VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi, Karnataka, INDIA

A
Project Report
on

"POTHOLE DIMENSION DETECTION"

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted By

2JH21CS008	**ADITYA KADIMDIWAN**
2JH21CS072	**PRATHAM KUBSAD**
2JH21CS075	**PUSHPA NAYAK**
2JH21CS085	**SAHANKUMAR SUBHASACHANDRA POL**

Under the Guidance of
Prof. PRAVEEN HONGAL
Assistant Professor, Dept. of AIML

Department of Computer Science and Engineering
JAIN COLLEGE OF ENGINEERING AND TECHNOLOGY
Sai Nagar, Hubballi – 580 031

2024– 2025

JAIN COLLEGE OF ENGINEERING AND TECHNOLOGY
Department of Computer Science and Engineering
Sai Nagar, Hubballi – 580 031

CERTIFICATE

Certified that the Project Entitled **"POTHOLE DIMENSION DETECTION"** carried out by **ADITYA KADIMDIWAN**, bearing **USN 2JH21CS008, PRATHAM KUBSAD**, bearing **USN 2JH21CS072, PUSHPA NAYAK, bearing USN 2JH21CS075, SAHANKUMAR SUBHASACHANDRA POL, bearing USN 2JH21CS085**, bonafide student of Jain College of Engineering and Technology, is in partial fulfillment for the award of the BACHELOR OF ENGINEERING in Computer Science and Engineering from **Visvesvaraya Technological University, Belagavi** during the year **2024-2025**. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the report submitted to the department library. The Project report has been approved as it satisfies the academic requirements in respect of the Project prescribed for the said Degree.

Prof. Praveen Hongal Assistant Professor Dept. of AIML JCET, Hubballi	Dr. Maheshkumar Patil, HOD, Dept. of CSE JCET, Hubballi	Dr. Prashanth Banakar Principal JCET, Hubballi

Name of the Examiners	Signature with date
1. _____	_____
2. _____	_____

JAIN COLLEGE OF ENGINEERING AND TECHNOLOGY
Department of Computer Science and Engineering
Sai Nagar, Hubballi – 580 031

DECLARATION

We, **ADITYA KADIMDIWAN**, bearing USN **2JH21CS008**, **PRATHAM KUBSAD**,bearing USN **2JH21CS072**, **PUSHPA NAYAK**, bearing USN **2JH21CS075**, **SAHANKUMAR SUBHASACHANDRA POL**, bearing USN **2JH21CS085** students of Eighth Semester B.E, Department of Computer Science and Engineering, Jain College of Engineering and Technology, Sai Nagar, Hubballi, declare that the Project Work entitled "**POTHOLE DIMENSION DETECTION"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering** in **Computer Science and Engineering** from **Visvesvaraya Technological University, Belagavi** during the academic year **2024-2025**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree.

2JH21CS008	**ADITYA KADIMDIWAN**
2JH21CS072	**PRATHAM KUBSAD**
2JH21CS075	**PUSHPA NAYAK**
2JH21CS085	**SAHANKUMAR SUBHASACHANDRA POL**

Place: Hubballi

Date:

ABSTRACT

In the modern era, road maintenance and urban safety have become critical concerns, especially with the rapid increase in vehicular movement. One of the most impactful applications of computer vision is automated pothole detection, which uses advanced object detection models to identify road surface defects. This project leverages YOLOv8, a state-of-the-art deep learning model, for real-time and offline pothole detection through images, videos, and live camera feeds (PC/mobile via DroidCam).

The system integrates image processing, real-time detection, dimension estimation, and GPS-based localization, making it suitable for large-scale deployment in smart city infrastructure. The model is trained on a custom dataset and utilizes a FastAPI backend for handling inference and a Streamlit frontend for an interactive user interface. Key features include automatic pothole detection with bounding boxes and size estimation, support for image/video uploads, and live camera detection, along with planned integration of GPS mapping and alert systems to notify drivers of upcoming potholes.

This solution can significantly reduce manual inspection efforts, improve road safety, and aid municipal authorities in proactive maintenance. Future improvements will further incorporate GPS-tagged pothole reports, user-contributed data, and real-time alerting systems based on location tracking.

ACKNOWLEDGEMENT

The satisfaction and the euphoria accompanying the successful completion of any task would be incomplete without the mention of the people who made it possible. The constant guidance of these persons and the encouragement provided crowned our efforts with success and glory. Although it is impossible to thank all the members who helped complete the project individually, we take this opportunity to express our gratitude to one and all.

We are grateful to the management and our institute **JAIN COLLEGE OF ENGINEERING AND TECHNOLOGY** for its very ideals and inspiration for having provided me with the facilities, that made this, work a success.

We express our sincere gratitude to **Dr. Prashanth Banakar,** Principal, of Jain College of Engineering and Technology for their support and encouragement.

We wish to place on record, my thanks to **Dr. Maheshkumar Patil**, HOD, Department of CSE, Jain College of Engineering and Technology, for the constant encouragement provided to me.

Guidelines and deadlines play a very important role in the successful completion of the project report on time. We convey our gratitude to **Prof. Trupti Thite,** Project Co-ordinator, Department of CSE, Jain College of Engineering and Technology, for her constant motivation in the development of project reports and setting up precise deadlines.

We are indebted with a deep sense of gratitude for the constant inspiration, encouragement, timely guidance, and valid suggestions given to us by our guide, **PRAVEEN HONGAL, Assistant Professor**, Department of AIML, Jain College of Engineering and Technology

We are thankful to all the teaching and non-teaching staff members of the department for providing relevant information and helping in different capacities in carrying out this project.

Last, but not least, we owe our debt to our parents, friends, and also those who directly or indirectly have helped us to make the project a success.

2JH21CS008 ADITYA KADIMDIWAN
2JH21CS072 PRATHAM KUBSAD
2JH21CS075 PUSHPA NAYAK
2JH21CS085 SAHANKUMAR SUBHASACHANDRA POL

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Machine Learning is a dynamic field of Artificial Intelligence (AI) that empowers computer systems to learn from experience and adapt over time without being explicitly programmed. The key objective of machine learning is to design intelligent systems capable of analyzing data and making predictions or decisions. This is especially valuable in real-world applications such as object detection, predictive maintenance, medical diagnostics, autonomous driving, and, in our context, automated road infrastructure analysis.

One pressing concern in modern urban and rural development is the detection and monitoring of potholes on roads. Manual road inspections are labor-intensive, inconsistent, and dangerous in some environments. This challenge presents a strong use case for leveraging computer vision and machine learning to automate pothole detection, quantify damage, and aid road maintenance authorities in timely interventions.

Machine learning offers a multitude of benefits, including automation of decision-making, scalability across large datasets, continuous improvement with more data, and the ability to uncover patterns not easily discernible by humans. In safety-critical applications like ours, ML reduces human error and increases the efficiency of road maintenance operations.

Additionally, ML models can be retrained and updated as new data becomes available, ensuring adaptability to changing environments. In urban infrastructure monitoring, this adaptability is key to handling evolving traffic conditions and road degradation patterns.

Despite its power, machine learning is not without limitations. Data quality, representativeness, and bias can severely affect model performance. Overfitting, interpretability, and resource intensive training are also major concerns. In real-world deployment, especially in embedded or edge devices, computational constraints necessitate lightweight yet accurate models.

The trajectory of machine learning is poised to redefine every aspect of society,from autonomous systems to personalized healthcare, climate modeling, and smart governance.

The integration of ML with IoT, edge computing, and 5G will unlock new possibilities for real-time analytics, especially in transport systems and smart cities

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its 0predictions. There are four basic approaches: supervised learning, unsupervised learning, semi- supervised learning and reinforcement learning. The type of algorithm data scientists choose to use dependson what type of data they want to predict.

Machine learning is split into the following categories:

1. Supervised Learning.

2. Semi-supervised Learning

3. Unsupervised Learning

4. Reinforcement Learning

1.Supervised Learning

Supervised learning involves training models on labeled data datasets where each input has a known corresponding output. The model learns to map inputs to outputs by minimizing the prediction error. Applications include spam detection, image classification, and predictive maintenance. This paradigm is highly effective when large, annotated datasets are available, and the objective is clear and measurable.

In our pothole detection system, supervised learning forms the backbone, where thousands of annotated images (with bounding boxes indicating potholes) are used to train the detection model. The model learns the visual characteristics of potholes and generalizes this to detect them in new images or videos.

2.Unsupervised Learning

Unsupervised learning deals with unlabeled data. The goal is to discover hidden patterns or intrinsic structures in the data without explicit supervision. Clustering and dimensionality reduction are common tasks. Techniques like K-Means, DBSCAN, and Principal Component Analysis (PCA) are widely used in customer segmentation, anomaly detection, and data

visualization.While unsupervised learning is not directly used in our system, it could be valuable in clustering road types or regions with similar pothole distributions to optimize repair operations.

