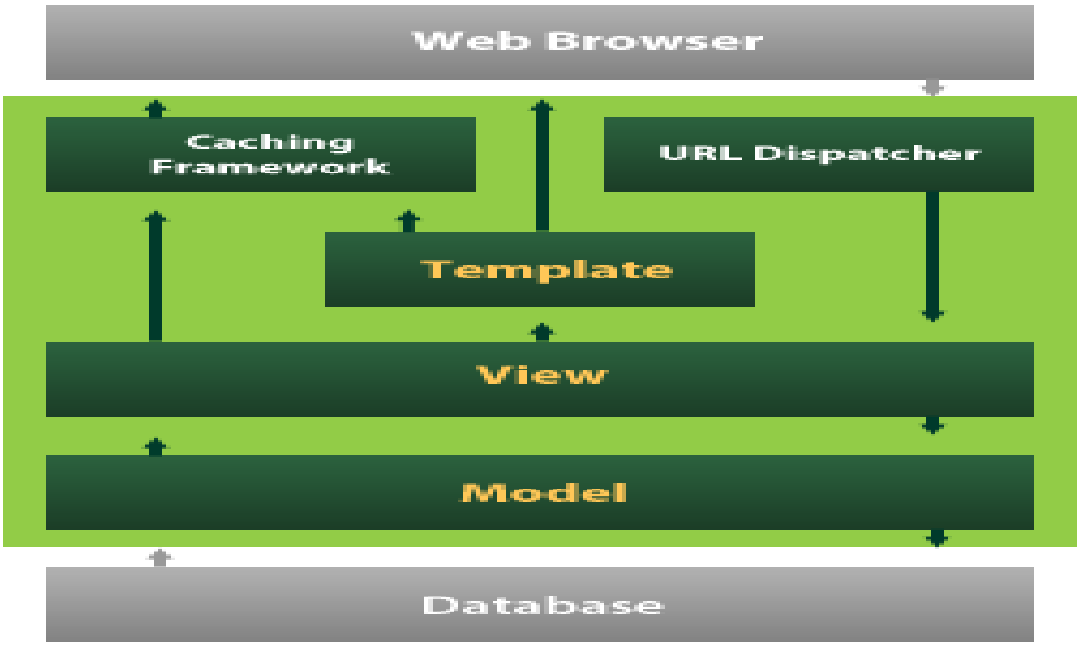


FIG 5.2.7 Django Data Model

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

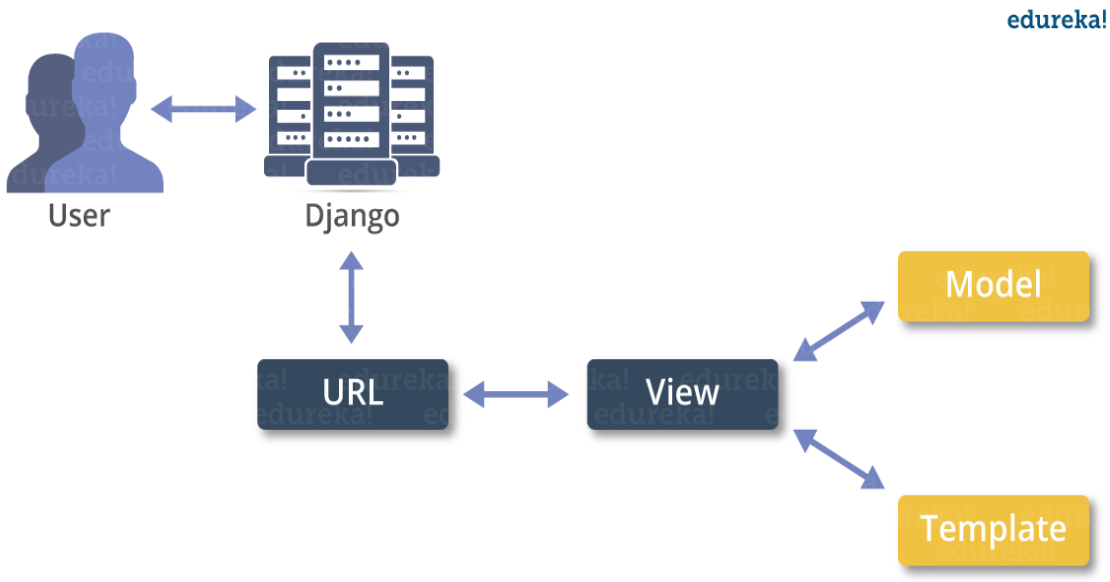


FIG 5.2.8 Django Implementation

5.3. CODING

```
# How to run?: python real_time_object_detection.py --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel
# python real_time.py --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel

# import 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 predictions")
args = vars(ap.parse_args())

# Arguments used here:
# prototxt = MobileNetSSD_deploy.prototxt.txt (required)
# model = MobileNetSSD_deploy.caffemodel (required)
# confidence = 0.2 (default)

# SSD (Single Shot MultiBox Detector) is a popular algorithm in object detection
```

```
# It has no delegated region proposal network and predicts the boundary boxes and the classes directly from feature maps in one single pass
# To improve accuracy, SSD introduces: small convolutional filters to predict object classes and offsets to default boundary boxes
# Mobilenet is a convolution neural network used to produce high-level features

# SSD is designed for object detection in real-time
# The SSD object detection composes of 2 parts: Extract feature maps, and apply convolution filters to detect objects

# Let's start by initialising the list of the 21 class labels MobileNet SSD was trained to.
# Each prediction composes of a boundary box and 21 scores for each class (one extra class for no object),
# and we pick the highest score as the class for the bounded object
CLASSES = ["aeroplane", "background", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]

# Assigning random colors to each of the classes
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# COLORS: a list of 21
R,G,B values, like ['101.097383   172.34857188 111.84805346'] for each label
# length of COLORS = length of CLASSES = 21

# load our serialized model
#The model from Caffe: MobileNetSSD_deploy.prototxt.txt; MobileNetSSD_deploy.caffemodel;
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
# print(net)
# <dnn_Net 0x128ce1310>
```

```
# initialize the video stream,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# warm up the camera for a couple of seconds
time.sleep(2.0)

# FPS: used to compute the (approximate) frames per second
# Start the FPS timer
fps = FPS().start()

