



KLE Technological
University
Creating Value
Leveraging Knowledge

BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Mini Project Report

On

Real-time Object Detection using YOLO11

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Mangalgouri P Kademani	01FE23BCS422
Yuvaraj P Rathod	01FE23BCS423
Shruthi A Nagave	01FE23BCS421
Nisha B Kubasad	01FE23BCI401

Under the guidance of

Dr. G S Hanchinamani

School of Computer Science and Engineering

KLE Technological University, Hubballi

2024-2025



KLE Technological
University
Creating Value
Leveraging Knowledge

BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2024-25

CERTIFICATE

This is to certify that project entitled “Real-Time Object Detection using YOLO11” is a bonafied work carried out by the student team Mangalgouri P Kademani 01FE23BCS422, Yuvaraj P Rathod 01FE23BCS423, Shruthi A Nagave 01FE23BCS421, Nisha B Kubasad 01FE23BCI401, in partial fulfillment of the completion of 5th semester B. E. course during the year 2024 – 2025. The project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said course.

Guide

Dr. G S Hanchinamani

Head, SoCSE

Dr. Vijayalakshmi.M.

External Viva-Voce

Name of the examiners

Signature with date

1 _____

2 _____

ABSTRACT

Real-time object detection into advanced deep learning models that provide quick and precise recognition of multiple things in a changing environment, making them suitable for use in applications like human-computer interaction, autonomous systems, and surveillance. With a focus on object classification and real-time analysis, the most advanced **YOLO11** architecture is used in the proposed system.

Strong performance in a range of situations and lighting circumstances is made possible by the system's use of extensive image preprocessing techniques like scaling, normalization, and augmentation. The quality of the input data is improved by this preprocessing, which guarantees effective feature extraction and dependable detection using the detection framework of **YOLO11** in challenging circumstances.

The model YOLO11 achieves a faster and more accurate detection by utilizing the latest advancements in transformer-based modules, enhanced attention mechanisms, and optimized convolutional layers. In a single forward pass, the model predicts both object classes and bounding boxes concurrently, providing outstanding performance for real-time applications.

Using an advanced and effective architecture, the suggested solution simplifies the process and reduces dependency on conventional, resource-intensive object detection practices. With improved speed and accuracy in detection, this method offers great promise for using YOLO11 for real-time applications. It is well-suited to many kinds of use cases in machine learning, autonomous vehicles, security, and other high-performance industries that need object detection.

Keywords : *Deep Learning Models, YOLO11 Architecture, Object Classification, Image Preprocessing, Scaling and Normalization, Data Augmentation, Feature Extraction, Neural Networks, Bounding Box Detection, Autonomous Systems, Human-Computer Interaction, Surveillance Systems, Computer Vision, Dynamic Environment Analysis.*

ACKNOWLEDGEMENT

We would like to thank our faculty and management for their professional guidance towards the completion of the mini project work. We take this opportunity to thank Dr. Ashok Shettar, Pro-Chancellor, Dr. P.G Tewari, Vice-Chancellor and Dr. B.S.Anami, Registrar for their vision and support.

We also take this opportunity to thank Dr. Meena S. M, Professor and Dean of Faculty, SoCSE and Dr. Vijayalakshmi M, Professor and Head, SoCSE for having provided us direction and facilitated for enhancement of skills and academic growth.

We thank our guide Dr. G S Hanchinamani, Senior Lecturer and SoCSE for the constant guidance during interaction and reviews.

We extend our acknowledgment to the reviewers for critical suggestions and inputs. We also thank Project coordinator Dr. Uday Kulkarni, and reviewers for their suggestions during the course of completion. We express gratitude to our beloved parents for constant encouragement and support.

Mangalgouri P Kademani - 01FE23BCS422

Yuvaraj P Rathod - 01FE23BCS423

Shruthi A Nagave - 01FE23BCS421

Nisha B Kubasad - 01FE23BCI401

CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	i
CONTENTS	iii
LIST OF TABLES	iv
LIST OF FIGURES	v
1 INTRODUCTION	1
1.1 Preamble	1
1.2 Motivation	1
1.3 Objectives of the project	2
1.4 Literature Review / Survey	2
1.5 Problem Definition	4
2 SOFTWARE REQUIREMENT SPECIFICATION	5
2.1 Overview of SRS	5
2.2 Requirement Specifications	5
2.2.1 Functional Requirements	5
2.2.2 Use case diagram	6
2.2.3 Use Case descriptions	6
2.2.4 Nonfunctional Requirements	9
2.3 Software and Hardware requirement specifications	10
2.3.1 Software requirements	10
2.3.2 Hardware requirements	10
3 PROPOSED SYSTEM	11
3.1 Description of Proposed System.	11
3.2 Description of Target Users	12
3.3 Advantages of Proposed System	13
3.4 Scope (Boundary of proposed system)	13
4 SYSTEM DESIGN	14
4.1 Architecture of the system	14

4.2	Class Diagram	16
4.3	Sequence Diagram	18
5	IMPLEMENTATION	20
5.1	Proposed Methodology	20
5.2	Description of Modules	21
6	TESTING	23
6.1	Test Cases	23
7	RESULTS & DISCUSSIONS	26
8	CONCLUSION AND FUTURE SCOPE	30
	REFERENCES	30
9	Plagiarism Report	32

LIST OF TABLES

6.1	Test Cases for Proposed System	23
-----	--	----

LIST OF FIGURES

2.1	Use Case Diagram for Real-time Object Detection	6
3.1	Workflow for real-time object detection	11
4.1	Architecture diagram of the system	14
4.2	Class diagram of the system	16
4.3	Sequence Diagram for Real-time object detection	18
5.1	Activity Diagram for Real-time object detection using YOLO11	20
7.1	Input Selection Dashboard	26
7.2	Object detection	27
7.3	Image detection	28
7.4	Video detection	29

Chapter 1

INTRODUCTION

Real-time object detection is essential to many computer vision applications, from surveillance systems to driverless cars. YOLO (You Only Look Once), an innovative method that transformed object detection by considering it as a single regression problem, was first described. In contrast to more conventional techniques, YOLO uses a single neural network to process the entire image, resulting in remarkably high speed and accuracy [1]. For real-time object detection in a variety of real-world applications, the work described above established the groundwork and opened the way for further improvements.

1.1 Preamble

This project uses YOLO11, the most recent iteration of the YOLO family of algorithms, to detect objects in real-time applications. The best performance is achieved through the integration of complex neural architecture design, improved feature extraction techniques, and optimal training procedures. The architecture of YOLO11 is covered in this project, with particular focus on how well it can provide capabilities in a variety of difficult situations. This project shows that YOLO11 has raised the bar for real-time object detection [2] by analyzing its contributions.

1.2 Motivation

The need for good performance object detection systems and their growing complexity are the driving forces behind this project. While early iterations of YOLO and traditional techniques are highly successful, they are limited in situations involving speed or extremely complex circumstances. In this context, YOLO11 uses unique characteristics including transformer-based modules, improved loss functions, and more advanced data augmentation techniques to address these problems. An overview of the recent advancements influencing computer vision can be obtained by studying YOLO11. The study is motivated by the potential uses, such as industrial automation, medical imaging, and autonomous navigation, where accuracy and efficiency are essential.