3.Semi-Supervised Learning

This paradigm combines the strengths of supervised and unsupervised learning. It uses a small amount of labeled data along with a large quantity of unlabeled data. This approach is especially beneficial when labeling is expensive or time-consuming. Semi-supervised learning can significantly improve model performance without requiring exhaustive annotation.

For a pothole detection system deployed city-wide, where only a fraction of road data is labeled, semi-supervised learning could enhance detection performance by leveraging unlabeled dashcam footage.

4.Reinforcement Learning

Reinforcement learning (RL) is based on an agent interacting with an environment to maximize cumulative rewards. The agent learns through trial and error, using feedback signals. RL is widely applied in robotics, gaming, and autonomous control systems. In advanced pothole systems integrated with self-driving cars, RL could be employed to optimize routing algorithms to avoid poor road segments dynamically.

Machine Learning focuses on the question of how to get computers to program themselves (from experience plus some initial structure). Whereas Statistics KDD Pattern recognition Neuro computing AI Databases Machine learning Data Machine Learning has focused primarily on what conclusions can be inferred from data, Machine Learning incorporates

additional questions about what computational architectures and algorithms can be used to most effectively capture, store, index, retrieve and merge these data, how multiple learning subtasks can be orchestrated in a larger system, and questions of computational tractability.

Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular than ever. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable.

1.1 YOLO: You Only Look Once: A Real-Time Object Detection Framework

In the realm of computer vision, object detection has become one of the most challenging yet significant tasks, especially in real-time applications. Among the numerous object detection architectures developed over the years, the YOLO (You Only Look Once) framework has emerged as one of the most powerful and efficient solutions. YOLO revolutionized the approach to object detection by proposing a single-stage detector that treats detection as a regression problem, allowing for both classification and localization in one unified pass through a neural network. Its real-time capabilities, combined with high accuracy, make it particularly well-suited for applications such as autonomous driving, surveillance, traffic monitoring, and,most relevant to this project,pothole detection and road condition assessment.

1.1.1 Background and Motivation

Traditional object detection models relied heavily on region proposal methods, such as the twostage architecture of R-CNN and its variants (Fast R-CNN, Faster R-CNN). These models first generated candidate object regions and then classified them separately, often resulting in high computational complexity and slower inference times. YOLO broke away from this paradigm by treating object detection as a single regression problem that directly predicts bounding boxes and class probabilities from full images in a single evaluation. This design allows YOLO to be significantly faster than its predecessors, making it highly favourable for time-sensitive applications where both speed and accuracy are paramount.

1.1.2 Architecture of YOLO

The architecture of YOLO has evolved through multiple versions, YOLOv1 through YOLOv8, each bringing performance enhancements in speed, accuracy, and scalability. The core concept remains consistent: the image is divided into a grid, and each grid cell is responsible for predicting a fixed number

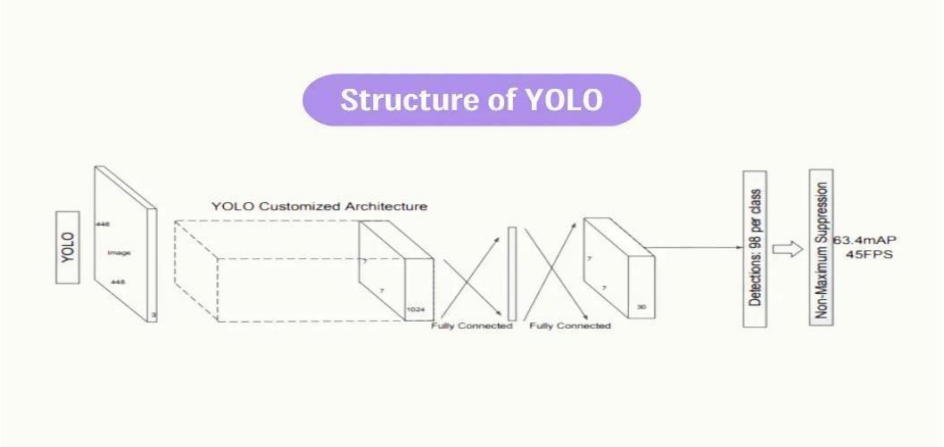


Fig: 1.1 Structure of YOLO

YOLOv1 pioneered the concept of using a single neural network for both object classification and localization.

YOLOv2 (YOLO9000) enhanced accuracy and introduced anchor boxes to better predict object dimensions.

Table 1.1: Comparison of object detection models.

Algorithm	mAP (COCO)	Speed (FPS)	Architecture Type
YOLOv8	~53.9	60+	One-stage
SSD	~45.5	22–46	One-stage
Faster R-CNN	~42.1	5–7	Two-stage

YOLOv3 incorporated deeper network layers and improved multi-scale prediction capabilities.

YOLOv4 and YOLOv5 emphasized model efficiency, aiming to reduce computational load while preserving high accuracy.

YOLOv7 brought optimized training strategies and auxiliary prediction heads, enhancing detection performance—especially for small objects. YOLOv8, the most recent major release, integrates Ultralytics' innovations like automatic learning rate selection, multi-task support (including detection, segmentation, and classification), and a native PyTorch implementation.

In our project, "Pothole Detection and Dimension Estimation using YOLOv8", we utilize stateof-theart object detection algorithms to identify potholes in images and videos, estimate their dimensions, and visualize results in both real-time and batch modes. The system also supports GPS integration, real-time alerts, and mobile camera compatibility via DroidCam, ensuring field usability.

We use the YOLOv8 (You Only Look Once) deep learning architecture, which is highly effective for real-time object detection tasks. YOLOv8 offers significant improvements over its predecessors in terms of speed, accuracy, and ease of integration, making it suitable for lowlatency, high-precision road analysis.

One of the most significant advancements in object detection is the YOLO (You Only Look Once) family of models. Unlike traditional methods that perform detection in multiple stages, YOLO treats detection as a single regression problem, making it extremely fast and efficient for real-time applications. YOLOv8, the version used in our system, brings improvements in accuracy and efficiency.

YOLO divides the image into grids and predicts bounding boxes and class probabilities directly, allowing for simultaneous detection and classification. In the context of our system, YOLOv8 has been fine-tuned on custom pothole datasets. It enables high-speed detection with bounding box overlays and real-time inference from live camera feeds,supporting both PC webcams and mobile cameras through integrations like DroidCam.

In pothole detection, false positives (e.g., misidentifying shadows as potholes) or false negatives (missed detections) could lead to ineffective maintenance decisions. Thus, continuous monitoring, evaluation, and retraining are critical for maintaining system robustness.

For pothole detection, the YOLO framework is an ideal choice due to its ability to detect multiple potholes in a single frame, even in challenging conditions such as poor lighting, occlusion, or varying road textures. In this project, YOLOv8 was utilized for detecting potholes in both static images and real-time video streams, enabling precise localization and dimension estimation. The model was trained on a custom dataset collected from real-world road environments and annotated accordingly.

YOLO's bounding box outputs were further used to estimate pothole dimensions, allowing integration of both detection and measurement within the same pipeline.

1.2 PREAMBLE

Welcome to a smarter approach to road safety. Our Real-Time Pothole Detection System leverages deep learning and computer vision to automatically detect potholes from images, videos, and live camera feeds. By combining speed and precision through the YOLOv8 model, the system provides instant visual feedback with estimated dimensions and sets the stage for GPS integration and early alerts.

This project represents a step toward intelligent infrastructure, making it easier for citizens and authorities to identify and respond to road damage in real time. Whether through a smartphone, webcam, or uploaded media, detecting potholes has never been this efficient or accessible.

1.3 PROBLEM STATEMENT

Implement a real-time pothole detection system that can accurately identify and locate potholes in images, video files, and live camera streams using deep learning. The system should utilize advanced object detection models like YOLOv8 and provide dimension estimates of detected potholes. It must be capable of processing both PC and mobile camera inputs using frameworks

like Streamlit and FastAPI. To further enhance the practicality, it should support output visualization, downloadable media, and pave the way for future integration with GPSbased alerts and mapping tools. The system must address challenges like real-time responsiveness, model accuracy, and diverse environmental conditions (lighting, angles, motion blur).