# OpenCV provides two functions to facilitate image preprocessing for deep learning classification: cv2.dnn.blobFromImage and cv2.dnn.blobFromImages. Here we will use cv2.dnn.blobFromImage
# These two functions perform: Mean subtraction, Scaling, and optionally channel swapping

# Mean subtraction is used to help combat illumination changes in the input images in our dataset. We can therefore view mean subtraction as a technique used to aid our Convolutional Neural Networks
# Before we even begin training our deep neural network, we first compute the average pixel intensity across all images in the training set for each of the Red, Green, and Blue channels.
# we end up with three variables: mu_R, mu_G, and mu_B (3-tuple consisting of the mean of the Red, Green, and Blue channels)
# For example, the mean values for the ImageNet training set are R=103.93, G=116.77, and B=123.68
# When we are ready to pass an image through our network (whether for training or testing), we subtract the mean, \mu, from each input channel of the input image:
# R = R - mu_R
# G = G - mu_G
# B = B - mu_B

# We may also have a scaling factor, \sigma, which adds in a normalization:
```

```
# R = (R - mu_R) / sigma
# G = (G - mu_G) / sigma
# B = (B - mu_B) / sigma

# The value of \sigma may be the standard deviation across the training set (thereby turning the preprocessing step into a standard score/z-score)
# sigma may also be manually set (versus calculated) to scale the input image space into a particular range — it really depends on the architecture, how the network was trained

# cv2.dnn.blobFromImage creates 4-dimensional blob from image. Optionally resizes and crops image from center, subtract mean values, scales values by scalefactor, swap Blue and Red channels
# a blob is just an image(s) with the same spatial dimensions (width and height), same depth (number of channels), that have all be preprocessed in the same manner

# Consider the video stream as a series of frames. We capture each frame based on a certain FPS, and loop over each frame
# 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
        # vs is the VideoStream
        frame = vs.read()
        frame = imutils.resize(frame, width=400)
        print(frame.shape) # (225, 400, 3)
        # grab the frame dimensions and convert it to a blob
        # First 2 values are the h and w of the frame. Here h = 225 and w = 400
        (h, w) = frame.shape[:2]
        # Resize each frame
        resized_image = cv2.resize(frame, (300, 300))
        # Creating the blob
        # The function:
```