1.3 Objectives of the project

The project "Real-Time Object Detection Using YOLO11" aims to create and implement an effective real-time object detection system by comprehending the architecture of YOLO11, assessing its accuracy, speed, and robustness, putting it into practice on real-world datasets, and determining how well it adapts to various environments. The main objective's are:

- To recognize objects on Webcam and images.
- To keep track of objects even when they are moving in real-time.
- To perform in different conditions like varying lighting, angles and backgrounds.
- To work on all types of operating system.

1.4 Literature Review / Survey

1] Title: You Only Look Once:Unified, Real-Time Object Detection [3]

Authors:Joseph Redmon, Santosh Divvala.

Goal: Present a real-time object detection technique that enhances cross-domain generalization.

Methodology: YOLO predicts bounding boxes and class probabilities from complete images using a single convolutional neural network. The image is divided into an $S \times S$ grid, where each grid cell predicts B bounding boxes and confidence scores. YOLO is based on GoogLeNet architecture and processes images at up to 155 frames per second.

Key Findings: YOLO outperforms traditional methods like R-CNN and DPM in real-time detection, showing high accuracy and generalizing well to new domains such as artwork. It is faster and more accurate than other detection methods.

Gaps: YOLO struggles with small objects and unusual aspect ratios, causing localization issues. Its architecture may not match the performance of specialized systems for certain object classes and could impact bounding box accuracy.

2] Title: A real-time object detection algorithm for video [4]

Authors:Shengyu Lu, Beizhan Wang, Hongji Wang.

Goal: Addresses the high computing time in video-based object detection by proposing Fast YOLO, a fast object detection technique that increases detection speed and accuracy.

Methodology: By using short convolutions to reduce parameters, preparing images to remove background effects, and comparing its performance with Faster R-CNN and SSD, Fast

YOLO improves upon the original YOLO network. **Key Findings:** Real-time object recognition in videos is made possible by Fast YOLO's high detection speed, accuracy, and computing economy.

Gaps: The algorithm's adaptation to a variety of objects or situations needs more research, as the publication ignores possible constraints in dynamic surroundings or changing lighting conditions. Future research recommends improving the algorithm and incorporating cutting-edge techniques.

3] Title: YOLOv1 to YOLOv10: The Fastest and Most Accurate Real-time Object Detection Systems [5]

Authors: Chien-Yao Wang, Hong-Yuan Mark Liao.

Goal: This study examines the development of the YOLO series, evaluating its influence on real-time computer vision and motivating further developments in the area.

Methodology: It highlights applications in domains such as autonomous driving and visual surveillance while analyzing the technological developments and integration of YOLO versions with innovative methods.

Key Findings:: The YOLO series has had a significant impact on real-time object detection, surpassing conventional techniques in accuracy and exhibiting model generalizability and wide applicability.

Gaps: The study draws attention to the paucity of studies on YOLO's integration with current technologies and its potential for advancement to meet novel real-time detection difficulties.

4] Title: An improved deep learning-based optimal object detection system from images [6]

Authors: Satya Prakash Yadav, Muskan Jindal, Preeti Rani.

Goal: By contrasting the YOLO, SSD, and Faster R-CNN algorithms, the study seeks to enhance object recognition in challenging situations.

Methodology: It examines current approaches, applies the algorithms to real-world datasets, and evaluates their effectiveness using metrics like accuracy, recall, precision, and loss.

Key Findings: Comparison tables show that R-CNN performs better than YOLO and SSD in terms of accuracy, recall, precision, and loss.

Gaps: Current techniques have trouble resolving time-space complexity and capturing high-definition information; in order to maximize performance, additional advancements and hybrid approaches are needed.

1.5 Problem Definition

Creating a real-time object detection system that uses the YOLO11 algorithm to detect safely and effectively and categorize multiple items maintaining high performance for a variety of dynamic inputs.

Chapter 2

SOFTWARE REQUIREMENT SPECIFICATION

The major goal of the SRS paper is to offer readers with a full overview of our model, including all of its qualities and goals. This document describes the software requirements for the project.

2.1 Overview of SRS

To perform real-time object identification using YOLO11, the requirements are outlined in depth in the Software Requirement Specification (SRS). It contains the software and hardware specs, use case descriptions, and functional and nonfunctional requirements for the system. The paper provides as a guide to guarantee that the system satisfies performance, usability, and compatibility requirements while addressing actual object detection problems.

2.2 Requirement Specifications

Requirement specification tells how a software should work and what is expected from the software, these requirements should be relevant and detailed, there are functional and non-functional requirements. it should also contain a description about the verification and working of the project.

2.2.1 Functional Requirements

Functional requirements explain the service that the software/project offers, about the input and output to the software, and about its behavior. Functional requirements specify what the system must accomplish.

- The user should be able to start the system, and select the option where user want to detect object (e.g.: webcam, video , image).
- The system shall connect to the specified webcam and capture a continuous video stream in real-time.

- The system shall use the YOLO algorithm to detect objects in real-time and display bounding boxes, labels, and confidence scores on the video stream.
- The user should be able to stop the object detection process.
- The system shall stop the detection process, terminate the webcam stream, and release system resources when the detection is stopped.

2.2.2 Use case diagram

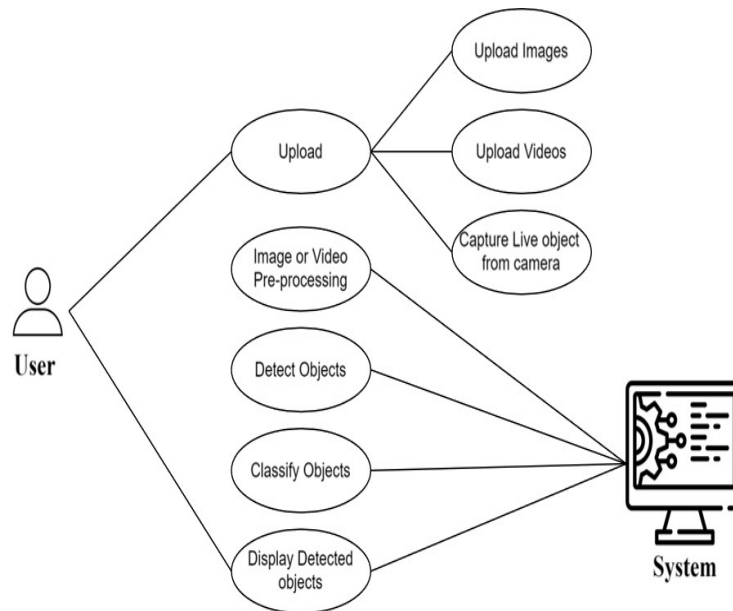


Figure 2.1: Use Case Diagram for Real-time Object Detection

2.2.3 Use Case descriptions

Use Case 1: Add Pictures or Videos

Actors: User

Description: To detect objects, the user uploads a picture or video file.