1.4 OBJECTIVES

To achieve the development of the proposed system, the following objectives are defined:

- Develop a model capable of detecting potholes using real-time data from images, videos, and live camera feeds.
- Integrate YOLOv8-based deep learning object detection with frontend and backend services for seamless user interaction.
- Estimate pothole dimensions in meters using pixel-to-real-world scaling techniques.
- Develop a web-based interface supporting image and video uploads, real-time detection, and visual display of results.
- Integrate dual-camera input, enabling support for both PC webcams and mobile feeds via DroidCam.
- Provide functionality for users to preview, download, and interact with processed media.
- Establish a foundation for future enhancements such as GPS-based pothole tagging, advance warning alerts within 100 meters, and database integration for pothole logging

1.5 SCOPE OF THE STUDY

The scope of study for the Real-Time Pothole Detection System includes the following key domains:

1.Deep Learning for Object Detection

Study and apply cutting-edge object detection architectures, particularly YOLOv8, to train a model capable of accurately detecting potholes in varied scenarios.

2.Image and Video Processing

Enable detection across multiple input formats, static images, recorded videos, and real-time streams. Handle preprocessing tasks and ensure compatibility across devices and resolutions.

3.Real-Time Application Development

Design a lightweight and responsive real-time application using FastAPI for backend inference and Streamlit for user interaction, integrating camera feeds and live preview functionality.

4.Computer Vision for Measurement Estimation

Use computer vision techniques to calculate pothole width and height in meters, providing visual overlays to enhance interpretability.

5.Frontend-Backend Communication

Ensure secure and efficient data flow between the web-based frontend and detection backend using REST APIs, file handling, and base64 encoding/decoding.

Scope of the Study

The scope of this project is centered around the development and deployment of a real-time pothole detection system that leverages machine learning techniques, specifically using the YOLOv8 object detection model. The system is designed to process inputs from various media sources — images, pre-recorded videos, and real-time camera streams — to identify and localize potholes effectively.

The study encompasses the integration of a FastAPI backend for processing and a Streamlit-based frontend for user interaction. It includes dimension estimation of potholes, support for both PC and mobile cameras, and a dynamic dashboard to visualize usage statistics and detection history. The project also logs each detection for later analysis and allows users to download session summaries and processed outputs.

While the system currently focuses on detection and basic dimension estimation, future extensions are scoped to include GPS-based localization, predictive analytics, alert generation, mapping integration, and enhanced support for variable road and lighting conditions.

CHAPTER 2

LITERATURE SURVEY

Pothole detection has garnered significant attention due to its implications on road safety and maintenance. Recent advancements integrate technologies like IoT, computer vision, and deep learning to enhance detection accuracy and real-time responsiveness. Below is an overview of pertinent studies in this domain

1. Author(s): Bhavan Kumar S B, Guhan S, Manyam Kishore, Santhosh

Year: 2023

Title of Paper: Real-time Pothole Detection using YOLOv5 Algorithm: A Feasible Approach for Intelligent Transportation Systems

Methodology Used:

The authors proposed a real-time pothole detection framework using the YOLOv5 deep learning algorithm. The model processes road images and videos to detect potholes with high accuracy and speed, suitable for deployment on edge devices such as onboard vehicle systems. The system includes data collection, annotation, model training, and conversion to OpenVino IR format for real-time detection.

Dataset/Tools Used:

The dataset used contains 850 real-world pothole images under varied lighting, weather, and motion conditions. The YOLOv5 model was trained and evaluated on this data. Tools included YOLOv5, OpenVINO, and image processing frameworks in Python.

Key Findings:

- YOLOv5 effectively detects potholes in real - time video streams with fast frame rates.
- The model is lightweight and compatible with low – power edge devices.
- Improved road safety and maintenance efficiency through automated alerts.

Limitations:

- Performance may drop under extreme visual noise like glare or heavy shadow.
- The algorithm still relies heavily on image quality and clarity.
- Non-pothole road defects (e.g., manholes, shadows) may result in false positives.

Future Work:

- Incorporate additional road condition features like surface elevation for better discrimination.
- Explore YOLOv8 or hybrid models for improved precision.
- Integrate the model with GPS and cloud systems for centralized reporting.

2. Author(s): T.C. Mahalingesh, Anubhav, Harshit Mishra, R.V. Arun, Anshuman Anand

Year: 2024

Title of Paper: Pothole Detection using Image Processing and Machine Learning

Methodology Used:

This paper introduces an automated pothole detection and filling system using the YOLOv8 object detection algorithm integrated with Raspberry Pi, PiCamera, and Arduino Mega. YOLOv8 was trained via transfer learning to detect potholes in real-time, with further verification using ultrasonic sensors for accurate depth assessment. Once confirmed, a pump system fills the pothole with a diluted cement mix.

Dataset/Tools Used:

A custom dataset built using images from Kaggle, Robo flow, and real-world captures; annotated using Robo flow and trained on Google Colab.

Key Findings:

- Achieved a mean Average Precision (mAP) of 0.82 and recall of 0.79
- Combined hardware and software pipeline for detection and autonomous filling

Limitations:

- Operates in stop-go motion due to processing delay on low-powered devices
- Filling mechanism limited to predefined material and may not adapt to varying road types
- System's real-time performance may degrade in high-speed scenarios

Future Work:

Expansion includes deploying on higher-performance hardware, optimizing pump systems for multimaterial filling, and integration with cloud services for real-time city-wide pothole databases.

3. Author(s): Dhwani Desai, Abhishek Soni, Dhruv Panchal, Sachin Gajjar

Year: 2019

Title of Paper: Design, Development and Testing of Automatic Pothole Detection and Alert System

Methodology Used:

This paper proposes a system combining **ultrasonic sensors, accelerometers, stereo cameras, and GPS** modules on a Raspberry Pi-based platform. It detects potholes and speed breakers in real-time, calculates their dimensions, and sends alerts using SMTP.

Dataset/Tools Used:

Real-world sensor readings and depth measurements from ultrasonic sensors, accelerometer feedback, and GPS coordinates.

Limitations:

- Performance may drop under extreme visual noise like glare or heavy shadow.
- The algorithm still relies heavily on image quality and clarity.
- Non-pothole road defects (e.g., manholes, shadows) may result in false positives.

Key Findings:

- Pothole detection accuracy of **90%**
- Successfully deployed on a two-wheeler with real-time alert notifications
- Cost efficiency noted, but not scalable for all weather and lighting conditions
- Image-based recognition is limited; detection depends on combined sensor output

Future Work:

- Scalability to national-level infrastructure; integration with government portals and repair ticketing systems for preventive maintenance.

4. Author(s): Mahesh H.L., Aishwarya N., Arun N., Pooja S., Pranitha P.

Year: 2022

Title of Paper: Design of a Real-time Detection System for Potholes and Bumps Using Deep Learning

Methodology Used:

The authors proposed a real-time detection system that employs deep learning, specifically a Convolutional Neural Network (CNN) architecture using the YOLOv5 model. This system processes video streams from a vehicle-mounted camera to detect road anomalies such as potholes and speed breakers. The model performs object detection in real time, drawing bounding boxes around identified hazards and classifying them accordingly. The detected information is then relayed visually to alert the driver, thereby enhancing road safety.

Dataset/Tools Used:

The dataset comprises images of potholes and speed breakers sourced from Kaggle and other publicly available platforms. These images represent varied lighting conditions and road environments. For implementation, the authors used Python programming along with OpenCV for image processing and YOLOv5 for model training and detection tasks

Limitations:

- Operates in stop-go motion due to processing delay on low-powered devices
- Filling mechanism limited to predefined material and may not adapt to varying road types
- System's real-time performance may degrade in high-speed scenarios

Key Findings:

- The proposed YOLOv5-based model successfully detects potholes and speed bumps in realtime.
- High levels of precision and recall were achieved during testing phases.
- The system is capable of generating immediate visual alerts, providing proactive warnings to drivers.

CHAPTER 3

SOFTWARE REQUIREMENTS SPEACIFICATIONS

3.1 FUNCTIONAL REQUIREMENTS

Functional requirements define the core features and behaviors of the Pothole Detection System from the user's point of view. These are the essential services the system must deliver for effective pothole identification, dimension estimation, visualization, and interaction. These functionalities are visible in the final application and are fundamental to fulfilling user expectations.

What is a Functional Requirement:

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used. Functional user requirements may be high-level statements of what the system should do but functional system requirements should also describe clearly about the system services in detail.

Functional Requirement Specifications:

The following are the key fields, which should be part of the functional requirements specifications document:

- Purpose of the Document
- Scope
- Business Processes
- Functional Requirements
- Data and Integration
- Security Requirements
- Data Migration & Conversion

3.2 Non-Functional Requirement

Non-Functional Requirements are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security,

and many more. Non- Functional Requirements address vital issues of quality for software systems. If NFRs not addressed properly, the results can include:

- Users, clients, and developers are unsatisfied.
- Inconsistent software. Time and cost overrun to fix the software which was prepared without keeping NFRs in mind.

