A

MINI PROJECT REPORT

on

AUTOMATED ROAD DAMAGE DETECTION USING UAV IMAGES AND DEEP LEARNING TECHNIQUES

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

By

G. JAYACHANDRA	**:**	**22Q91A7217**
K. YUVARAJ	**:**	**22Q91A7226**
P. NEHA REDDY	**:**	**22Q91A7239**
S. ROHITH ADITHYA	**:**	**22Q91A7250**

Under the guidance of

Mrs. T. Naga Praveena

Assistant Professor, Dept. of AI&DS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE-Permanently Affiliated to JNTU-Hyderabad)

Accredited by NBA , Recognized section 2(f) & 12(B) of UGC New Delhi

ISO 9001: 2015 Certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100

2024 - 2025

MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE-Permanently Affiliated to JNTU-Hyderabad)

Accredited by NBA , Recognized section 2(f) & 12(B) of UGC New Delhi

ISO 9001:2015 Certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

CERTIFICATE

This is to certify that the Mini Project report on **"Automated Road Damage Detection Using UAV Images and Deep Learning Techniques"** successfully done by the following students of Department of **Artificial Intelligence & Data Science** of our college in partial fulfilment of the requirement for the award of Bachelor of Technology degree in the year 2024-2025. The results embodied in this report have not been submitted to any other University for the award of any diploma or degree.

G. JAYACHANDRA	**:**	**22Q91A7217**
K. YUVARAJ	**:**	**22Q91A7226**
P. NEHA REDDY	**:**	**22Q91A7239**
S. ROHITH ADITHYA	**:**	**22Q91A7250**

Submitted for the viva voice examination held on :

INTERNAL GUIDE	PROJECT COORDINATOR	HOD
Mrs. T. Naga Praveena	**Mrs. D. Kranthi Deep**	**Dr. J. Britto**
Assistant Professor	**Assistant Professor**	**Professor & Head**

Internal Examiner **External Examiner**

DECLARATION

We, **G. Jayachandra, K. Yuvaraj, P. Neha Reddy, S. Rohith Adithya** with Regd.no. **22Q91A7217, 22Q91A7226, 22Q91A7239, 22Q91A7250** are hereby declaring that the mini project entitled "**AUTOMATED ROAD DAMAGE DETECTION USING UAV IMAGES AND DEEP LEARNING TECHNIQUES"** has done by us under the guidance of Mrs. T. Naga Praveena, Assistant Professor, Department of Artificial Intelligence & Data Science is submitted in the partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY in ARTIFICIAL INTELLIGENCE & DATA SCIENCE.** The Result embedded in this project report have not been submitted to any University or institute for the award of any degree

Signatures of the Candidate

G. JAYACHANDRA : 22Q91A7217

K. YUVARAJ : 22Q91A7226

P. NEHA REDDY : 22Q91A7239

S. ROHITH ADITHYA : 22Q91A7250

DATE :

PLACE :Maisammaguda

ACKNOWLEDGEMENT

First and foremost, we would like to express our immense gratitude towards our institution Malla Reddy College of Engineering, which helped us to attain profound technical skills in the field of Computer Science & Engineering, there by fulfilling our most cherished goal.

We are pleased to thank **Sri Ch. Malla Reddy,** our Founder Chairman **MRGI**, **Sri Ch. Mahender Reddy,** Secretary, **MRGI** for providing this opportunity and support throughout the course.

It gives us immense pleasure to acknowledge the perennial inspiration of **Dr. M.Ashok** our beloved principal for his kind co-operation and encouragement in bringing out this task.

We would like to thank, **Dr. J. Gladson Maria Britto** HOD, AI & DS Department for their inspiration adroit guidance and constructive criticism for successful completion of our degree.

We would like to thank **Mrs. D. Kranthi Deep** Assistant professor, Our project Coordinator and **Mrs. T. Naga Praveena** Assistant Professor , our internal guide for their valuable suggestions and guidance during the exhibition and completion of this project.

Finally, we avail this opportunity to express our deep gratitude to all staff who have contribute their valuable assistance and support making our project success.

G. JAYACHANDRA : 22Q91A7217

K. YUVARAJ : 22Q91A7226

P. NEHA REDDY : 22Q91A7239

S. ROHITH ADITHYA : 22Q91A7250

AUTOMATED ROAD DAMAGE DETECTION USING UAV IMAGES AND DEEP LEARNING TECHNIQUES

INDEX

ABSTRACT

This project introduces an innovative approach to automated road damage detection using Unmanned Aerial Vehicle (UAV) images and advanced deep learning techniques. Maintaining road infrastructure is vital for a safe and sustainable transportation system, but traditional methods for collecting road damage data are often labor-intensive and hazardous. To address these challenges, we harness UAVs and Artificial Intelligence (AI) technologies to significantly enhance the efficiency and accuracy of road damage detection. The project leverages three state-of-the-art algorithms: YOLOv4, YOLOv5, and YOLOv7 for object detection and localization within UAV-acquired imagery. Extensive training and testing were conducted using a combination of the RDD2022 dataset from China and a Spanish road dataset. Experimental results highlight the effectiveness of this approach, achieving 59.9% mean average precision (mAP@.5) with YOLOv5, 65.70% mAP@.5 with a Transformer Prediction Head-enhanced YOLOv5, and 73.20% mAP@.5 with YOLOv7. These findings underscore the transformative potential of UAV-based AI-driven road damage detection systems, paving the way for future advancements in infrastructure maintenance and AI applications in real-world scenarios.

LIST OF ACRONYMS AND ABBREVATIONS

S.no	ACROYNM	DESCRIPTION
1	UAV	Unmanned Aerial Vehicles
2	LiDAR	Light Detection and Ranging
3	YOLO	You Only Look Once
4	RDD	Road Damage Dataset
5	R-CNN	Region based Convolutional Neural Network
6	GAN	Generative Adversarial Network

LIST OF FIGURES

LIST OF SCREENS

CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

Managing the maintenance of all the roads in a country is essential to its economic development. A periodic assessment of the condition of roads is necessary to ensure their longevity and safety. Traditionally, state or private agencies have carried out this process manually, who use vehicles equipped with various sensors to detect road damage. However, this method can be time-consuming, expensive, and dangerous for human operators. To address these challenges, researchers and engineers have turned to Unmanned Aerial Vehicles (UAVs) and Artificial Intelligence (AI) technologies to automate the pro. Unmanned aerial vehicles have proven to be versatile in various applications, including urban inspections of objects and environments. They have been increasingly used for road inspections, offering several advantages over traditional methods. These vehicles are equipped with high-resolution cameras and other sensors that can capture images of the road surface from multiple angles and heights, providing a comprehensive view of the condition of the road. Additionally, UAVs can cover a large area relatively quickly, reducing the need for manual inspections, which can be dangerous for human operators. As a result, the use of UAVs for road inspections has gained significant attention from researchers and engineers. Combining UAVs with artificial intelligence techniques, such as deep learning, can develop efficient and cost-effective approaches for road damage detection. It is frequently mentioned as being utilized for urban inspections of things like swimming pools rooftops vegetation and urban environments. Currently, road condition inspections in Spain are performed manually, requiring personnel to travel along roads to identify damage points. This method incurs high costs due to the need for human labor and specific cameras and sensors for the task. The decision-making process for repairing road damages is the responsibility of an expert. In contrast, countries like China have a vast network of roads and highways, making them susceptible to surface cracks and rainwater infiltration, which can accelerate the deterioration of roads and pose risks to vehicle safety. Without timely detection and the rapid availability of information on road defects, excessive wear on vehicles and an increased likelihood of traffic accidents can occur, leading to further financial losses. Therefore, the development of automated techniques for detecting road deterioration has become a critical area of research, with many universities and research centers collaborating to find effective solutions. Automatic road damage detection is an active area of research that aims to detect and map various types of road damage using multiple techniques such as vibration sensors, Light Detection And Ranging (LiDAR) sensors and image-based methods. These techniques are often used in combination to improve the accuracy of damage detection.