Prerequisites:

- The system is up and running.
- The user can post legitimate image or video files.

Events in Progress:

1. The user chooses to upload.

2. The user is prompted by the system to select whether to submit a video or an image.
3. From their device, the user chooses a file.
4. The uploaded file format (such as JPEG, PNG, or MP4) is verified by the system.
5. The file is temporarily stored by the system for processing.

Postconditions:

- The video or picture file has been uploaded successfully and is prepared for pre-processing.
- The system shows an error notice if a file is uploaded that is not valid.

Use Case 2: Capture Live Object from Camera

Actors: User

Description: In order to detect objects, the user records a live stream from a linked camera.

Prerequisites:

- The system is linked to a camera.
- The camera has permissions to be accessed by the system.

Events in Progress:

1. The user agrees to use live capture.
2. After setting up the camera, the system starts recording the live footage.
3. Pre-processing is done on the acquired frames.

Postconditions:

- Real-time processing and successful collection of the live video feed.

Use Case 3: Identifying Objects

Actors: System

Description: The system uses YOLO11 to analyze the input and recognize objects.

Prerequisites:

- A live feed or input file is accessible for detection.
- The system is integrated with the pre-trained YOLO11 model.

Events in Progress:

1. The YOLO11 model is loaded by the system.
2. If the input is video, it is separated into frames.
3. For the purpose of object detection, every frame or image is examined.
4. The system creates labeled bounding boxes and recognizes items.

Postconditions:

- Bounding boxes and class labels are used to identify and tag every object that is recognized.

Use Case 4: Object Classification

Actors: System

Description: The system divides observed objects into groups based on predetermined criteria.

Prerequisites:

- The input contains objects.
- The necessary categories of classification are supported by the YOLO11 model.

Events in Progress:

1. The confidence score determines the category to which each object is assigned.
2. The output is prepared by the system using items that are classified.

Postconditions:

- After standards are met, the items are categorized and prepared for presentation.

Use Case 5: Display Identified Objects

Actors: User

Description: The user sees the labels and bounding boxes that are produced by item detection.

Prerequisites:

- Complete object classification and detection.

Events in Progress:

1. The system applies labels and bounding boxes to the input image or video.
2. The output that has been processed is shown on the screen.

Postconditions:

- In real time, the viewer can view the items that have been spotted and their classifications.

2.2.4 Nonfunctional Requirements

Constraints that interfere with the operation of the system are known as nonfunctional requirements. If the software is delivered before even if the non functional requirements are not met then the software won't meet the users expectations.

Performance requirements

- System should predict the result in less than 30 seconds (test timing).
- Configuration should be easy, taking no more than 5 minutes.

Nonfunctional Requirements

- The accuracy should be presented in the form of percentage
- Work well in different conditions (lighting, weather, obstructions).
- Have a simple, clear display of detected objects.
- Be easy to use and configure.

Usability

- The UI should be easy to understand, showing detected objects with clear labels and boxes.
- If something goes wrong, the system should give clear error messages and help.

2.3 Software and Hardware requirement specifications

2.3.1 Software requirements

- Linux or Windows 10/11.
- TensorFlow/PyTorch for model development

2.3.2 Hardware requirements

- 8 Gb of RAM.
- PC with intel core i5 or i7 processor.
- Camera to Access a webcam.

Chapter 3

PROPOSED SYSTEM

The proposed system contains a thorough explanation of the YOLO11 based real-time object detection system. The workflow, target users, benefits, and system scope are all described in this chapter. Utilizing YOLO11's sophisticated capabilities, the suggested system seeks to provide accurate and efficient object recognition for a range of real-world uses. The components and features of the system are described in detail in the sections that follow, with a focus on its real-time processing capabilities and simplicity of integration into various domains.

3.1 Description of Proposed System.

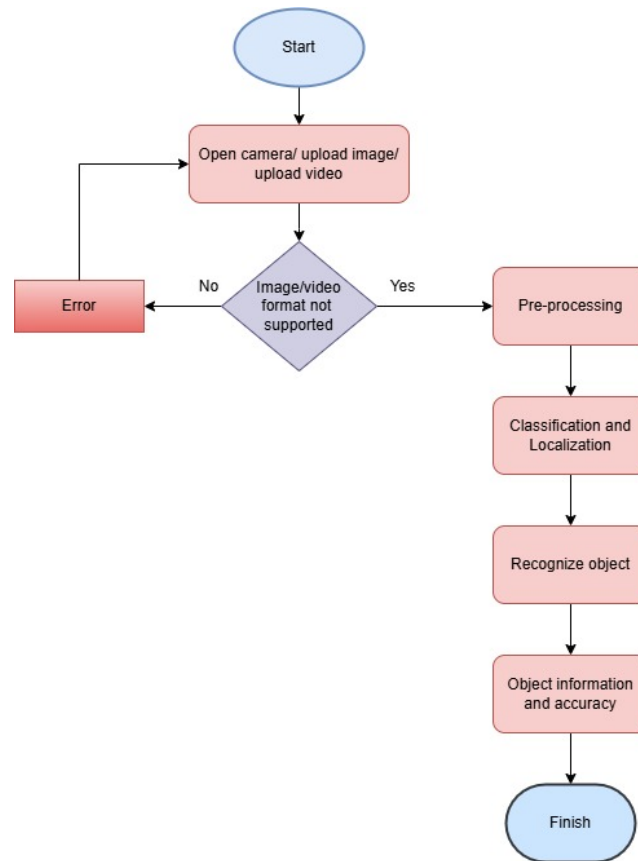


Figure 3.1: Workflow for real-time object detection

A number of processes are involved in the suggested YOLO11 real-time object recognition system in order to identify and categorize items from live camera feeds, uploaded photos, or videos. The workflow of the system is structured:

- **Start:**After initiating, the system awaits input.
- **Input:** Users have the option to submit a video, an image, or a camera feed.
- **Format Check:** The system checks for compatibility with the input format. An error notice appears if the format is not supported. The system moves on to the following stage if the format is supported.
- **Pre-processing:**To make sure the input is in the right format for the YOLO11 model, it is pre-processed.
- **Localization and Classification:** The YOLO11 model locates and classifies items in the input.
- **Object Recognition:**Identified and named items are recognized.
- **Accuracy and Object Information:** The system gives details on the objects it has detected as well as the detection's accuracy.
- **Finish:** The procedure is finished, and the outcomes are displayed.

3.2 Description of Target Users

The following are the system's target users:

- **Security employees:** To keep an eye on and quickly identify any suspicious activity.
- **Developers of autonomous vehicles:** To improve perception systems for obstacle avoidance and navigation.
- **Retail managers:** For automated tracking of inventory and analysis of consumer behavior.
- **Robotics engineers:** To make it possible for robots to recognize and engage with objects in their surroundings.
- **City planners:** To oversee public safety and regulate traffic.
- **AI Researchers and Developers:** In the domains of artificial intelligence and computer vision.