Types of Non-Functional Requirement:

- Scalability
- Reliability
- Regulatory
- Maintainability
- Serviceability
- Utility
- Security
- Manageability
- Data integrity
- Capacity
- Regulatory
- Availability
- Usability
- Interoperability

These can be classified as:

- **Performance constraints:** Reliability, security, response time, etc.
- **Operating constraints:** These include physical constraints (size, weight), personnel availability, skill level considerations, system accessibility for maintenance, etc
- **Interface constraints:** These describe how the system is to interface with its environment, users, and other systems. For example, user interfaces and their qualities (e.g., user- friendliness).
- **Economic constraints:** Immediate and/or long-term costs
- **Lifecycle requirements**: Quality of the design: These measured in terms such as maintainability, enhance ability, portability.

Advantages of Non-Functional Requirement:

- They ensure the software system follows legal and adherence rules.
- They specify the quality attribute of the software.
- They ensure the reliability, availability, performance, and scalability of the software system
- They help in constructing the security policy of the software system.
- They ensure good user experience, ease of operating the software, and minimize the cost factor.

Disadvantages of Non-functional requirement:

- The nonfunctional requirement may affect the various high-level software subsystem.
- They generally increase the cost as they require special consideration during the software architecture/high-level design phase. It is difficult to change or alter non-functional requirements once you pass them to the architecture.

3.3 Hardware and Software Requirements

Hardware Requirements

Component	Specification / Description
Processor	Intel Core i5/i7 or AMD Ryzen 5/7 (Minimum Quad-Core)
RAM	Minimum 8 GB (16 GB recommended for optimal performance)
GPU	NVIDIA GPU with CUDA support (e.g., GTX 1660, RTX 3060 or higher)
Storage	Minimum 50 GB of free disk space
Camera	HD webcam or mobile camera (via DroidCam, USB, or IP camera)
Internet Connection	Required for model updates and API communication

Software Requirements

Software Component	Version / Description
Operating System	Windows 10/11, Ubuntu 20.04 or later, or macOS
Python	Version 3.8 or newer
Streamlit	Frontend UI framework (version ≥ 1.20.0)
FastAPI	Backend inference API framework
OpenCV	For image processing and camera input handling
PyTorch	Required for YOLOv8 model execution
Ultralytics	YOLOv8 model and utility tools (version ≥ 8.0.0)
Requests	For HTTP-based communication between frontend and backend
Pillow (PIL)	For image decoding, manipulation, and rendering
DroidCam App	Optional; enables mobile camera usage for real-time detection

CHAPTER 4

PROPOSED METHODOLOGY

4.1 EXISTING SYSTEM

The current pothole detection methods mainly rely on manual inspections or basic sensor data from vehicles. While some systems use classical image processing techniques or simple machine learning models, they lack accuracy, especially in real-world conditions like poor lighting or cluttered roads. These approaches are often not real-time, do not estimate pothole dimensions, and are not integrated with GPS or mobile camera support. Existing systems also fail to provide visual alerts or user-friendly interfaces for quick response.

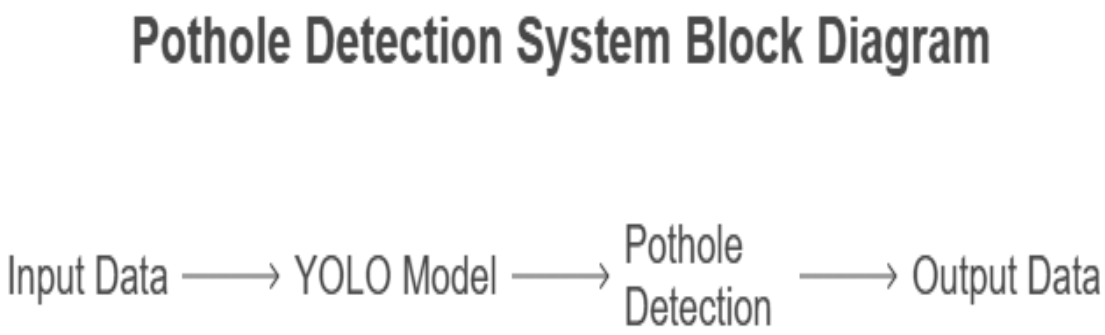


Fig 4.1: Existing Pothole Detection System.

4.1.1 Challenges in Existing Methodology

- Existing pothole detection systems often lack real-time capabilities, limiting their effectiveness in dynamic environments like moving vehicles.
- Many models struggle with inconsistent lighting, blurred motion, or low-resolution footage, reducing detection accuracy on real-world roads.
- Most conventional solutions focus only on detection without estimating pothole dimensions, which is crucial for prioritization and severity analysis.

4.2 PROPOSED SYSTEM

The proposed system is designed to detect potholes in real-time using deep learning techniques. It integrates the YOLOv8 object detection model to identify potholes from various input sources such as images, videos, and real-time camera feeds. For enhanced user experience, the system includes GPS-based location tagging and provides visual overlays of pothole dimensions on the detected output. The system supports mobile and desktop interfaces, enabling users to upload files or stream live footage. It also includes a notification system to alert users of upcoming potholes based on geolocation. This comprehensive setup enhances road safety and supports city authorities in monitoring infrastructure conditions efficiently.

4.3 PRE-PROCESSING

The proposed pothole detection system involves several key pre-processing steps to enhance detection accuracy and model performance. Initially, image augmentation techniques such as rotation, scaling, and brightness adjustment are applied to diversify the training data and improve generalization. These augmented images are then processed through the YOLOv8compatible pipeline, where visual features are extracted using convolutional layers. Before training, all images are resized to a standard dimension (e.g., 640×640), normalized, and annotated in YOLO format. Label cleaning and format validation ensure consistency in annotation. This structured pre-processing ensures robust input to the detection model, enabling it to accurately identify potholes across varying lighting and environmental conditions.

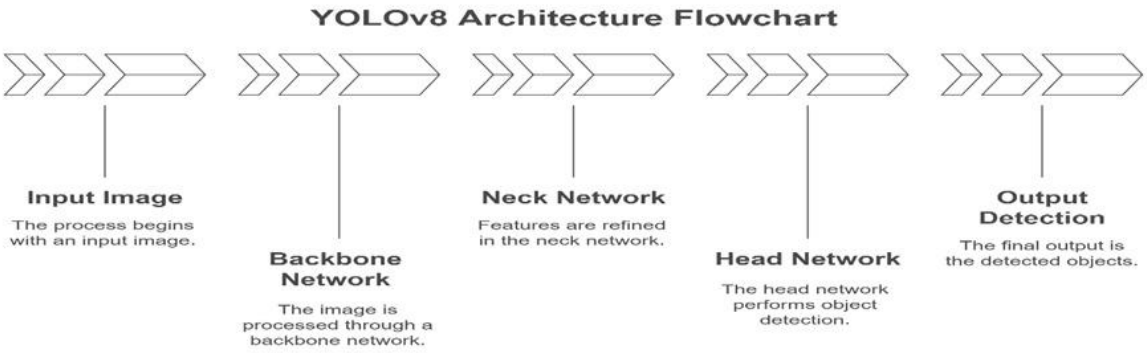


Fig:4.3 YOLOv8 Model

CHAPTER 5

SYSTEM ARCHITECTURE

Design of the system

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

5.1 Data Flow Diagram

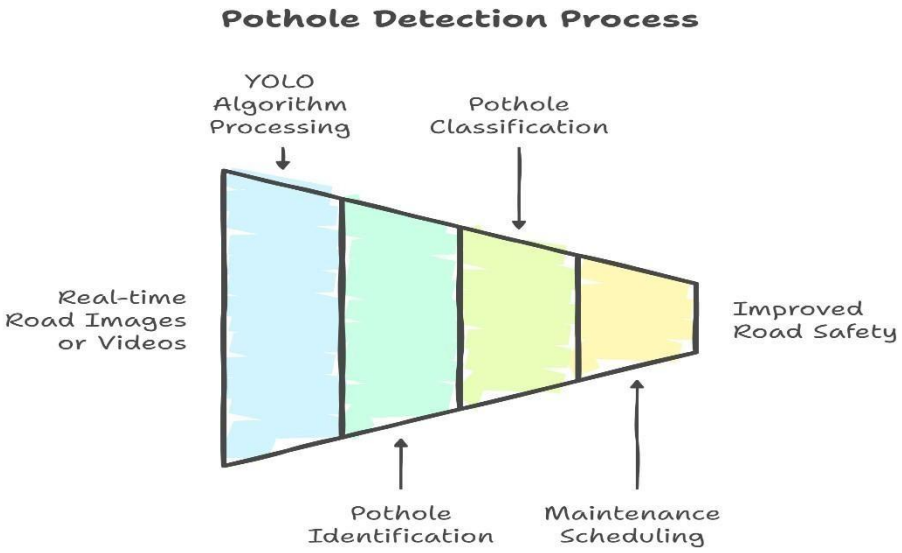


Fig 5.1: System Design

The 3 basic steps that are necessary to build this system are

1. Data Collection and Preprocessing.
2. Model Training Using YOLO
3. Real-time and file Detection and Deployment.

1.Data Collection and Preprocessing.