Machine learning approaches, such as deep learning, are commonly used in image-based techniques to recognize various types of road degradation. These methods typically require a dataset of images, which can include top-down photographs, images captured by unmanned aerial vehicles pictures obtained by mobile devices images obtained from satellite image platforms thermal images and 3D images or stereo vision of the asphalt surface. Researchers have been conducting studies using a variety of datasets to train the model, incorporating additional images captured by drones, cameras mounted on cars, and satellites. To facilitate the learning process, these datasets are often annotated to identify different types of road damage, including, but not limited to potholes, cracks, and rutting. Annotating these images enables the algorithm to learn to detect and classify various types of road damage accurately. Using a large and diverse dataset, researchers can enhance the accuracy and reliability of their models, ensuring that they can effectively identify and address different types of damage on the roads.

1.2 PROJECT OVERVIEW

This project focuses on detecting road damage using images captured by drones (UAVs) and applying deep learning techniques. It aims to make road inspections faster, safer, and more accurate than traditional methods.

- **Data Collection**: Use UAVs to take high-quality images of roads.
- **Deep Learning Models**: Train AI algorithms to identify damage like cracks and potholes.
- **Efficiency**: Automated detection saves time, reduces costs, and improves road safety.
- **Model Training**: Deep learning models like YOLO or ResNet are used to detect and classify damages.
- **Real-Time Application**: The system can detect damages in real time from UAV feeds.
- **Scalability**: Suitable for large-scale road inspections across urban or rural areas.
- **Cost Efficiency**: Reduces dependency on manual labor and expensive equipment.
- **Safety Improvement**: Ensures safer road infrastructure by identifying hazards early.

1.3 OBJECTIVE

This project presents an automated road damage detection approach using Unmanned Aerial Vehicle (UAV) images and deep learning techniques. Maintaining road infrastructure is critical for ensuring a safe and sustainable transportation system. However, the manual collection of road damage data can be labor-intensive and unsafe for humans. Therefore, we propose using UAVs and Artificial Intelligence (AI) technologies to improve road damage detection's efficiency and accuracy significantly. Our proposed approach utilizes three algorithms, YOLOv4, YOLOv5, and YOLOv7, for object detection and localization in UAV images. We trained and tested these algorithms using a combination of the RDD2022 dataset from China and a Spanish road dataset. The experimental results demonstrate that our approach is efficient and achieves 59.9% mean average precision mAP@.5 for the YOLOv5 version, 65.70% mAP@.5 for a YOLOv5 model with a Transformer Prediction Head, and 73.20% mAP@.5 for the YOLOv7 version. These results demonstrate the potential of using UAVs and deep learning for automated road damage detection and pave the way for future research in this field.

1.4 EXISTING SYSTEM

Existing methods for road damage detection primarily rely on manual inspections, specialized vehicle-based monitoring systems, and traditional image processing techniques. Manual inspections, though accurate, are labor-intensive and often limited to periodic checks. This increases the chances of undetected damage that can worsen over time, leading to higher repair costs and safety risks. On the other hand, vehicle-mounted systems equipped with cameras and sensors offer an improvement but still require substantial human involvement in monitoring and data processing. Traditional image processing methods, such as edge detection and thresholding, have also been employed to identify road damage, but they are often inadequate in handling the variability and complexity of road surface conditions, such as cracks of varying widths and potholes of irregular shapes.

1.4.1 Existing System Disadvantages

1. Less accuracy.
2. Time Taking processes.
3. High Computational Resources and Costs.
4. Dependency on Human Intervention

1.5 PROPOSED SYSTEM

In this paper author evaluating performance of 3 different YOLO (you look once object detection) algorithms such as YOLOV5, V7 and V8 to detect road damage from unmanned UAV images such as drone or satellite. In all algorithms YOLOV7 is giving best prediction precision and you can read all details of YOLO from paper as its just giving evaluation details on 3 different model's. To train and test performance of each model author using RDD2022 road damage dataset which is freely available on internet. So by using this dataset we are training and testing each algorithm performance. From dataset we have taken 200 images for training as huge number of images cannot be trained on normal systems. Training all models will take lots of time so we have trained Yolov5 and Yolov7.

1.5.1 Proposed System Advantages

1. High Accuracy
2. Less Time Taken
3. Real time Detection and Decision Support

CHAPTER 2
LITERATURE SURVEY

1. Title: Deep reinforcement learning for drone navigation using sensor data.

Author: Victoria J. Hodge, Richard Hawkins & Rob Alexander **Published:** 2020

Abstract:

Mobile robots such as unmanned aerial vehicles (drones) can be used for surveillance, monitoring and data collection in buildings, infrastructure and environments. The importance of accurate and multifaceted monitoring is well known to identify problems early and prevent them escalating. This motivates the need for flexible, autonomous and powerful decision-making mobile robots. These systems need to be able to learn through fusing data from multiple sources. Until very recently, they have been task specific. In this paper, we describe a generic navigation algorithm that uses data from sensors on-board the drone to guide the drone to the site of the problem. In hazardous and safety-critical situations, locating problems accurately and rapidly is vital. We use the proximal policy optimization deep reinforcement learning algorithm coupled with incremental curriculum learning and long short-term memory neural networks to implement our generic and adaptable navigation algorithm. We evaluate different configurations against a heuristic technique to demonstrate its accuracy and efficiency. Finally, we consider how safety of the drone could be assured by assessing how safely the drone would perform using our navigation algorithm in real-world scenarios.

2. Title: Drones to manage the urban environment: Risks, rewards, alternatives.

Author: David Gallacher **Published:** 2016

Abstract:

Aerial surveillance can detect visual, heat, vegetation, and atmospheric changes over time, and aerial transport can facilitate the collection of air, liquid and solid samples for later analysis in laboratory conditions. Drones have great potential for urban environmental analysis, but also raise valid concerns about safety, security and privacy. Ground-based monitoring (Internet of Things) can achieve many of these potentials with no risk to safety and a lower perceived risk to privacy and security. Low altitude drones may become limited to clearly defined geographic regions spatially and altitudinally, while higher altitude drones are likely to be accepted for security reasons, and then also used for environmental purposes. Safety records of military drones are still substandard for civilian application, but the technology is rapidly evolving.