3.3 Advantages of Proposed System

- **High Accuracy:** Makes use of the comprehensive YOLO11 model to provide accurate object recognition and classification.
- **Real-Time Processing:** Suitable for time-sensitive applications, this feature can process live video inputs with minimal delay.
- **Flexibility:** Useful in a number of fields, such as robotics, retail, automotive, and security.
- **Scalability:** Adaptable to a variety of hardware platforms.
- **Efficiency:** Designed to operate smoothly even in contexts with limited resources, efficiency is maximized for speed and performance.
- **Integration Ease:** Because of its modular design, it is simple to integrate with current workflows and systems.

3.4 Scope (Boundary of proposed system)

The proposed system's scope consists of:

- **Real-Time Detection:** Dedicated to activities involving the real-time detection of objects.
- **Input Formats:** Only picture and video formats are supported as input types.
- **Standard Object Classes:** Only predefined classes that are supported by the YOLO11 model are detected.
- **Hardware Requirements:** Designed to operate on PCs with high-performance CPUs or GPUs, or other systems with sufficient processing capability.
- **Environmental Conditions:** Capable of managing certain real-world variations, but optimized for controlled situations.

Chapter 4

SYSTEM DESIGN

An overview of the real-time object detection system's implementation details utilizing YOLO11 is given in this chapter. The functionality, code skeleton, and detailed descriptions of each system component are provided in terms of the algorithm. Real-time performance, precise object detection, and efficient processing are all assured by the components design. The system architecture, important component interactions diagrams, and an explanation of the dataset utilized in the implementation are covered in the parts that follow.

4.1 Architecture of the system

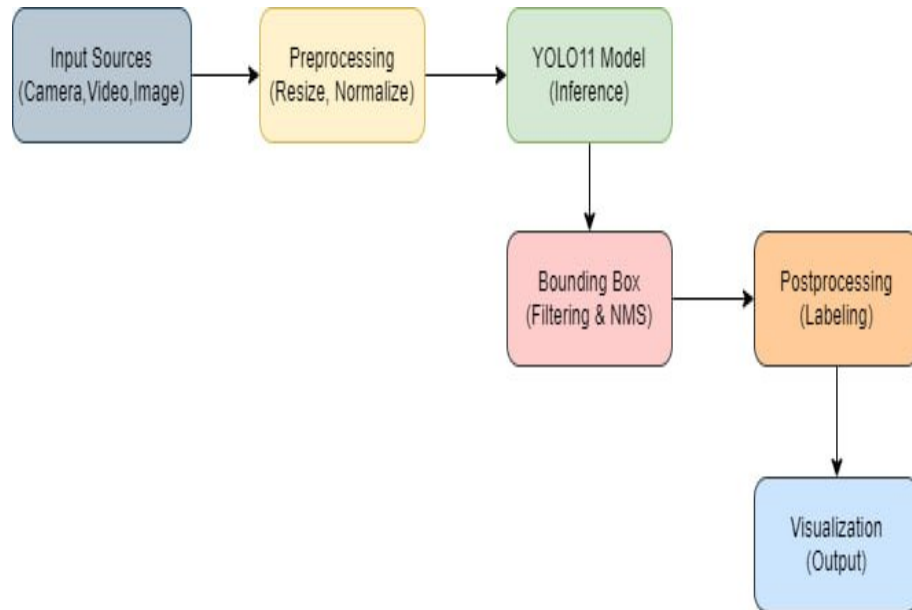


Figure 4.1: Architecture diagram of the system

The YOLO11 real-time object detection system's architecture in Figure 4.1 is made to assure accurate and effective object recognition in real-time situations. The following elements make up the system:

Description of the Components:

- **Input Sources (Cameras, Images):** This module acts as the system's entrance point, gathering input data from either pre-recorded image/video datasets or live camera feeds.

- **Preprocessing (Resize, Normalize):** To guarantee compatibility with the YOLO11 model, the incoming data is preprocessed. To improve model performance, steps include normalizing pixel values and scaling the input to match the expected size of the model.
- **YOLO11 Model (Inference):** YOLO11, the system's central component, uses the preprocessed input to recognize objects. For the identified items, it provides confidence scores, class labels, and bounding box coordinates.
- **Bounding Box Filtering Non-Max Suppression (NMS):** NMS is used to eliminate overlapping or redundant bounding boxes, leaving only the most certain detections.
- **Postprocessing (Labeling):** To get ready for output display, the improved bounding boxes are labeled with the appropriate class labels and confidence ratings.
- **Visualization (Output):** The input stream displays the processed output, with labels and bounding boxes used to identify any discovered objects.

Algorithm 1 YOLO11-Based Real-Time Object Detection Architecture Flow

Require: YOLO11 model, input frames (images/videos).

Ensure: Detected objects with bounding boxes and class labels for all frames.

1. Initialize the input source (e.g., cameras, datasets) and YOLO11 model.
 2. frame f captured from the input source
 3. Preprocess the frame f : Resize and normalize.
 4. Perform inference on f using the YOLO11 model to generate raw detections.
 5. Apply Non-Max Suppression (NMS) to filter redundant bounding boxes.
 6. Annotate the detections with class labels and confidence scores.
 7. Display the annotated frame in the output visualization.
 8. Release resources upon exit (e.g., close input streams).
 9. **return** Detected objects.
-

An Algorithm1 for detecting objects in real time Implementing an effective object detection system involves a series of steps that are described in Using YOLO11. Accurate processing of the incoming data and the generation of real-time outputs with low latency are ensured by each stage of the architecture.

4.2 Class Diagram

The Figure4.2 is a class diagram for a system that uses YOLO11 to provide real-time object detection. It illustrates the connections and exchanges between the system's several components.