To build a robust pothole detection model, the first step involves collecting a diverse dataset of road images containing potholes under different lighting, road types, and weather conditions.

Each image must be annotated with bounding boxes around potholes, and optionally include dimensionrelated metadata (e.g., depth or area classes). Tools like Roboflow, CVAT,orLabelImg are used for this step. The dataset is then split into training, validation, and testing sets.

2.Model Development.

Model development for our pothole detection system involves training a deep learning model — specifically the YOLOv8 (You Only Look Once) object detection architecture. Unlike conventional captioning systems that rely on Convolutional Neural Networks (CNNs) and LSTMs for describing images in words, our system focuses on recognizing and detecting potholes and estimating their physical dimensions from images or video streams.

During training, the model is fed a large dataset of labeled road images containing potholes marked with bounding boxes. YOLO's neural architecture learns to identify the shape, size, and spatial features of potholes. The network refines its internal parameters through backpropagation, optimizing how it predicts bounding boxes and classifies objects (potholes in this case).

Backbone

- **Purpose:** Extracts basic visual features from the input image (like edges, textures, and shapes).
- **Typical Components:**
- Convolutional layers (C)
- Bottleneck blocks (B)
- Often uses architectures like CSPDarknet or MobileNet variants.
- Output: A set of feature maps at different scales and levels of abstraction.
- Purpose: Enhances and combines these features to help the model understand context and scale variance (e.g., small and large objects).
- Typical Components: o FPN (Feature Pyramid Network) o PAN (Path Aggregation Network) o In YOLOv8, often uses SPPF (Spatial Pyramid Pooling – Fast)

3.Real-time and file Detection and Deployment.

Once trained, the model is deployed into a real-time system. Using a FastAPI backend and Streamlit frontend, users can upload images or videos, or stream live feed (from a PC or mobile camera via DroidCam). The YOLO model detects potholes in real-time, overlays bounding boxes, and displays estimated dimensions. The system is also designed to be scalable and extendable with GPS tagging, voice alerts, and mobile accessibility.

4. Working

The pothole detection system is built on the YOLOv8 deep learning architecture, designed to perform real-time object detection with high accuracy and speed. The core idea is to detect potholes from images or videos and estimate their dimensions for alerting or logging purposes. The system follows a pipeline that processes input data, detects potholes using trained weights, and visualizes the results in a user-friendly format.

5.1.4 Bounding Box Coordinates Prediction

YOLOv8 predicts bounding boxes using this formula for each object:

$$\boldsymbol{Boxpred} = (\boldsymbol{xc}, \boldsymbol{yc}, \boldsymbol{w}, \boldsymbol{h}) \quad \textbf{(5.1.1)}$$

Where:

- xc,yc: center of the predicted box (relative to the grid cell or image)
- w,h: width and height of the box (in normalized image units)

In Our Project:

- You receive YOLOv8 bounding boxes as x1, y1, x2, y2 which are converted from (xc,yc,w,h).
- These are used to draw boxes around potholes and compute dimensions.

5.4.2 Dimension Calculation From Bounding Boxes

After converting bounding boxes to pixel values:

$$\boldsymbol{Pixel\ Width} = x2 - x1 \boldsymbol{\ and\ Pixel\ Height} = y2 - y1 \quad \textbf{(5.1.2)}$$

Then

using pixel-to-meter ratio (empirical):

$$\boldsymbol{Width}\ (m) = (x2 - x1) \times \boldsymbol{Pixel - to - Meter\ Ratio} \quad \textbf{(5.1.3)}$$

$$\boldsymbol{Height}\ (m) = (y2 - y1) \times \boldsymbol{Pixel - to - Meter\ Ratio} \quad \textbf{(5.1.4)}$$

.

5.2 USE CASE DIAGRAM

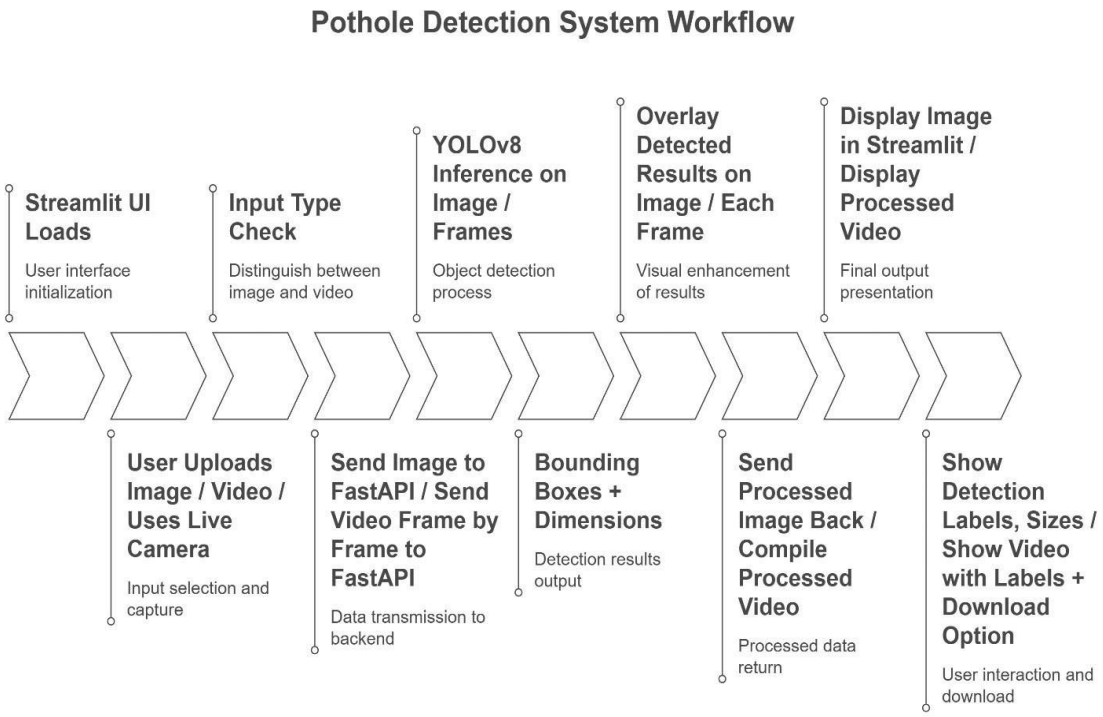


Fig 5.2: Use Case Block Diagram

5.3 DETAILED OVERVIEW OF THE FLOWCHART

1. **Input Acquisition:**

   The user uploads an image, a video, or starts a live camera stream (PC or mobile via DroidCam) through a Streamlit-based interface.

2. **Pre-processing:**

   The uploaded frame is resized and normalized as per the input requirements of the YOLOv8 model. For real-time streams, frames are read one at a time.

3. **Inference with YOLOv8:**

   The processed input is fed into the trained YOLOv8 model. The model quickly scans the image using a single forward pass to predict bounding boxes around potholes.

4. **Pothole Dimension Estimation:**

   For each detected pothole, the model calculates its width and height in pixels. Using a calibrated pixel-to-meter ratio (e.g., 1 pixel = 1/320 meter), it converts these measurements into real-world dimensions.

5. **Post-processing and Visualization:**

   The results — including bounding boxes and size annotations — are drawn directly on the image or video frame using OpenCV. These are returned to the frontend for user display.

6. **Frontend Display:**

   - Images: Detected potholes with dimensions are shown alongside the original upload.
   - Videos / Streams: The video is saved with detections and played back in the Streamlit app.
   - Download Option: The processed video can be downloaded by the user with a button click.

7. **Real-time Capability:**

The system supports live video input from both PC webcams and mobile cameras using DroidCam. YOLO's speed ensures minimal latency in detection.

5.4 Unified Modeling Language (UML)

Unified Modeling Language (UML) is a standardized visual language used to model the architecture and design of a software system. For the **Pothole Dimension Detection System using YOLOv8**, UML diagrams help in visualizing the interaction between users, modules, and components within the application.

The following diagrams have been used to represent various aspects of the system:

- **Use Case Diagram**: Depicts how end users (such as road surveyors or municipal workers) interact with the system by uploading media files, viewing detection results, and downloading reports.
- **Component Diagram**: Outlines the major components of the system, including the Streamlitbased frontend, FastAPI backend, YOLOv8 model, and the database or output modules.
- **Activity Diagram**: Illustrates the workflow of pothole detection, from input submission to final result rendering and optional GPS tagging or alerting.