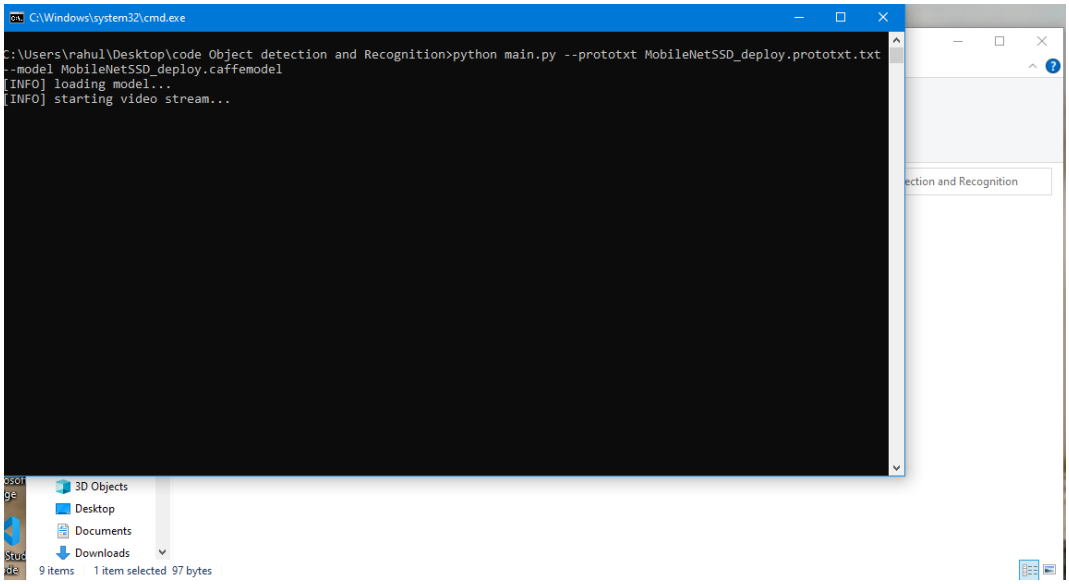
```
        # blob = cv2.dnn.blobFromImage(image, scalefactor=1.0, size, mean, swapRB=True)
        # image: the input image we want to preprocess before passing it through our deep neural network for classification
        # mean:
        # scalefactor: After we perform mean subtraction we can optionally scale our images by some factor. Default = 1.0
        # scalefactor  should be 1/sigma as we're actually multiplying the input channels (after mean subtraction) by scalefactor (Here 1/127.5)
        # swapRB : OpenCV assumes images are in BGR channel order; however, the 'mean' value assumes we are using RGB order.
        # To resolve this discrepancy we can swap the R and B channels in image  by setting this value to 'True'
        # By default OpenCV performs this channel swapping for us.

        blob = cv2.dnn.blobFromImage(resized_image, (1/127.5), (300, 300), 127.5, swapRB=True)
        # print(blob.shape) # (1, 3, 300, 300)
        # pass the blob through the network and obtain the predictions and predictions
        net.setInput(blob) # net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
        # Predictions:
        predictions = net.forward()

        # loop over the predictions
        for i in np.arange(0, predictions.shape[2]):
                # extract the confidence (i.e., probability) associated with the prediction
                # predictions.shape[2] = 100 here
                confidence = predictions[0, 0, i, 2]
                # Filter out predictions lesser than the minimum confidence level
                # Here, we set the default confidence as 0.2. Anything lesser than 0.2 will be filtered
                if confidence > args["confidence"]:
                        # extract the index of the class label from the 'predictions'
```

```
                    # idx is the index of the class label
                    # E.g. for person, idx = 15, for chair, idx = 9, etc.
                    idx = int(predictions[0, 0, i, 1])
                    # then compute the (x, y)-coordinates of the bounding box for
the object
                    box = predictions[0, 0, i, 3:7] * np.array([w, h, w, h])
                    # Example, box = [130.9669733    76.75442174 393.03834438
224.03566539]
                    # Convert them to integers: 130 76 393 224
                    (startX, startY, endX, endY) = box.astype("int")

                    # Get the label with the confidence score
                    label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)
                    print("Object detected: ", label)
                    # Draw a rectangle across the boundary of the object
                    cv2.rectangle(frame, (startX, startY), (endX, endY),
                            COLORS[idx], 2)
                    y = startY - 15 if startY - 15 > 15 else startY + 15
                    # Put a text outside the rectangular detection
                    #    Choose    the    font    of    your    choice:
FONT_HERSHEY_SIMPLEX,                FONT_HERSHEY_PLAIN,
FONT_HERSHEY_DUPLEX,                FONT_HERSHEY_COMPLEX,
FONT_HERSHEY_SCRIPT_COMPLEX, FONT_ITALIC, etc.
                    cv2.putText(frame,    label,    (startX,    y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

        # show the output frame
        cv2.imshow("Frame", frame)

        # HOW TO STOP THE VIDEOSTREAM?
        # Using cv2.waitKey(1) & 0xFF

        # The waitKey(0) function returns -1 when no input is made
        # As soon an event occurs i.e. when a button is pressed, it returns a 32-bit integer
```