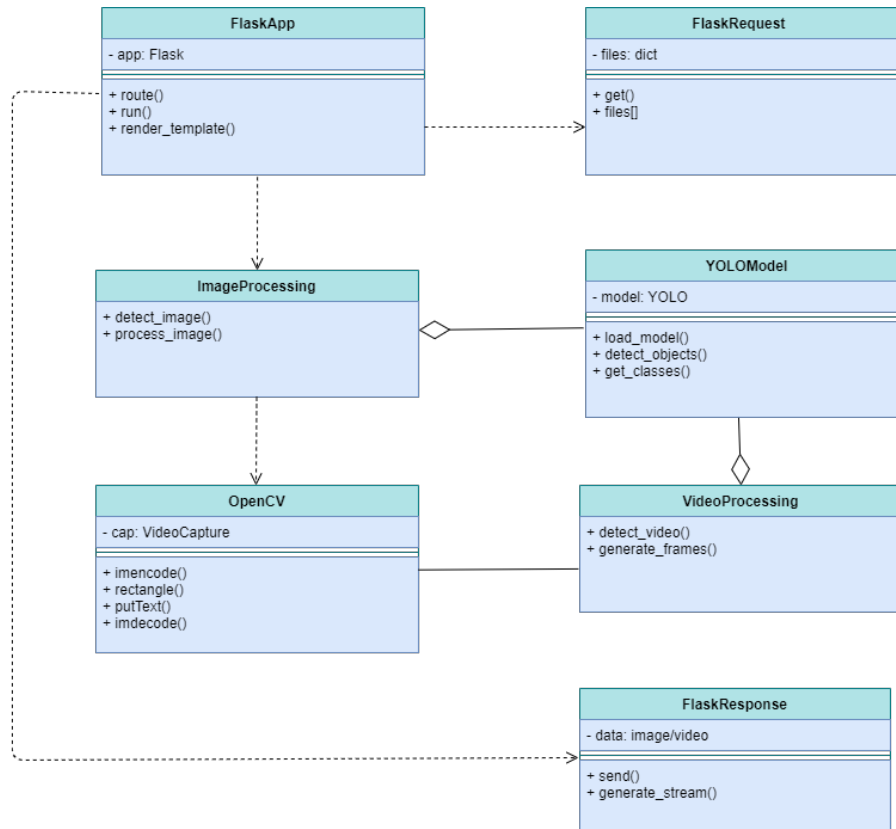


Figure 4.2: Class diagram of the system

- **FlaskApp**: The primary application layer in charge of managing HTTP requests and answers is called FlaskApp. It has functions to start the server, define application routes, and render HTML templates for the user interface.
- **FlaskRequest**: Incoming client requests, like file uploads, are handled by the FlaskRequest component. It offers the ability to access uploaded media, like pictures or movies, and retrieve data from the requests.
- **FlaskResponse**: This module manages responses that are sent to the client, such as live video streams and processed photos. It can broadcast real-time video data or provide the customer with results.

- **YOLO11 model:** The YOLO11 model's fundamental component for object detection is called YOLO Model. Methods for loading the YOLO11 model, identifying objects in input data, and obtaining the classes of objects found are all included.
- **ImageProcessing:** This part is in charge of managing inputs that are static images. It offers techniques for preprocessing photos and applying the YOLO11 model to identify things in them.
- **VideoProcessing:** This module handles video input processing. It successively processes video frames in order to find objects in video files.
- **OpenCV:** This useful tool offers crucial features for processing images and videos. It can be used for things like managing video frames, adding labels, and creating bounding boxes around identified objects.

Algorithm 2 Real-Time Object Detection Using YOLO11

Require: Input source (webcam, image, or video), YOLO11 model

Ensure: Detected objects with bounding boxes and labels

```

1: Initialize Flask application and load YOLO11 model.
2: Accept input source (webcam, image, or video).
3: if Input Source is Webcam then
4:   Capture frames using OpenCV.
5:   while Webcam is active do
6:     Process frame and detect objects.
7:     Annotate frame with bounding boxes and stream back.
8:   end while
9: else if Input Source is Image then
10:  Upload and preprocess the image.
11:  Detect objects and annotate the image.
12:  Return the processed image to the user.
13: else if Input Source is Video then
14:  Upload and preprocess the video.
15:  while Frames are available do
16:    Process frame and detect objects.
17:  end while
18:  Return the processed video to the user.
19: else
20:  Return an error message for invalid input.
21: end if

```

4.3 Sequence Diagram

The Figure 4.3 is a sequence diagram that shows how different parts of a system using YOLO11 for object detection interact with one another.

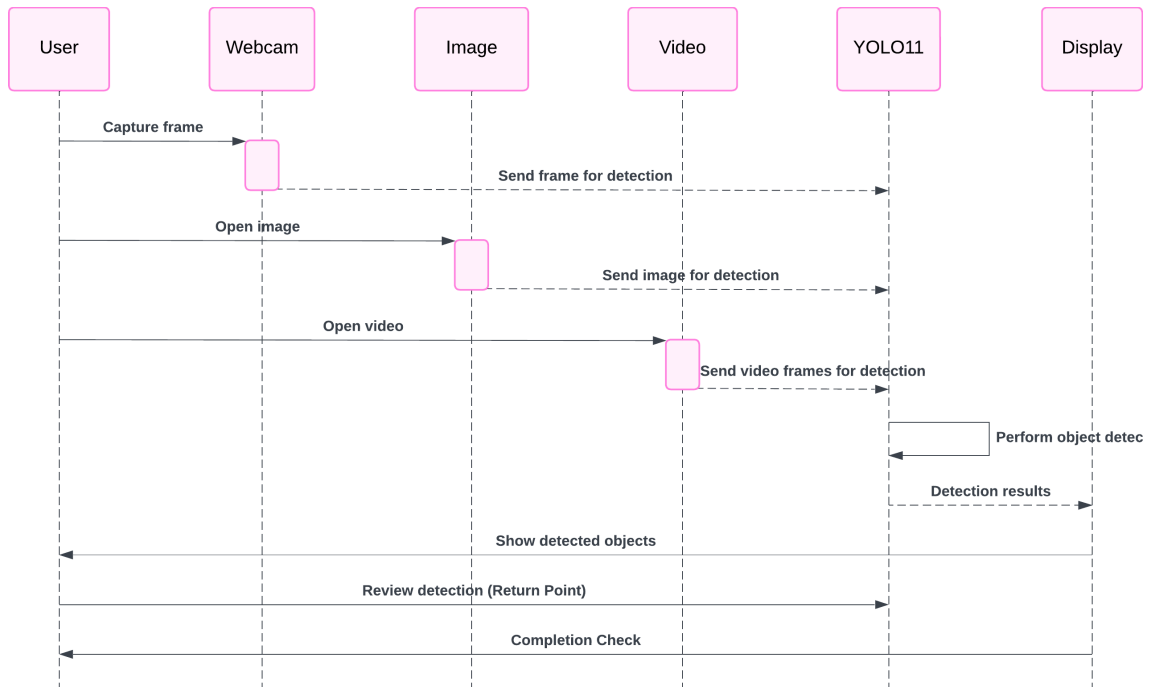


Figure 4.3: Sequence Diagram for Real-time object detection

The flow of operations in a real-time object identification system utilizing YOLO11 is illustrated by the sequence diagram. The user, webcam, image, video, YOLO11, and display are the six primary participants. The following are the interactions:

- **Capture a Webcam Frame:** By taking a picture using the webcam, the user starts the process. The YOLO11 module receives the captured frame and uses it to detect objects.
- **Uploading and Detecting Images:** For object detection, the user can also manually upload a picture. The YOLO11 module receives the image and analyzes it.
- **Uploading videos and processing frames:** The user uploads a video file for video inputs. After removing frames from the video, the system successively transmits them to YOLO11 for detection.
- **Object Detection with YOLO11:** Using the received frames or images, the YOLO11 module detects objects. It recognizes and categorizes items in the visual data.

- **Presentation of Detection Outcomes:** The display module receives the item detection results, including labels and bounding boxes. The user is shown the things that have been detected.
- **Examining and finishing:** The user examines the findings of the detection. Once the user certifies that the detection work has been completed, the process comes to an end.