CHAPTER 6

APPLICATION TESTING

Application testing refers to the process of testing any software application using scripts, tools, or any test automation frameworks in order to identify errors. It helps teams release bug-free and robust software applications into the real world. It also enables teams to identify bugs in the early stages of development and save development time.

6.1 Unit testing

Unit testing in this project refers to the isolated testing of individual components such as the detection functions, video frame processing routines, API endpoints, and frontend interaction handlers. The goal is to validate that each module behaves as expected when provided with known inputs. For example, we independently test:

YOLOv8 detection functions: Whether bounding boxes and dimensions are correctly returned for a given input image.

FastAPI endpoints: Whether the /detect route correctly processes image and video formats and returns appropriate responses.

Streamlit UI functions: Whether uploaded media previews correctly, and whether API responses are properly rendered. and its .Net counterpart, X Unit. Suitable parameters for the unit tests may be supplied manually or in some cases are automatically generated by the test framework. In recent years support was added

6.2 Integration Testing

Integration testing ensures that multiple system components interact correctly.

For this system, integration was tested between:

- The Streamlit frontend and FastAPI backend (file transmission and response handling)

- The YOLOv8 model and the FastAPI API, verifying if inference runs correctly and results are returned in the expected format (base64 string for images and binary stream for videos)

Example Integration Test:

- Upload a valid .mp4 video in Streamlit
- FastAPI returns processed video stream
- Streamlit correctly displays it and offers a download option

6.3 Functional Testing

Functional testing ensures the application performs all its intended operations:

Feature Tested	Test Steps	Expected Result
Image Upload	Upload .jpg or .png file	Image previewed, processed, and potholes displayed with annotations
Video Upload	Upload .mp4 file	Video previewed, each frame processed with pothole detection
Pothole Detection	Send image/video to backend	Bounding boxes and dimension labels displayed on detected potholes
Real-Time Detection	Activate camera	Live video stream with real-time detection and bounding boxes
Download Processed Output	Click "Download" button	Processed image/video downloaded successfully