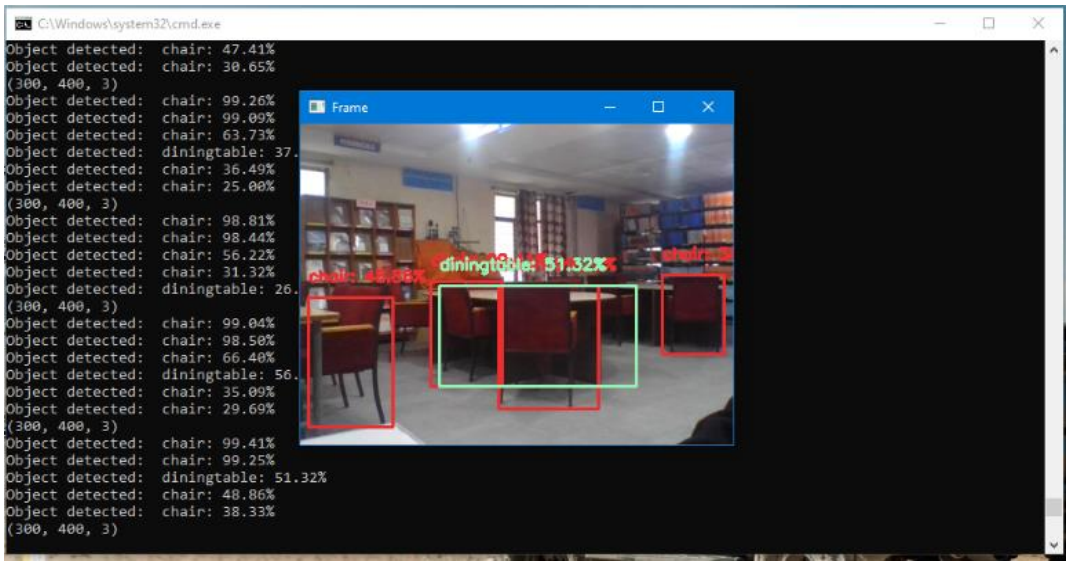
```
        # 0xFF represents 11111111, an 8 bit binary
        # since we only require 8 bits to represent a character we AND waitKey(0) to 0xFF, an integer below 255 is always obtained
        # ord(char) returns the ASCII value of the character which would be again maximum 255
        # by comparing the integer to the ord(char) value, we can check for a key pressed event and break the loop
        # ord("q") is 113. So once 'q' is pressed, we can write the code to break the loop
        # Case 1: When no button is pressed: cv2.waitKey(1) is -1; 0xFF = 255; So -1 & 255 gives 255
        # Case 2: When 'q' is pressed: ord("q") is 113; 0xFF = 255; So 113 & 255 gives 113

        # Explaining bitwise AND Operator ('&'):
        # The & operator yields the bitwise AND of its arguments
        # First you convert the numbers to binary and then do a bitwise AND operation
        # For example, (113 & 255):
        # Binary of 113: 01110001
        # Binary of 255: 11111111
        # 113 & 255 = 01110001 (From the left, 1&1 gives 1, 0&1 gives 0, 0&1 gives 0,... etc.)
        # 01110001 is the decimal for 113, which will be the output
        # So we will basically get the ord() of the key we press if we do a bitwise AND with 255.
        # ord() returns the unicode code point of the character. For e.g., ord('a') = 97; ord('q') = 113

        # Now, let's code this logic (just 3 lines, lol)
        key = cv2.waitKey(1) & 0xFF

        # Press 'q' key to break the loop
        if key == ord("q"):
                break
```

```
        # update the FPS counter
        fps.update()

# stop the timer
fps.stop()

# Display FPS Information: Total Elapsed time and an approximate FPS over the entire video stream
print("[INFO] Elapsed Time: {:.2f}".format(fps.elapsed()))
print("[INFO] Approximate FPS: {:.2f}".format(fps.fps()))

# Destroy windows and cleanup
cv2.destroyAllWindows()
# Stop the video stream
vs.stop()

# In case you removed the opaque tape over your laptop cam, make sure you put them back on once finished ;)
# YAYYYYYYYYYY WE ARE DONE!
```