Chapter 5

IMPLEMENTATION

In the implementation chapter, this describe in detail methods to implement YOLO11 to develop a real-time object detection system. Preprocessing, model integration, postprocessing, visualization, and data collection are all included in this. Each step is described along with the technical methodology, instruments, and technologies that guarantee the precision, effectiveness, and scalability of the system.

5.1 Proposed Methodology

The proposed approach describes how to use the YOLO11 model to detect objects in real-time. Among other things, the procedure entails gathering input data, preprocessing, inferring models, and visualizing the outcomes. Here is a thorough explanation supported by diagrams.

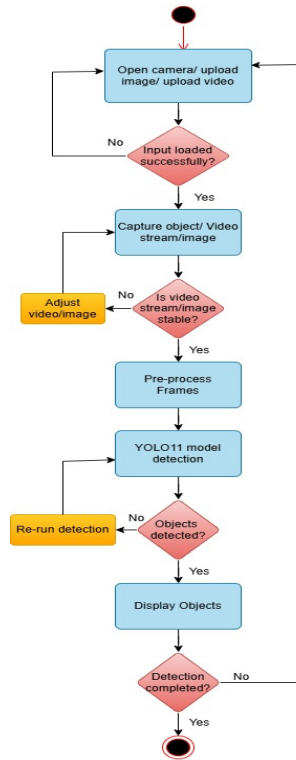


Figure 5.1: Activity Diagram for Real-time object detection using YOLO11

5.2 Description of Modules

1. Module Name: Input Handling

- **Input:** Camera feed, uploaded image, or video file.
- **Output:** Success or failure in loading the input.
- **Description:** This module ensures that the input source (camera, image, or video) is properly loaded for processing. If loading fails, it prompts the user to retry.

2. Module Name: Preprocessing

- **Input:** Stable video stream or image frames.
- **Output:** Preprocessed frames ready for object detection.
- **Description:** Frames are preprocessed to ensure they meet the YOLO11 model's input requirements, such as resizing, normalization, and format conversion.

3. Module Name: Object Detection

- **Input:** Preprocessed frames.
- **Output:** Detected objects or a re-run request.
- **Description:** The YOLO11 model processes the frames to detect objects. If no objects are detected, it retries detection with adjusted parameters.

4. Module Name: Results Display

- **Input:** Detected objects.
- **Output:** Displayed results on the interface.
- **Description:** This module visualizes the detected objects on the video or image, showing bounding boxes, labels, and confidence scores.

5. Module Name: Completion Check

- **Input:** Detection results.
- **Output:** Confirmation of detection completion.
- **Description:** Ensures that the detection process is successfully completed and loops back if needed.

Algorithm 3 Proposed methodology workflow**Require:** Input source (camera, image, or video).**Ensure:** Detected objects displayed on the output.

```

1: Open the camera or upload an image/video file.
2: if input is not loaded successfully then
3:   return Failure.
4: end if
5: Capture object from the video stream or image.
6: if video streaming is not stable then
7:   Adjust video/image settings.
8:   Line 3.
9: end if
10: Pre-process frames (resize, normalize, etc.).
11: Perform object detection using the YOLO11 model.
12: if no objects are detected then
13:   Re-run detection with adjusted parameters.
14:   Line 7.
15: end if
16: Display detected objects with bounding boxes and labels.
17: if detection is not completed then
18:   Line 7.
19: end if
20: return Success.

```

The suggested methodology's Algorithm 3 begins by opening and verifying the input source, which could be a camera, picture, or video. If the input is unsuccessful, the process ends. The uploaded material is processed and frames are captured, and the stability of the video stream is checked. Adjustments are made and the process resumes if it is unstable. For detection, preprocessed frames—which have been scaled and normalized—are fed into the YOLO11 model. The procedure is repeated if no items are found, and the detection parameters are changed. When detection is finished, the algorithm stops and the detected items are shown with labels and bounding boxes.

Chapter 6

TESTING

The testing approach tested the real-time object detection system's performance and functionality. It ensured precise object recognition in live webcam feeds, videos, and pictures. Many scenarios were investigated, including managing big files, extended usage, and improper inputs (e.g., corrupted files).

6.1 Test Cases

Table 6.1: Test Cases for Proposed System

TestCase ID & Name	TestCase Description	TestCase Steps	Test Data	Expected Output	Actual Output	Status
TC_01: Detect object-cam	Validate the functionality of gen_frames to produce frames from the webcam.	Click on the start webcam button.	Webcam hardware	The function should detect objects from the live camera feed.	It displays the detected objects.	Pass
TC_02: Detect object-cam	Validate gen_frames behavior when the camera feed cannot be started due to hardware or permission issues.	Disable the webcam or deny access permissions.	No webcam or permissions revoked	The function should raise an error indicating the camera feed cannot be accessed.	Error: Camera feed not accessible.	Fail

TestCase ID & Name	TestCase Description	TestCase Steps	Test Data	Expected Output	Actual Output	Status
TC_03: Detect image	Validate the functionality of the detect_image function to process an image and return detected objects.	Load the appropriate image format and click on the detect object button.	Test1-image.png	The uploaded image should detect objects.	It displays the detected objects.	Pass
TC_04: Detect image	Validate the behavior of detect_image when provided a video file instead of an image.	Load the appropriate image format and click on the detect object button.	Test1-video.mp4	The function should raise a validation error indicating the input is not an image file.	Error: Image file format is invalid.	Fail
TC_05: Detect video	Validate the functionality of generate_frames to produce frames from a video.	Load video and click on submit to upload and detect objects from the video.	test video.mp4	The function should detect objects from the uploaded video.	It displays the detected objects.	Pass
TC_06: Detect video	Validate generate_frames behavior when provided an image file instead of a video.	Load video and click on submit to upload and detect objects from the video.	test image.png	The function should raise a validation error indicating the input is not a valid video file.	Error: Unsupported input.	Fail