Table 6.3: Functional Testing

CHAPTER 7

INTERPRETATION OF RESULTS

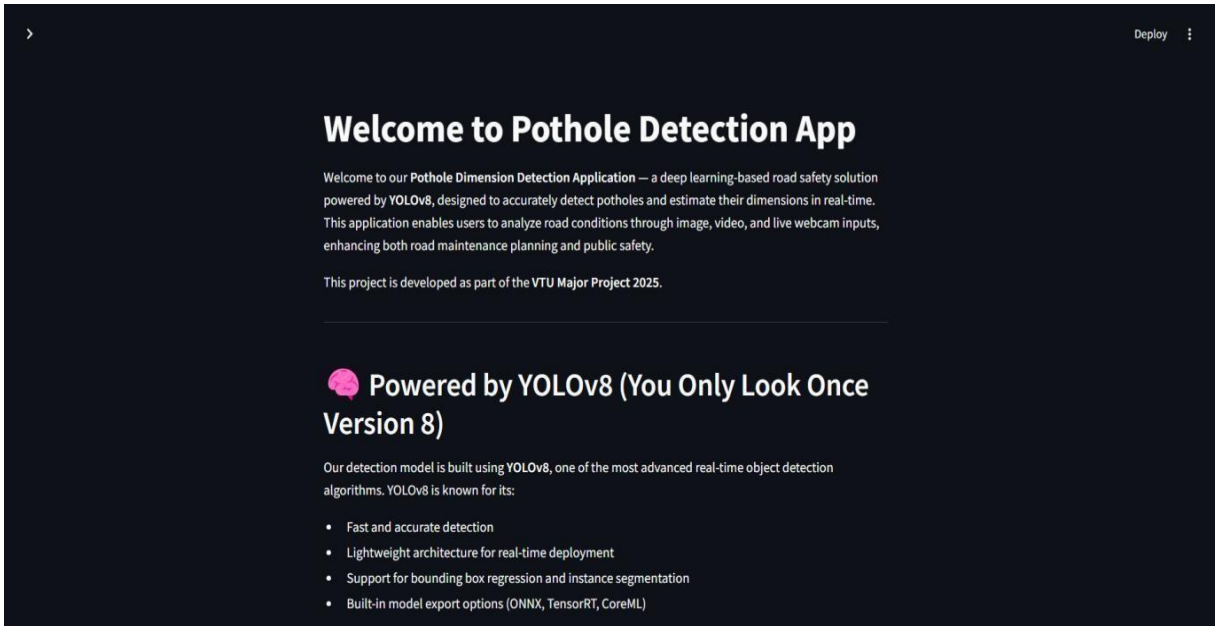


Fig7.1.1.: Home page

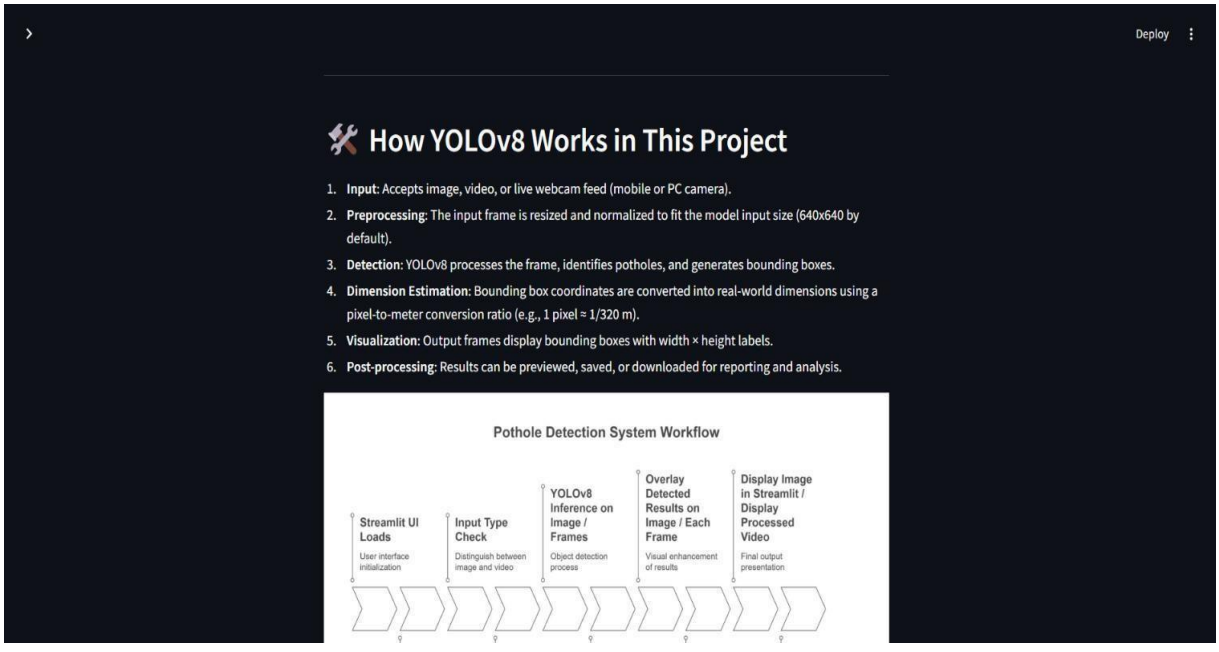


Fig7.1.2: About Section

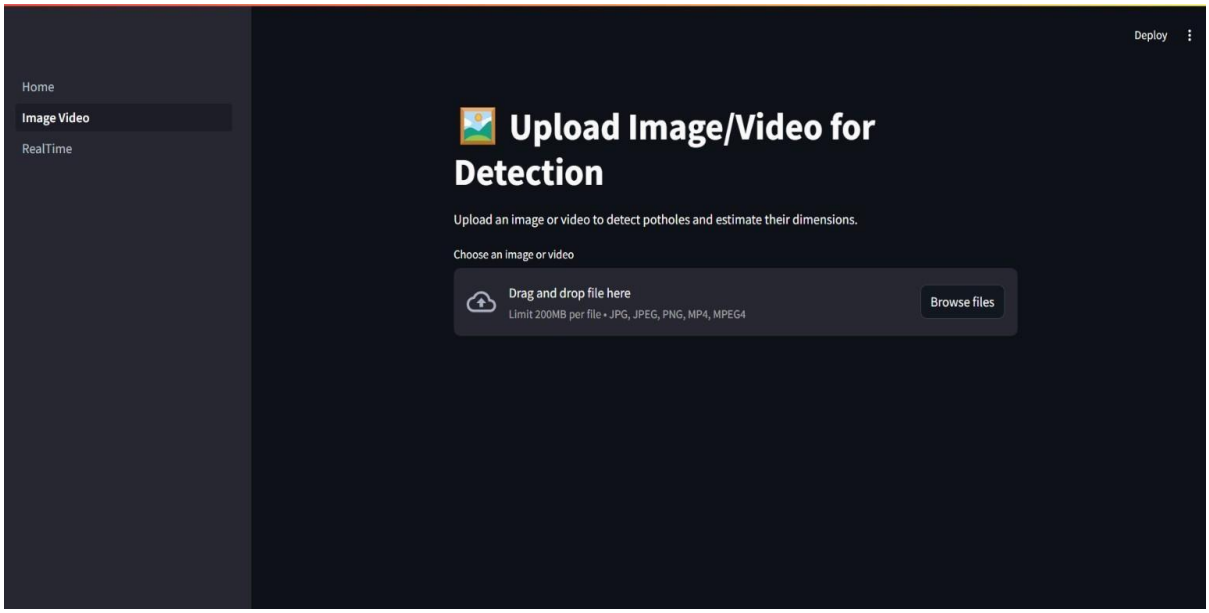


Fig7.1.3: Image and Video Input

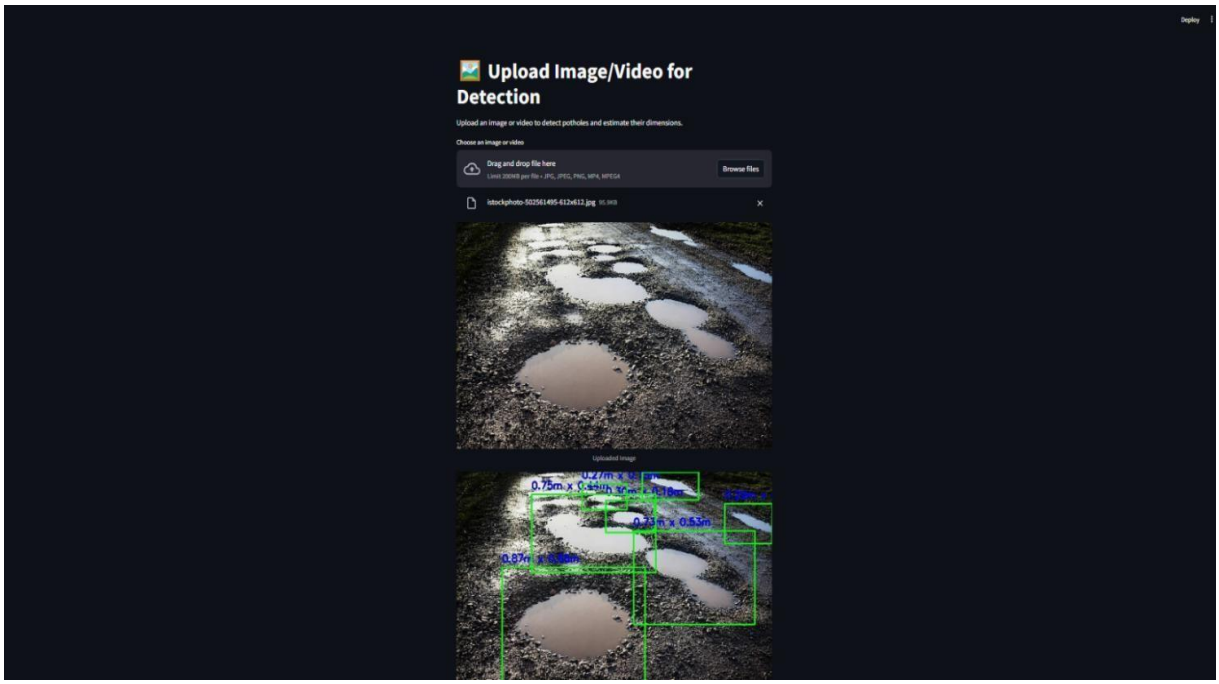


Fig7.1.4: Image Pothole Detected

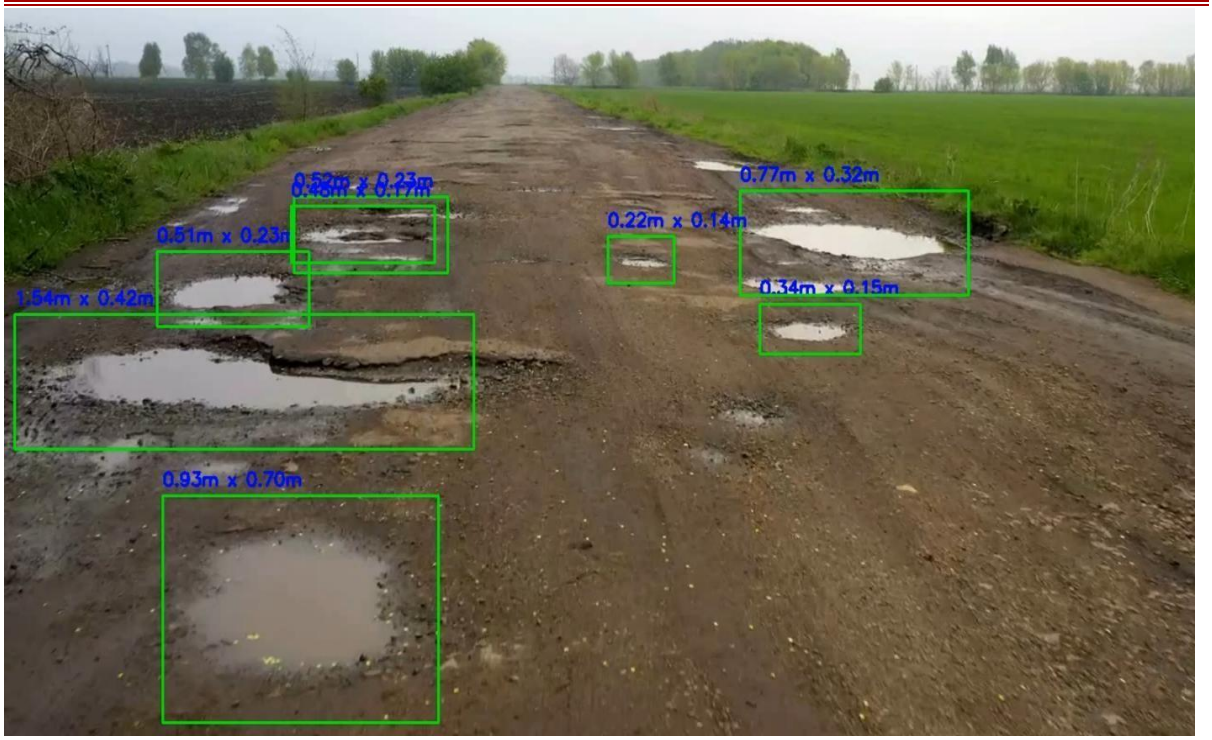
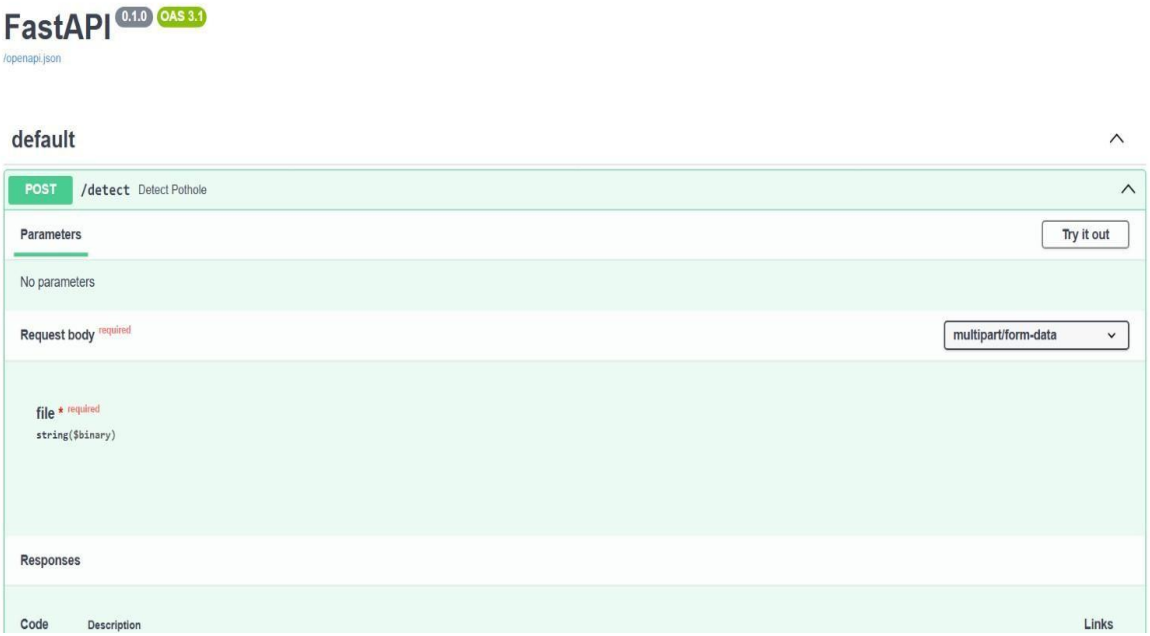
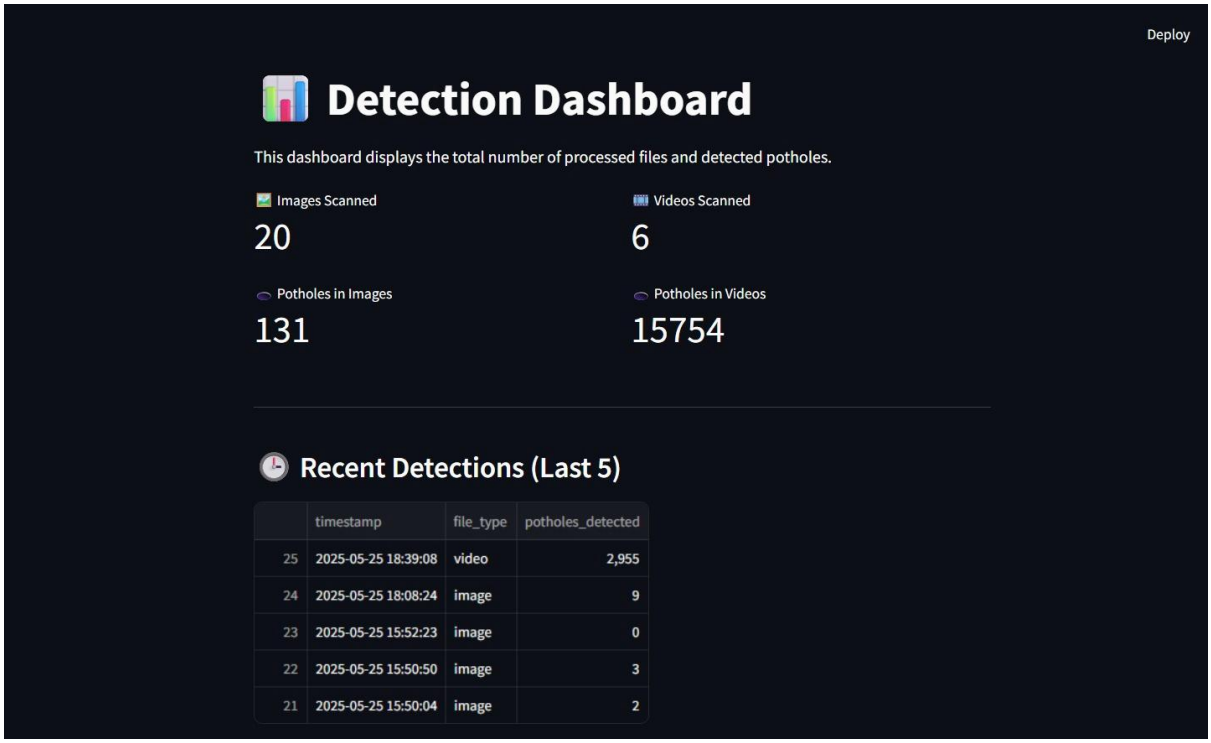