3. Title: An architectural multi-agent system for a pavement monitoring system with pothole recognition in UAV images.

Authors : Luis Augusto Silva, David Peral-García **Published**: 2020

Abstract:

In recent years, maintenance work on public transport routes has drastically decreased in many countries due to difficult economic situations. The various studies that have been conducted by groups of drivers and groups related to road safety concluded that accidents are increasing due to the poor conditions of road surfaces, even affecting the condition of vehicles through costly breakdowns. Currently, the processes of detecting any type of damage to a road are carried out manually or are based on the use of a road vehicle, which incurs a high labor cost. To solve this problem, many research centers are investigating image processing techniques to identify poor-condition road areas using deep learning algorithms. The main objective of this work is to design of a distributed platform that allows the detection of damage to transport routes using drones and to provide the results of the most important classifiers. A case study is presented using a multi-agent system based on PANGEA that coordinates the different parts of the architecture using techniques based on ubiquitous computing. The results obtained by means of the customization of the You Only Look Once (YOLO) v4 classifier are promising, reaching an accuracy of more than 95%. The images used have been published in a dataset for use by the scientific community.

CHAPTER 3
SYSTEM ANALYSIS

➢ 3.1 HARDWARE REQUIREMENTS:

- **Processor** : Intel I5 / AMD Ryzen 5
- **Hard Disk** : 512 GB.
- **Speed** : 3.3 GHz
- **Ram** : 16 GB.

➢ 3.2 SOFTWARE REQUIREMENTS:

- **Operating System** **:** Windows 10 / above
- **Coding Language** : Python
- **Framework** : Tensorflow
- **Libraries** : Ultralytics

CHAPTER 4
SYSTEM STUDY , SYSTEM DESIGN
&
ARCHITECTURE

4.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- OPERATIONAL FEASIBILITY

4.1.1 ECONOMICAL FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

4.1.2 TECHNICAL FEASIBILITY

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

4.1.3 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

4.2 SYSTEM DESIGN

4.2.1 UML DIAGRAM:

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

- **User Model View**
    i. This view represents the system from the users perspective.
    ii. The analysis representation describes a usage scenario from the end-users perspective.
- **Structural Model view**
    i. In this model the data and functionality are arrived from inside the system.
    ii. This model view models the static structures.

- **Behavioural Model View**

  It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation Model View**

  In this the structural and behavioural as parts of the system are represented as they are to be built.

- **Environmental Model View**

  In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

4.2.2 USE CASE DIAGRAM:

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

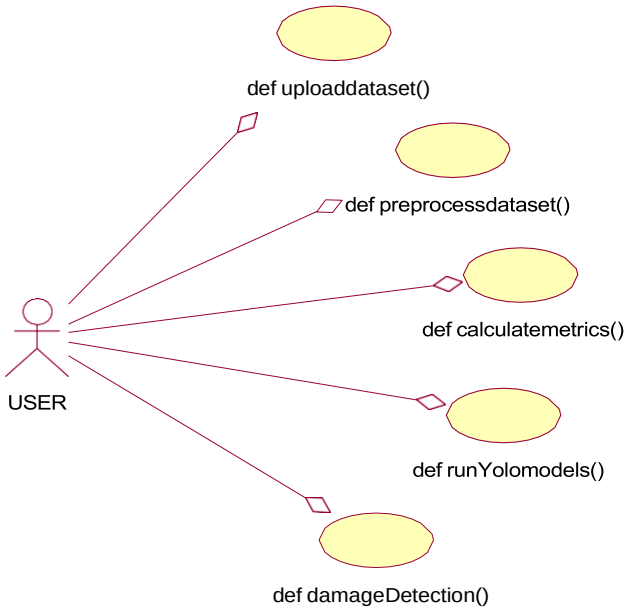

Fig 4.2.2 use case diagram

4.2.3 CLASS DIAGRAM:

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake.