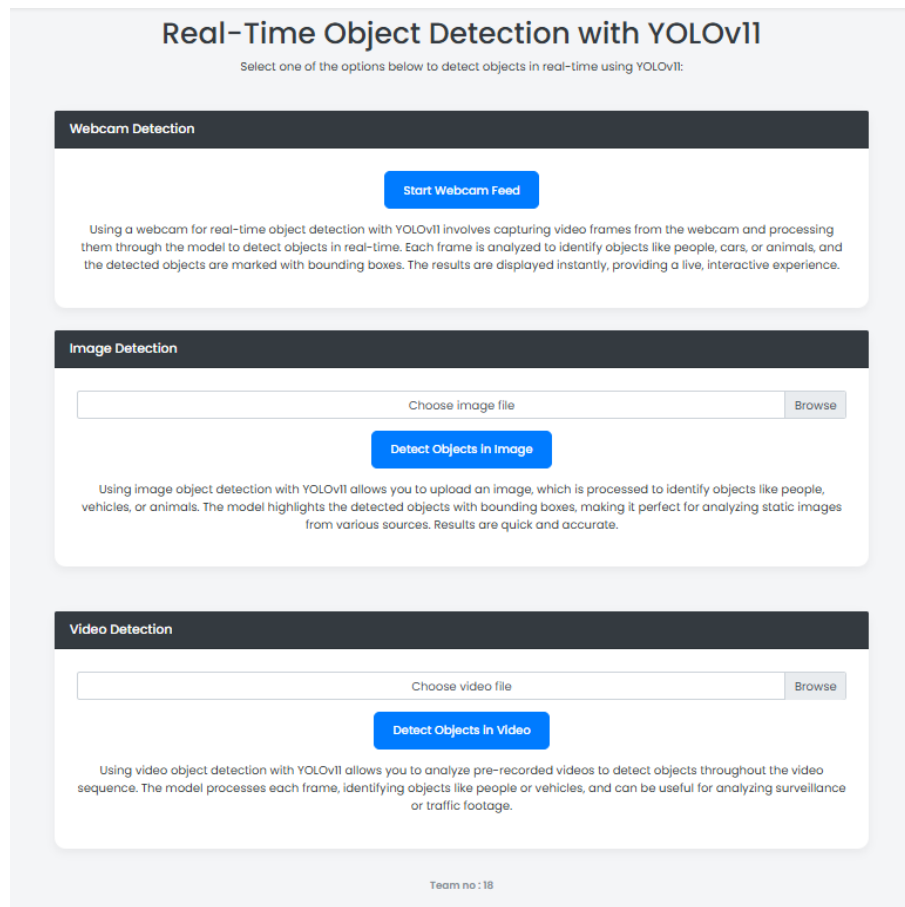
TestCase ID & Name	TestCase Description	TestCase Steps	Test Data	Expected Output	Actual Output	Status
TC_07: Detect corrupted image	Validate how detect_image handles an empty or corrupted image file.	Upload an empty or corrupted image file.	corrupted image.jpg	The function should raise an error indicating the image file is invalid or corrupted.	Error: Invalid or empty file.	Pass
TC_08: Detect image large	Validate detect_image functionality when processing a very large image file.	Upload a large image file and click on the detect object button.	large image.jpg	The function should process the large image and return detected objects.	It detects and displays the objects.	Pass
TC_09: Detect object cam	Validate gen_frames behavior when multiple webcams are connected.	Connect multiple webcams and start webcam detection.	Multiple webcam setup	The function should detect objects from the selected option.	Error: Camera feed not accessible.	Fail
TC_10: Detect object cam	Validate gen_frames performance for extended live detection.	Start webcam detection and let detection run for 2 hours.	Webcam hardware	The function should continue detecting objects without interruptions or resource issues.	It detects all objects successfully.	Pass

Chapter 7

RESULTS & DISCUSSIONS

The implementation of the real-time object detection algorithm using YOLO11 successfully demonstrated the ability to detect and identify objects in various input sources, including live camera feeds, uploaded images, and video files. The system accurately displayed bounding boxes, labels, and confidence scores for detected objects, showcasing the efficiency and robustness of the YOLO11 model.

1. Input Selection



The dashboard is titled "Real-Time Object Detection with YOLOv11" and includes a subtitle "Select one of the options below to detect objects in real-time using YOLOv11:". It features three main sections: "Webcam Detection", "Image Detection", and "Video Detection". Each section has a "Start Webcam Feed" button, a "Choose image file" input with a "Browse" button, and a "Detect Objects in Image" button. Below each button is a descriptive paragraph about the detection process. At the bottom, it says "Team no : 18".

Real-Time Object Detection with YOLOv11
Select one of the options below to detect objects in real-time using YOLOv11:

Webcam Detection

[Start Webcam Feed](#)

Using a webcam for real-time object detection with YOLOv11 involves capturing video frames from the webcam and processing them through the model to detect objects in real-time. Each frame is analyzed to identify objects like people, cars, or animals, and the detected objects are marked with bounding boxes. The results are displayed instantly, providing a live, interactive experience.

Image Detection

[Browse](#)

[Detect Objects in Image](#)

Using image object detection with YOLOv11 allows you to upload an image, which is processed to identify objects like people, vehicles, or animals. The model highlights the detected objects with bounding boxes, making it perfect for analyzing static images from various sources. Results are quick and accurate.

Video Detection

[Browse](#)

[Detect Objects in Video](#)

Using video object detection with YOLOv11 allows you to analyze pre-recorded videos to detect objects throughout the video sequence. The model processes each frame, identifying objects like people or vehicles, and can be useful for analyzing surveillance or traffic footage.

Team no : 18

Figure 7.1: Input Selection Dashboard

A user-friendly platform for YOLO11 real-time object recognition is provided by the input interface depicted in the Figure 7.1. To choose the preferred input method for object detec-

tion, it provides three primary functions.

By pressing the **"Start Webcam Feed"** button, users can start the Webcam Detection feature, which detects objects in real time. In order to recognize items like people, cars, or animals in real time, the system records video frames from the webcam and processes them using the YOLO11 model.

Users can submit a static image file using the Image Detection feature by choosing it using the "Choose image file" button. The **"Detect Objects in Image"** button allows users to examine the image after it has been uploaded. A single image from a variety of sources can be analyzed using the YOLO11 model.

The **"Choose video file"** button on the Video Detection feature lets users upload a recorded video file. When you select "Detect Objects in Video," the system systematically analyzes each video frame to find things in the video.

2. Object Detection

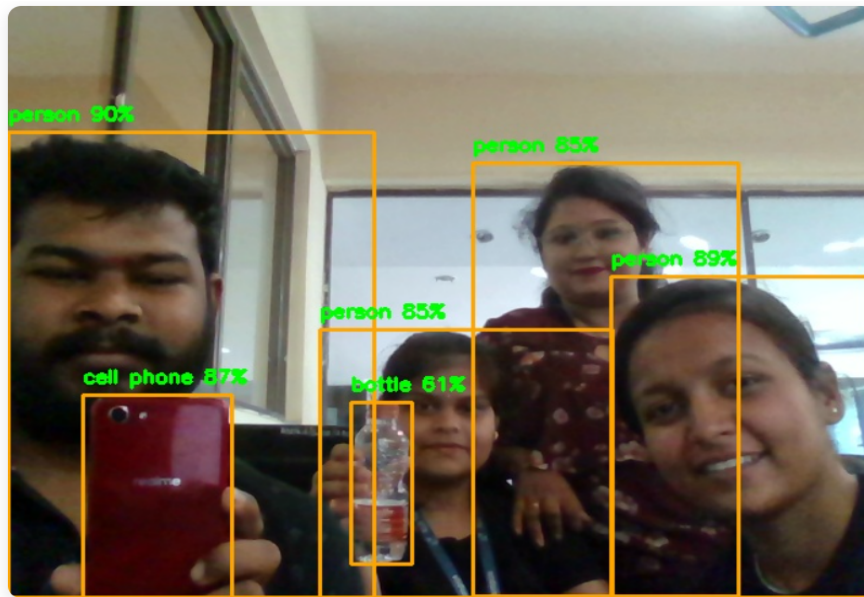


Figure 7.2: Object detection

The Figure 7.2 shows a webcam feed-displayed user interface with a real-time object detection system utilizing YOLO11. Several things in the frame are correctly identified and labeled by the system. Because each object is surrounded by a green bounding box, detection performance is clearly visible. A user-friendly design for real-time monitoring and engagement is suggested by the interface's buttons for starting and stopping the camera broadcast. The design principles of YOLO11, such as real-time processing, single-pass detection, high accuracy with efficiency, and versatility, are highlighted in this demonstration. These ideas make

the model appropriate for uses like interactive systems, autonomous driving, and surveillance that call for quick and precise object detection.