Fig7.1.5: Video Output



7.1.6: Backend Server



7.1.7: Detection Dashboard

7.1.8: Detection Trends

Table 7.1.9: Dataset Statistics

Dataset Name	Size		
	Train	Valid	Test
Custom Pothole Dataset	1275	120	405
CRDDC2022 Dataset	800	----	200

Table 7.2.1. Comparison between original and detected Output

Input	Original description	Predicted description
Pothole Image	**A pothole visible in the center of the road under daylight conditions**	**1.** Pothole detected in the middle of the lane. **2.** Visible road damage captured from a still image. **3.** Bounding box marked with dimensions: 0.45m × 0.60m. **4.** Clear surface crack identified as pothole. **5.** Model labels pothole with green box.
Pothole Video	**Multiple potholes appear along a narrow lane as the vehicle moves.**	1. Sequential detection of potholes in moving frames. 2. Bounding boxes dynamically overlaid on each pothole. 3. Real-time dimension estimation for every frame. 4. Frame-by-frame annotations for 4 potholes. 5. Width and height vary across frames with speed.

Real-Time	**Road captured using a mobile camera mounted on the dashboard shows a pothole near the left edge.**	1. Pothole detected near lane edge in live feed. 2. Detection triggered within 0.5 seconds of frame entry. 3. Bounding box shows: 0.38m × 0.52m. 4. Visual alert rendered in overlay. Responsive to different angles and lighting.

CHAPTER 8

APPLICATIONS

1. **Smart City Infrastructure Management**
    - Automates the process of identifying and measuring potholes on urban roads.
    - Enables municipalities to create real-time road health maps for maintenance planning.
2. **Road Maintenance Prioritization**
    - Helps civic authorities prioritize repairs based on pothole dimensions and frequency.
    - Reduces manual inspections by providing accurate digital logs.
3. **Driver Safety and Navigation Assistance**
    - Can be integrated with in-vehicle camera systems to alert drivers of upcoming potholes.
    - Improves vehicle safety by minimizing unexpected jolts or accidents.
4. **Traffic Management and Rerouting**
    - Helps traffic systems reroute vehicles away from severely damaged roads.
    - Can be used in dynamic traffic control systems and navigation apps.
5. **Data Collection for Civil Engineering**
    - Collects structured data for analyzing road wear patterns over time.
    - Assists in pavement design improvements and infrastructure resilience studies.
6. **Integration with GPS and IoT Systems (Future Scope)**

- Combined with GPS, pothole coordinates can be uploaded to centralized servers.

Helps create predictive models for future pothole formation using environmental data.

CONCLUSION

The Pothole Dimension Detection system developed in this project presents a practical and intelligent solution for automated road damage monitoring using deep learning. Leveraging the YOLOv8 object detection model, the system is capable of accurately identifying potholes and estimating their real-world dimensions across various input modes—image, video, and realtime camera feeds. This multi-input flexibility makes the system suitable for both offline analysis and live road surveillance.

By integrating features like pixel-to-meter conversion for dimension estimation and planning for future enhancements such as GPS tagging and early warning alerts, the application addresses key challenges in road maintenance automation. The system demonstrates strong potential in supporting municipal authorities, civil engineers, and transportation departments in their efforts to maintain safer roads and plan timely repairs. Overall, this project serves as a solid foundation for expanding AI-driven infrastructure monitoring in smart city environments.

FUTURE SCOPE

The pothole detection system developed in this project lays the groundwork for a comprehensive smart road monitoring solution. Future advancements can include integration with GPS modules to precisely geotag detected potholes, enabling authorities to map and prioritize road repairs. Real-time alerts could be sent to drivers via connected vehicle systems or mobile apps, warning them about upcoming potholes to improve road safety.

Further enhancements may involve training the YOLOv8 model on larger and more diverse datasets to improve detection accuracy under varied lighting, weather, and road conditions. Edge computing capabilities can be explored to allow processing directly on smart cameras or mobile devices, reducing latency and server dependency. The system can also be integrated with municipal road maintenance systems, triggering automated maintenance requests based on pothole severity and frequency.

REFERENCES

1. B. K. S B, G. S, M. Kishore, S. R and A. D. J, "Real-time Pothole Detection using YOLOv5 Algorithm: A Feasible Approach for Intelligent Transportation Systems," 2023
2. T. C. Mahalingesh, Anubhav, H. Mishra, R. V. Arun and A. Anand, "Pothole Detection and Filling System using Image processing and Machine Learning,"
3. D. Desai, A. Soni, D. Panchal and S. Gajjar, "Design, Development and Testing of Automatic Pothole Detection and Alert System," 2019
4. M. Nalawi, M. Baghdadi, B. Alyateem, Z. S. Abo-Hammour and A. Alsharkawi, "Design of a Real-time Detection System for Potholes and Bumps Using Deep Learning," 2024
5. D. Jyothirmai, N. S. Karthikeya, P. Jagadeesh, S. S. C. Reddy, S. D. Reddy and R. Pitchai, "Pothole Detection and Enhanced Road Safety Using Machine Learning," 2024
6. P. A. Chitale, K. Y. Kekre, H. R. Shenai, R. Karani and J. P. Gala, "Pothole Detection and Dimension Estimation System using Deep Learning (YOLO) and Image Processing," 2020
7. H. Widodo *et al*., "Experimental Evaluation of Pothole Detection and Its Dimension Estimation Using YOLOv8 and Depth Camera for Road Surface Analysis," 2024