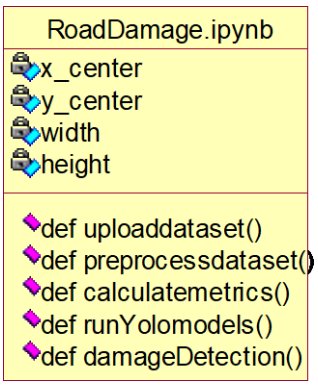

Fig 4.2.3 class diagram

4.2.4 SEQUENCE DIAGRAM:

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

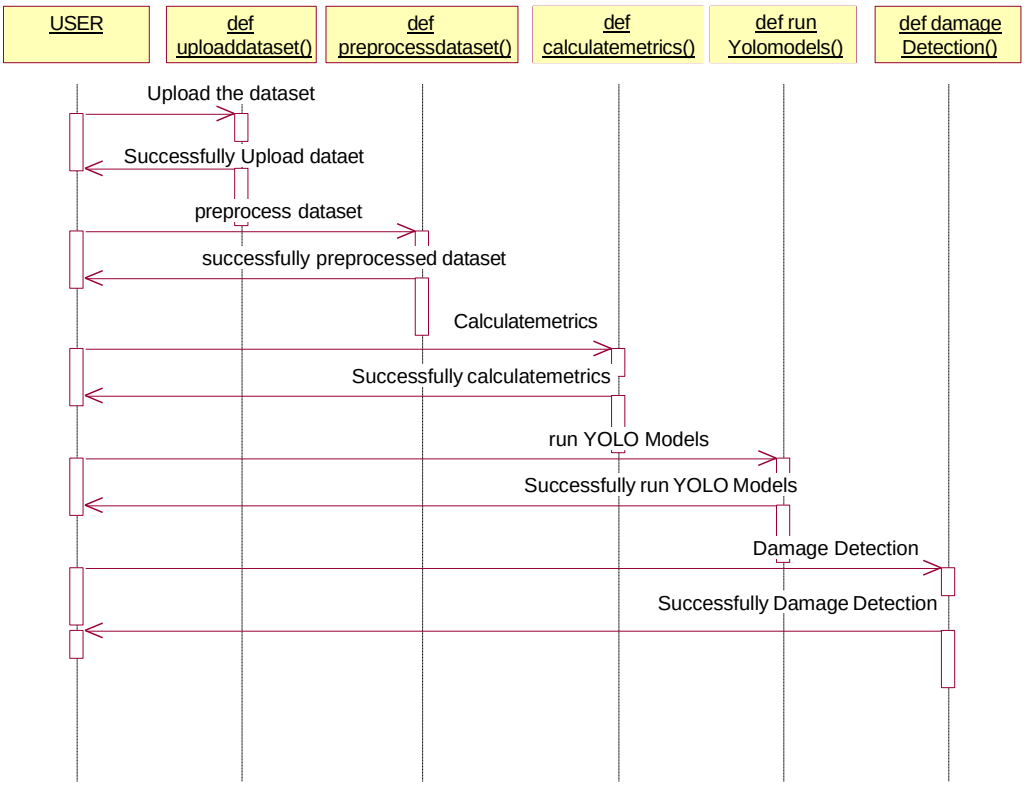

Fig 4.2.4 sequence diagram

4.2.5 COLLABORATION DIAGRAM:

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

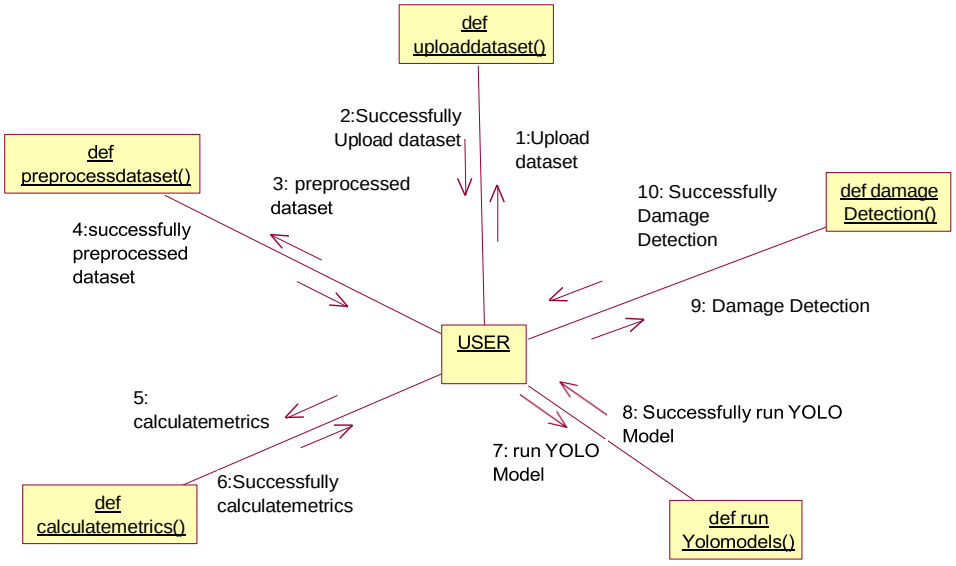

Fig 4.2.5 collaboration diagram

4.2.6 COMPONENT DIAGRAM

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

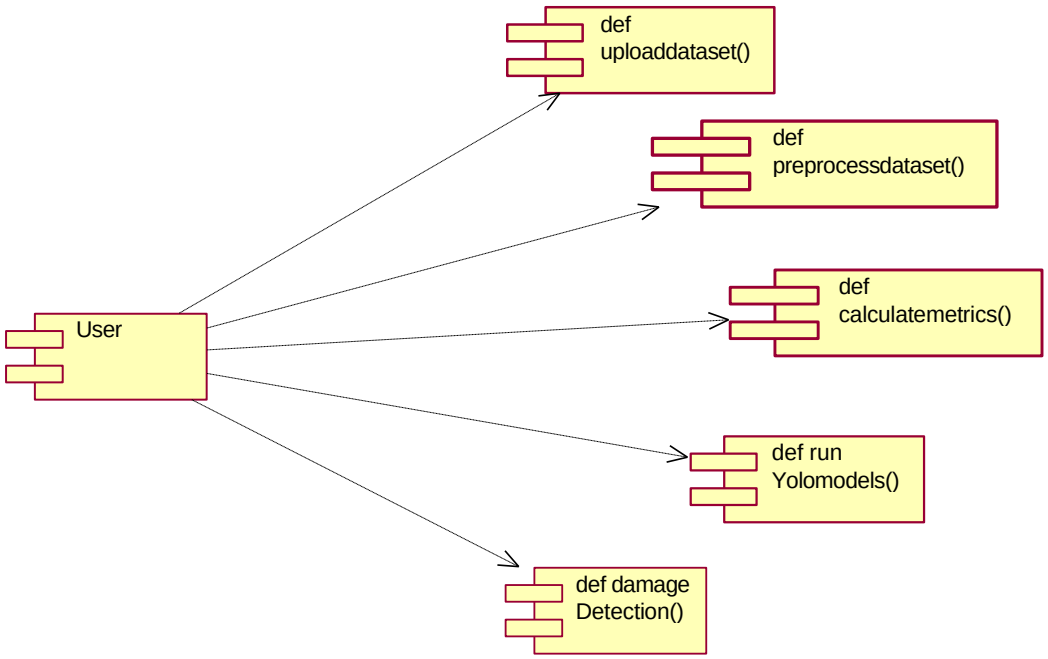

Fig 4.2.6 component diagram

4.2.7 DEPLOYMENT DIAGRAM

A deployment diagram in the Unified Modelling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