3. Image Detection

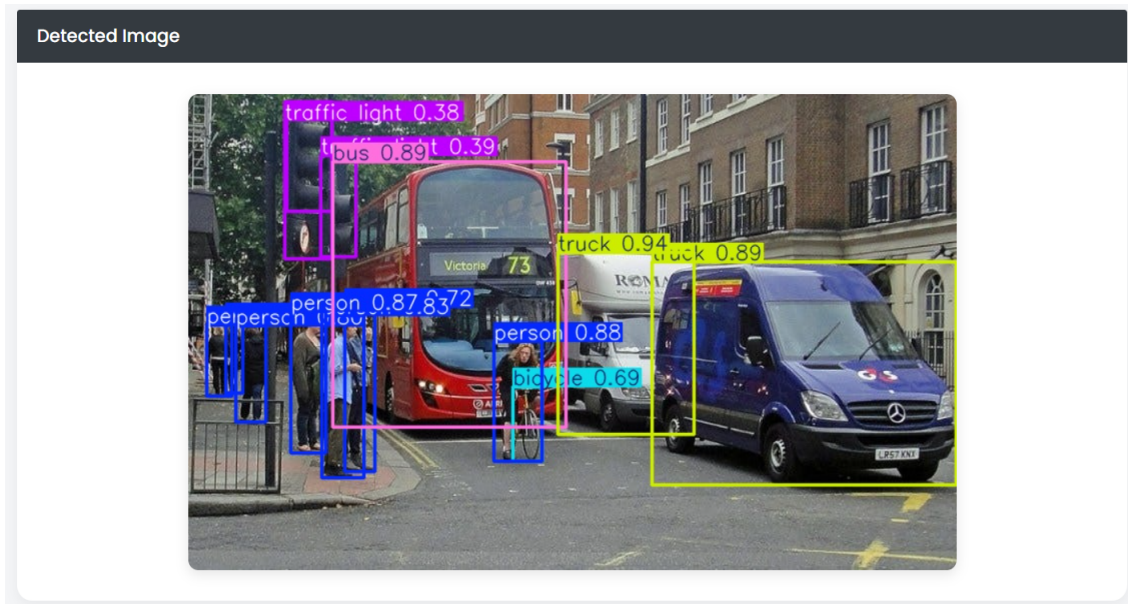


Figure 7.3: Image detection

The Figure 7.3 displays the YOLO11 Object Detection system's Image Detection features, which are intended to offer a user-friendly platform for identifying objects in images.

By choosing a picture using the "Browse" option, users can submit it to the system. The "Detect Objects in Image" button starts the object detection procedure after the picture has been uploaded. After processing the image, the YOLO11 model recognizes a variety of things, including "person," "dog," "car," and "suitcase." Each identified object is highlighted using a bounding box and labeled with its name and confidence score (e.g., "person 89%") in the "Detected Image" part of the results.

The detection results are easy for users to understand due to this graphic representation. The model's level of assurance in identifying each object is indicated by the confidence score, which promotes visibility. When examining static images from various sources, this function is especially helpful as it provides a rapid and precise solution for item detection and classification.

4. Video Detection

YOLO11 is used to identify things in a video feed in the Figure 7.4, which illustrates the Video Detection feature of an object detection system. In this particular case, the system is



Figure 7.4: Video detection

examining a scene on a highway in order to identify any automobiles.

Object Detection in Video Frames: To identify objects, the YOLO11 model progressively analyzes each video frame. Several autos, each surrounded by a bounding box, are identified in the frame that is shown.

Labels and Confidence Scores: The confidence score (such as "car 0.74", "truck 0.93") are annotated on each bounding box. Transparency and dependability in the results are ensured by the confidence score, which expresses the model's certainty about the detection.

Chapter 8

CONCLUSION AND FUTURE SCOPE

This project demonstrates a reliable real-time object identification system that integrates the YOLO11 model into a user friendly online interface. Real-time detection via a camera, object detection in images, and frame-by-frame video file analysis are the three main features offered by the system. Image analysis, traffic monitoring, and surveillance are just a few of the use cases that these characteristics guarantee flexibility and applicability for. An effective and smooth object detection pipeline is ensured by combining OpenCV for image and video processing with Flask for backend processing. The project demonstrates how the YOLO11 model can precisely identify items, label and are bounded by boxes, and provide users with real-time results.

The present implementation is successful, however it can be improved in a few ways to increase its scope. Adding advanced tracking algorithms to future work could allow multi-object tracking across video frames, enhancing the system's capacity to keep an eye on dynamic situations. To further maximize the real-time processing performance for high-resolution inputs, edge computing or GPU acceleration can be implemented.

REFERENCES

- [1] Gudala Lavanya and Sagar Dhanraj Pande. Enhancing real-time object detection with yolo algorithm. *EAI Endorsed Transactions on Internet of Things*, 10, 2024.
- [2] Momina Liaqat Ali and Zhou Zhang. The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection. *Computers*, 13(12):336, 2024.
- [3] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [4] Shengyu Lu, Beizhan Wang, Hongji Wang, Lihao Chen, Ma Linjian, and Xiaoyan Zhang. A real-time object detection algorithm for video. *Computers & Electrical Engineering*, 77:398–408, 2019.
- [5] Chien-Yao Wang, Hong-Yuan Mark Liao, et al. Yolov1 to yolov10: the fastest and most accurate real-time object detection systems. *APSIPA Transactions on Signal and Information Processing*, 13(1), 2024.
- [6] Satya Prakash Yadav, Muskan Jindal, Preeti Rani, Victor Hugo C de Albuquerque, Caio dos Santos Nascimento, and Manoj Kumar. An improved deep learning-based optimal object detection system from images. *Multimedia Tools and Applications*, 83(10):30045–30072, 2024.

Chapter 9

Plagiarism Report

ORIGINALITY REPORT			
17%	13%	7%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	Submitted to B.V. B College of Engineering and Technology, Hubli Student Paper	10%	
2	Submitted to Whitireia Community Polytechnic Student Paper	2%	
3	www.coursehero.com Internet Source	1%	
4	sportdocbox.com Internet Source	<1%	
5	ro.uow.edu.au Internet Source	<1%	
6	www.irjmets.com Internet Source	<1%	
7	Pradeep Singh, Balasubramanian Raman. "Deep Learning Through the Prism of Tensors", Springer Science and Business Media LLC, 2024 Publication	<1%	
arxiv.org			