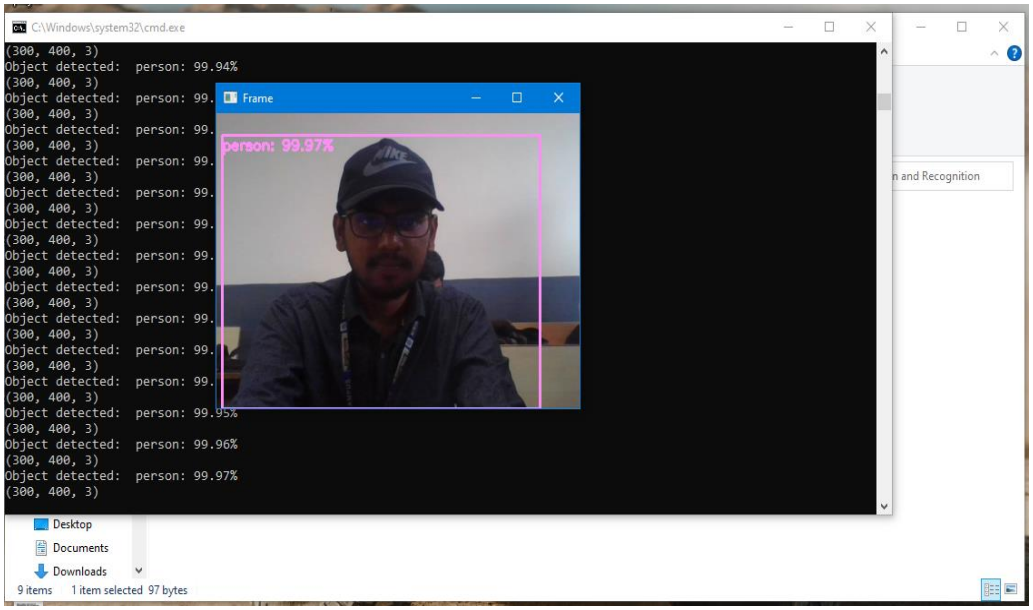
5.4 OUTPUT SCREENS



SCREEN 5.4.1 starting Execution



SCREEN 5.4.2 output 1



SCREEN 5.4.3 output -2

CHAPTER-6
SYSTEM TESTING

SYSTEM TEST

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

TYPES OF TESTS

Unit testing

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

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

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

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

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

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

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

Acceptance Testing

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

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

7.CONCLUSION

Deep-learning based object detection has been a search hotspot in recent years. This project starts on generic object detection, which give base architectures for other related tasks. With the assistance of this the 3 other common tasks, namely object detection, face detection and pedestrian detection, are often accomplished. Authors accomplished this by combing 2 things: Object detection with deep learning and OpenCV and Efficient, threaded video streams with OpenCV. The camera sensor noise and lightening condition can change the result because it can create problem in Recognizing the objects. Object Detection algorithms act as a mixture of both image classification and object localization. It takes the given image as input and produces the output having the bounding boxes adequate to the amount of objects present within the image, with the category label attached to every bounding box at the highest.

8.REFERENCES

1. Geethapriya S, N. Duraimurugan, S.P. Chokkalingam, “Real-Time Object Detection with Yolo”, International Journal of Engineering and Advanced Technology (IJEAT)
2. Abdul Vahab, Maruti S Naik, Prasanna G Raikar an Prasad S R4, “Applications of Object Detection System”, International Research Journal of Engineering and Technology (IRJET)
3. Hammad Naeem, Jawad Ahmad and Muhammad Tayyab, “Real-Time Object Detection and Tracking”, IEEE
4. Meera M K, & Shajee Mohan B S. 2016, "Object recognition in images", International Conference on Information Science (ICIS).
5. Astha Gautam, Anjana Kumari, Pankaj Singh: "The Concept of Object Recognition", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 3, March 2015
6. Joseph Redmon, Santosh Divvala, Ross Girshick, “You Only Look Once: Unified, Real-Time Object Detection”, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2016,pp. 779- 788
7. V. Gajjar, A. Gurnani and Y. Khandhediya, "Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach," in 2017 IEEE International Conference on Computer Vision Workshops, 2017.