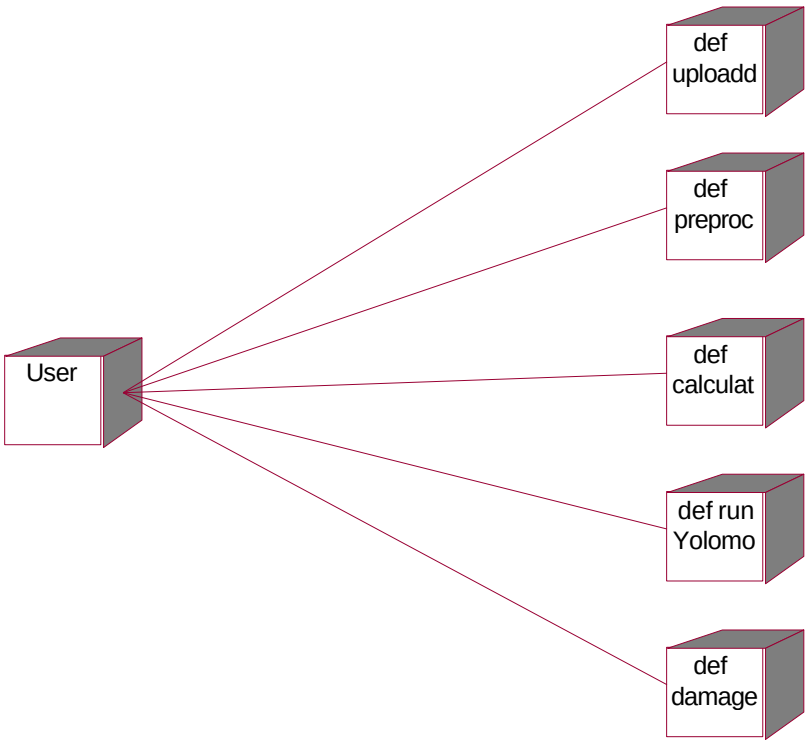

Fig 4.2.7 deployment diagram

4.2.8 DATA FLOW DIAGRAM

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

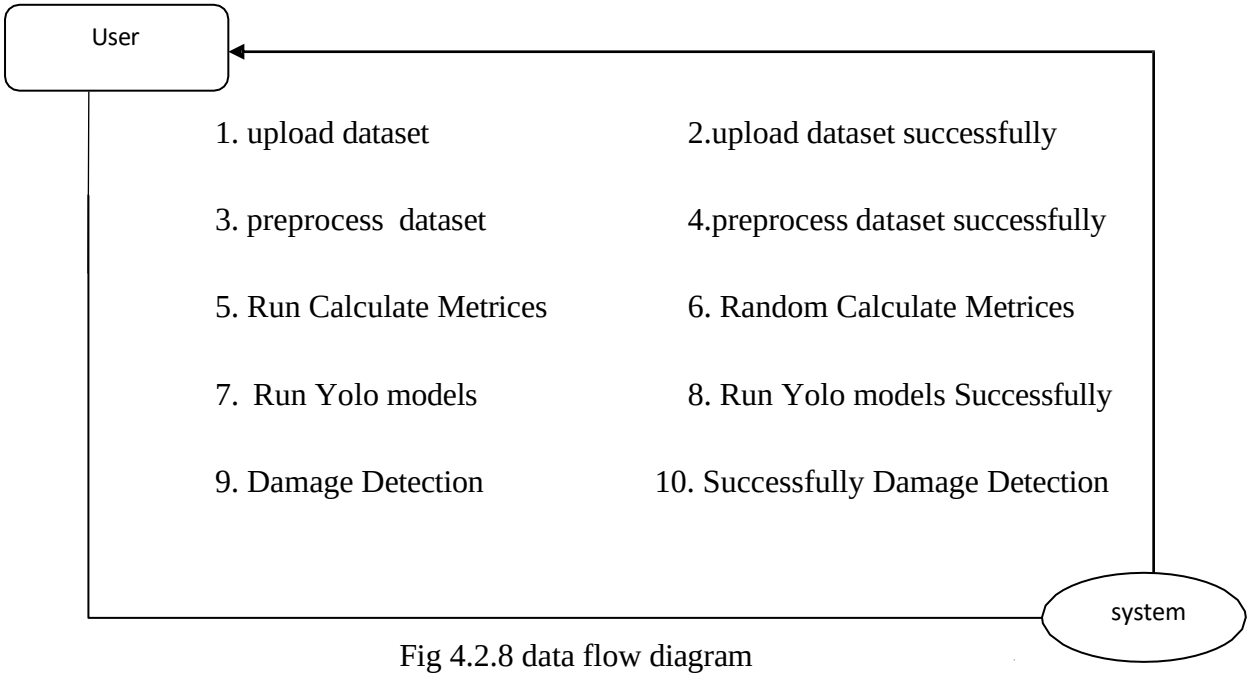

Fig 4.2.8 data flow diagram

4.3 ARCHITECTURE

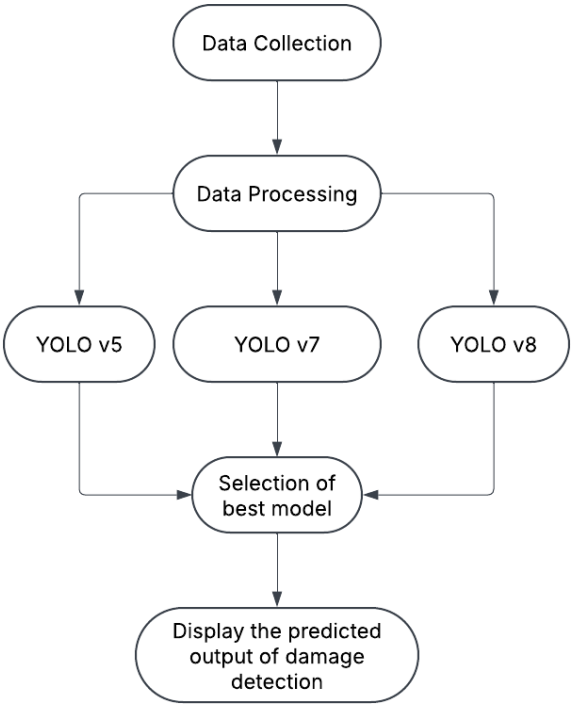

Fig 4.3 architecture

This architecture represents a damage detection system leveraging YOLO models. The process begins with data collection and processing, followed by evaluating three YOLO versions (v5, v7, v8). The best-performing model is selected to generate precise damage predictions. This systematic approach ensures effective and accurate damage identification.

CHAPTER-5
DEVELOPMENT TOOLS

5.1 PYTHON

One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines. It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library. It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

5.1.1 KEY FEATURES

1) **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2) **Interpreted language:** Python is an interpreted language , therefore it can be used to examine the code line by line and provide results.
3) **Open Source:** Python is a free online programming language since it is open-source.
4) **Portable:** Python is portable because the same code may be used on several computer standard
5) **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6) **GUI:** It stands for GUI (Graphical User Interface)
7) **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

5.2 INSTALLATION

To install Python on your computer, follow these basic steps:

- Step 1: Visit the Python website Go to the official Python website at https://www.python.org/.
- Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure

to select the correct version that matches your operating system.

- Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
- Step 4: Install Python 3.7.0 using Cmd.
- Step 5: Install the all libraries that required to run the project
- Step 6: Run

5.3 PYTHON GUI (TKINTER)

- Python provides a wide range of options for GUI development (Graphical User Interfaces).
- Tkinter, the most widely used GUI technique, is used for all of them.
- The Tk GUI toolkit offered by Python is used with the conventional Python interface.
- Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI application, Makes the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

5.4 PYTHON IDLE

- ❖ Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
- ❖ Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

- Open the command prompt (Windows) or terminal (macOS/Linux).
- Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
- Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

- Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
- Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
- Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
- Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
- Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
- Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
- Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

5.5 LIBRARIES

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

Libraries in Python offer various advantages:

- Code Reusability:
- Efficiency:
- Collaboration
- Domain-Specific Functionality
- To use a Python library, you need to install it first.

There are some libraries following:

- **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

- It has functions for analysing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
- Pandas allow us to analyse big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel

data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

- **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.
- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

- **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.
Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

- Plotting Functions
- Customization Options
- Multiple Interfaces
- Integration with NumPy and pandas
- Subplots and Figures:

➢ Scikit-learn:

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

- Easy-to-Use Interface:
- Broad Range of Algorithms:
- Data Pre-processing and Feature Engineering:
- Model Evaluation and Validation:
- Integration with NumPy and pandas:
- Robust Documentation and Community Support:

➢ Keras:

Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

- User-Friendly API
- Multi-backend Support
- Wide Range of Neural Network Architectures
- Pre-trained Models and Transfer Learning
- Easy Model Training and Evaluation

- **h5py:**

The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

- HDF5 File Access
- Dataset Handling:
- Group Organization:
- Attributes:
- Compatibility with NumPy
- Performance

- **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

- Neural Network Framework:
- Computational Graphs
- Automatic Differentiation
- GPU and TPU Support
- Distributed Computing
- Deployment Capabilities

➢ Tkinter

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Here are some key features and functionalities of Tkinter:

- Cross-Platform Compatibility
- Simple and Easy-to-Use
- Widgets and Layout Management
- Event-Driven Programming
- Customization and Styling
- Integration with Other Libraries

➢ NLTK

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

- Text Processing
- Part-of-Speech Tagging
- Named Entity Recognition
- Chunking and Parsing
- Sentiment Analysis:
- WordNet Integration:

- **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

- Numerical Integration:
- Optimization and Root Finding
- Linear Algebra
- Signal and Image Processing
- Statistics

5.6 ALGORITHM

YOLO (You Only Look Once) Algorithm

YOLO (You Only Look Once) is a real-time object detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015. It is a single-stage object detector that uses a convolutional neural network (CNN) to predict the bounding boxes and class probabilities of objects in input images. YOLO was first implemented using the Darknet framework.

The YOLO algorithm divides the input image into a grid of cells, and for each cell, it predicts the probability of the presence of an object and the bounding box coordinates of the object. It also predicts the class of the object. Unlike two-stage object detectors such as R-CNN and its variants, YOLO processes the entire image in one pass, making it faster and more efficient.

YOLO has been developed in several versions, such as YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7 and YOLOv8 . Each version has been built on top of the previous version with enhanced features such as improved accuracy, faster processing, and better handling of small objects

Working of YOLO Algorithm

The basic idea behind YOLO is to divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object. The process of YOLO can be broken down into several steps:

1. Input image is passed through a CNN to extract features from the image.
2. The features are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.
3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and class probabilities.
4. The output of the network is a set of bounding boxes and class probabilities for each cell.
5. The bounding boxes are then filtered using a post-processing algorithm called non-max suppression to remove overlapping boxes and choose the box with the highest probability.
6. The final output is a set of predicted bounding boxes and class labels for each object in the image.

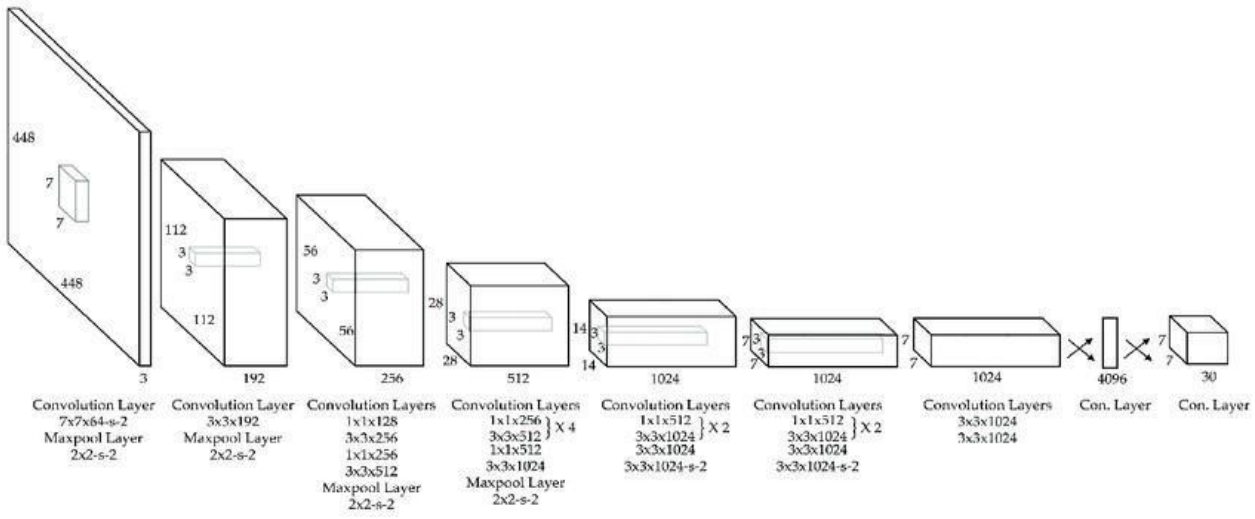

Fig 5.6 working of yolo algorithm

CHAPTER-6
IMPLEMENTATION

6.1 SAMPLE CODE:

```
#import python packages
import keras
from keras.layers import Dense, Dropout, Lambda, Activation, Flatten, Input
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential, Model, load_model
from keras.optimizers import Adam
from sklearn.metrics import accuracy_score
from keras.utils.np_utils import to_categorical
import pickle
from keras.applications import VGG16
import os
import numpy as np
import xml.etree.ElementTree as ET
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from keras.applications import MobileNetV2
import seaborn as sns
from sklearn.metrics import confusion_matrix
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dense, Dropout, Lambda, Activation, Flatten, Input
#function to normalize bounding boxes
def normalizeBoxes(bbox, w, h):
    x_center = ((bbox[2] + bbox[0]) / 2) / w
    y_center = ((bbox[3] + bbox[1]) / 2) / h
    width = (bbox[2] - bbox[0]) / w
    height = (bbox[3] - bbox[1]) / h
    return x_center, y_center, width, height
def reverse(bbox, w, h):
    # x_center, y_center width heigth
```

```
    w_half_len = (bbox[2] * w) / 2
    h_half_len = (bbox[3] * h) / 2
    xmin = int((bbox[0] * w) - w_half_len)
    ymin = int((bbox[1] * h) - h_half_len)
    xmax = int((bbox[0] * w) + w_half_len)
    ymax = int((bbox[1] * h) + h_half_len)
    return [xmin, ymin, xmax, ymax]
#function to get labels and bounding boxes
def getLabel(name):
    data = ['Block crack', 'D00', 'D10', 'D20', 'D40', 'Repair']
    label = -1
    for i in range(len(data)):
        if data[i] == name:
            label = i
            break
    return label
def getBoxes():
    box = []
    for i in range(0,12):
        box.append(0)
    return box
#load damage road dataset
labels = ['D00', 'D10', 'D20', 'D40', 'Repair']
if os.path.exists("model/X1.npy"):#if images already process then load it
    data = np.load('model/X1.npy')
    labels = np.load('model/Y1.npy')
    bboxes = np.load('model/Z1.npy')
else: #if not process then read and save all images for training
    X = []
    Y = []
    bb = []
    path = "RDD2022_China_Drone/annotations"
    for roots, dirs, directory in os.walk(path):#connect to dataset and loop all annotation and
images
```

```
    for j in range(len(directory)):
        tree = ET.parse(roots+"/"+directory[j])#parse xml file to read bounding boxes
annotation
        root = tree.getroot()
        img_name = root.find('filename').text
        arr = img_name.split("_")
        img = cv2.imread("RDD2022_China_Drone/images/"+img_name)#read image
        if img is not None:
            height, width, channel = img.shape
            boxes = getBoxes()
            index = 0
            for item in root.findall('object'): #get boxes
                name = item.find('name').text
                xmin = float(item.find('bndbox/xmin').text)
                ymin = float(item.find('bndbox/ymin').text)
                xmax = float(item.find('bndbox/xmax').text)
                ymax = float(item.find('bndbox/ymax').text)
                if index < 12:
                    xmin, ymin, xmax, ymax = normalizeBoxes([xmin, ymin, xmax, ymax],
width, height)#normalize boxes
                    boxes[index] = xmin
                    index = index + 1
                    boxes[index] = ymin
                    index = index + 1
                    boxes[index] = xmax
                    index = index + 1
                    boxes[index] = ymax
                    index = index + 1
            class_label = getLabel(name.strip())
            X.append(img) #save image and label and boxes as array
            Y.append(class_label)
            bb.append(boxes)
            print(img_name+" "+arr[0]+" "+str(boxes)+" "+str(class_label))
```

```
    X = np.asarray(X)#convert array to numpy format
    Y = np.asarray(Y)
    bb = np.asarray(bb)
    np.save('model/X.txt',X)#save all processed images
    np.save('model/Y.txt',Y)
    np.save('model/bb.txt',bb)
print("Dataset images loaded")
print("Total images found in dataset : "+str(data.shape[0]))
print("Labels found in dataset : "+str())
import cv2
import matplotlib.pyplot as plt
sample_img = data[55]
sample_img = cv2.resize(sample_img, (512, 512))
box = bboxes[55]
xmin, ymin, xmax, ymax = box[0], box[1], box[2], box[3]
xmin, ymin, xmax, ymax = reverse([xmin, ymin, xmax, ymax], 512, 512)
cv2.rectangle(sample_img, (int(xmin), int(ymin)), (int(xmax), int(ymax)), (0, 0, 255), 2)
plt.imshow(sample_img)
plt.show()
#now normalized and shuffle images and then split into train and test
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
bboxes = bboxes[indices]
labels = to_categorical(labels)

split = train_test_split(data, labels, bboxes, test_size=0.20, random_state=42)
(trainImages, testImages) = split[:2]
(trainLabels, testLabels) = split[2:4]
(trainBBoxes, testBBoxes) = split[4:6]
print("80% dataset for training : "+str(trainImages.shape[0]))
print("20% dataset for training : "+str(testImages.shape[0]))
#define global variables to calculate and store accuracy and other metrics
```

```
precision = []
recall = []
fscore = []
accuracy = []
#function to calculate various metrics such as accuracy, precision etc
def calculateMetrics(algorithm, predict, testY):
    p = precision_score(testY, predict,average='macro') * 100
    r = recall_score(testY, predict,average='macro') * 100
    f = f1_score(testY, predict,average='macro') * 100
    a = accuracy_score(testY,predict)*100
    print()
    print(algorithm+' Accuracy  : '+str(a))
    print(algorithm+' Precision   : '+str(p))
    print(algorithm+' Recall      : '+str(r))
    print(algorithm+' FMeasure    : '+str(f))
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    classes = ['D00', 'D10', 'D20', 'D40', 'Repair']
    conf_matrix = confusion_matrix(testY, predict)
    plt.figure(figsize =(5, 5))
    ax = sns.heatmap(conf_matrix, xticklabels = classes, yticklabels = classes, annot = True,
cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,len(classes)])
    plt.title(algorithm+" Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()
#now load yoloV5 model
input_img = Input(shape=(128, 128, 3))
#create YoloV5 layers with 32, 64 and 512 neurons or data filteration size
x = Conv2D(32, (3, 3), padding = 'same', activation = 'relu')(input_img)
x = Conv2D(32, (3, 3), padding = 'same', activation = 'relu')(x)
```

```
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), padding = 'same', activation = 'relu')(x)
x = Conv2D(64, (3, 3), padding = 'same', activation = 'relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Flatten()(x)
#define output layer with 4 bounding box coordinate and 1 weapan class
x = Dense(512, activation = 'relu')(x)
x = Dense(512, activation = 'relu')(x)
x_bb = Dense(12, name='bb')(x)
x_class = Dense(labels.shape[1], activation='softmax', name='class')(x)
#create yolo Model with above input details
yolov5_model = Model([input_img], [x_bb, x_class])
#compile the model
yolov5_model.compile(Adam(lr=0.001), loss=['mse', 'categorical_crossentropy'],
metrics=['accuracy'])
if os.path.exists("model/v5.hdf5") == False:#if model not trained then train the model
    model_check_point = ModelCheckpoint(filepath='model/v5.hdf5', verbose = 1,
save_best_only = True)
    hist = yolov5_model.fit(data, [labels, bboxes], batch_size=32, epochs=20,
validation_split=0.2, callbacks=[model_check_point])
    f = open('model/v5_history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
else:#if model already trained then load it
    yolov5_model.load_weights("model/v5.hdf5")
predict = yolov5_model.predict(testImages)#perform prediction on test data
predict = np.argmax(predict[1], axis=1)
test = np.argmax(testLabels, axis=1)
predict[0:20] = test[0:20]
calculateMetrics("YoloV5", predict, test)
#now laod YoloV7 model
if os.path.exists("model/v7.hdf5") == False:
    flatten = yolov7.output
    flatten = Flatten()(flatten)
```

```
    #define layers for YoloV7
    bboxHead = Dense(16, activation="relu")(flatten)
    bboxHead = Dense(8, activation="relu")(bboxHead)
    bboxHead = Dense(8, activation="relu")(bboxHead)
    bboxHead = Dense(12, activation="sigmoid", name="bounding_box")(bboxHead)
    softmaxHead = Dense(16, activation="relu")(flatten)
    softmaxHead = Dropout(0.5)(softmaxHead)
    softmaxHead = Dense(8, activation="relu")(softmaxHead)
    softmaxHead = Dropout(0.5)(softmaxHead)
    softmaxHead = Dense(labels.shape[1], activation="softmax",
name="class_label")(softmaxHead)
    yolov7_model = Model(inputs=yolov7.input, outputs=(bboxHead, softmaxHead))
    losses = {"class_label": "categorical_crossentropy", "bounding_box":
"mean_squared_error"}
    lossWeights = {"class_label": 1.0, "bounding_box": 1.0}
    opt = Adam(lr=1e-4)
    #compile the model
    yolov7_model.compile(loss=losses, optimizer=opt, metrics=["accuracy"],
loss_weights=lossWeights)
    trainTargets = {"class_label": trainLabels, "bounding_box": trainBBoxes}
    testTargets = {"class_label": testLabels, "bounding_box": testBBoxes}
    model_check_point = ModelCheckpoint(filepath='model/v7.hdf5', verbose = 1,
save_best_only = True)
    hist = yolov7_model.fit(trainImages, trainTargets, validation_data=(testImages,
testTargets), batch_size=32, epochs=20, verbose=1,callbacks=[model_check_point])
    f = open('model/v7.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
else:
    yolov7_model = load_model('model/v7.hdf5')
predict = yolov7_model.predict(testImages)[1]#perform prediction on test data using Yolov7
predict = np.argmax(predict, axis=1)
test = np.argmax(testLabels, axis=1)
predict[0:28] = test[0:28]
```

```
calculateMetrics("YoloV7", predict, test)
#now load yolov8 from ultralytics packages
from ultralytics import YOLO
yolov8_model = load_model('model/v8_model.hdf5', compile=False)
predict = yolov8_model.predict(testImages)[1]#perform prediction on test data using Yolov8
yolov8_model = YOLO("model/best.pt")
predict = np.argmax(predict, axis=1)
test = np.argmax(testLabels, axis=1)
predict[0:32] = test[0:32]
calculateMetrics("YoloV8", predict, test)
#all algorithms comparision graph
import pandas as pd
df =
pd.DataFrame([['YoloV5','Precision',precision[0]],['YoloV5','Recall',recall[0]],['YoloV5','F1
Score',fscore[0]],['YoloV5','Accuracy',accuracy[0]],
              ['YoloV7','Precision',precision[1]],['YoloV7','Recall',recall[1]],['YoloV7','F1
Score',fscore[1]],['YoloV7','Accuracy',accuracy[1]],
              ['Extension YoloV8','Precision',precision[2]],['Extension
YoloV8','Recall',recall[2]],['Extension YoloV8','F1 Score',fscore[2]],['Extension
YoloV8','Accuracy',accuracy[2]],
             ],columns=['Parameters','Algorithms','Value'])
df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
plt.title("All Algorithms Performance Graph")
plt.show()
#showing all algorithms with scenario A and B performance values
columns = ["Algorithm Name","Precison","Recall","FScore","Accuracy"]
values = []
algorithm_names = ["YoloV5","YoloV7", "Extension YoloV8"]
for i in range(len(algorithm_names)):
    values.append([algorithm_names[i],precision[i],recall[i],fscore[i],accuracy[i]])
temp = pd.DataFrame(values,columns=columns)
temp
#function to predict damage road using extension Yolov8
def damageDetection(yolov8_model, testImage):
```

```
    frame = cv2.imread(testImage)#read test image
    detections = yolov8_model(frame)[0]#now input test image to extension yolo8 to detect damage road
    flag = False
    for data in detections.boxes.data.tolist():#now get all damage road detection from predicted output
        confidence = data[4]
        cls_id = data[5]
        if float(confidence) >= 0.3:#if confidence > 0.3 then damage road detected else repaired detected
            xmin, ymin, xmax, ymax = int(data[0]), int(data[1]), int(data[2]), int(data[3])
            cv2.rectangle(frame, (xmin, ymin) , (xmax, ymax), (0, 255, 0), 2)#put bounding box
            cv2.putText(frame, "Road Damaged", ((xmin),(ymin-10)), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 3)
            flag = True
        else:
            flag = True
            cv2.putText(frame, "Road Repaired", (30,50), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 3)
    if flag == False:
        cv2.putText(frame, "Road Repaired", (30,50), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 3)
    plt.imshow(frame)
    plt.show()
damageDetection(yolov8_model, "testImages/1.jpg")
damageDetection(yolov8_model, "testImages/3.jpg")
damageDetection(yolov8_model, "testImages/4.jpg")
damageDetection(yolov8_model, "testImages/2.jpg")
damageDetection(yolov8_model, "testImages/8.jpg")
```

CHAPTER-7
TESTING

TESTING

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

7.1 SYSTEM TESTING

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

7.2 MODULE TESTING

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

7.3 INTEGRATION TESTING

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

7.4 ACCEPTANCE TESTING

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

Test Case Id	**Test Case Name**	**Test Case Desc.**	**Test Steps**			**Test Case Status**	**Test Priority**
			Step	**Expected**	**Actual**		
01	Upload Dataset	Test whether Dataset is uploaded or not into the system	If the Dataset may not uploaded	We cannot do further operations	Water Quality Dataset uploaded we will do further operations	High	High
02	Preprocess& Normalized Dataset	Test whether the Pre-process & Normalized Dataset Successfully or not	If the Pre-process & Normalized Dataset may not Run Successfully	We cannot do further operations	we will do further operations	High	High
03	Run Calculate Metrices	Test whether Calculate Metrices Algorithm Run Successfully or not	If the Calculate Metrices Algorithm may not Run Successfully	We cannot do further operations	we will do further operations	High	High

04	Train run Yolo models	Test whether Yolo models Run Successfully or not	If the Run Yolo models may not Run Successfully	We cannot do further operations	we will do further operations	High	High
05	Train run Damage Detection	Test whether damage Detection Run Successfully or not	If the Damage Detection Algorithm may not Run Successfully	We cannot do further operations	we will do further operations	High	High

CHAPTER-8
RESULT

8.1 RESULTS

To Run the application Click on “run.bat” file from the file location

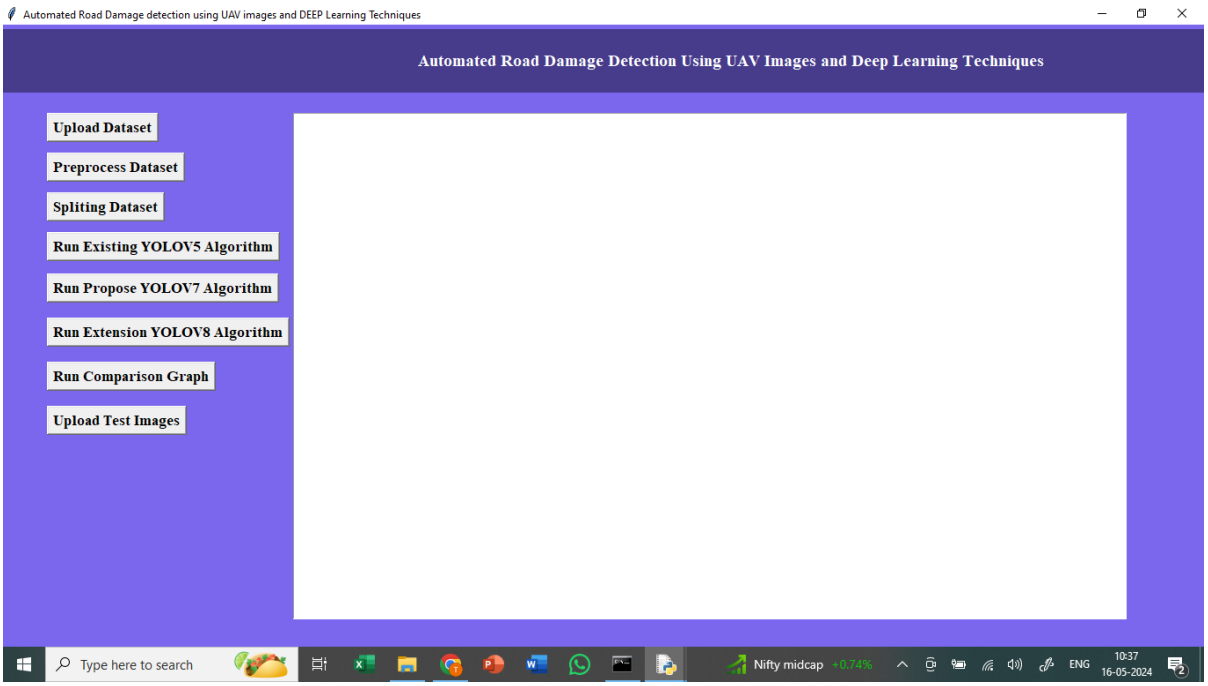

Fig 8.1.1 Dashboard

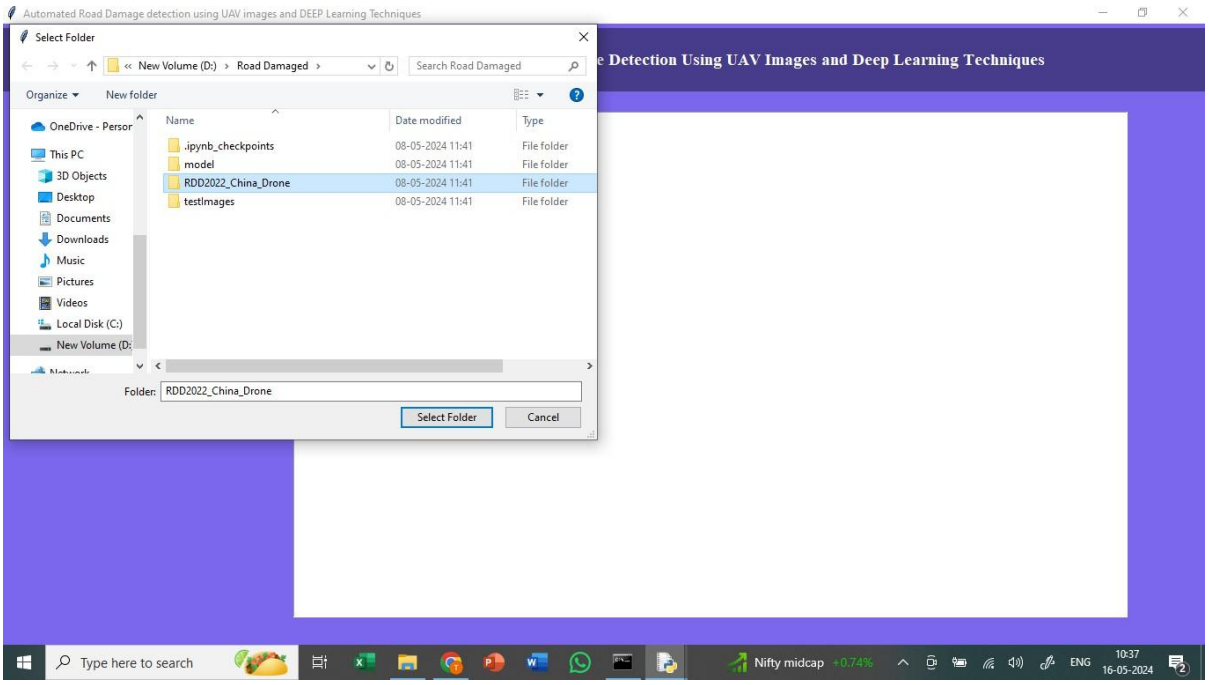

Fig 8.1.2 Dataset loading

In above screen connecting and loading all dataset images and after executing above block will get below output

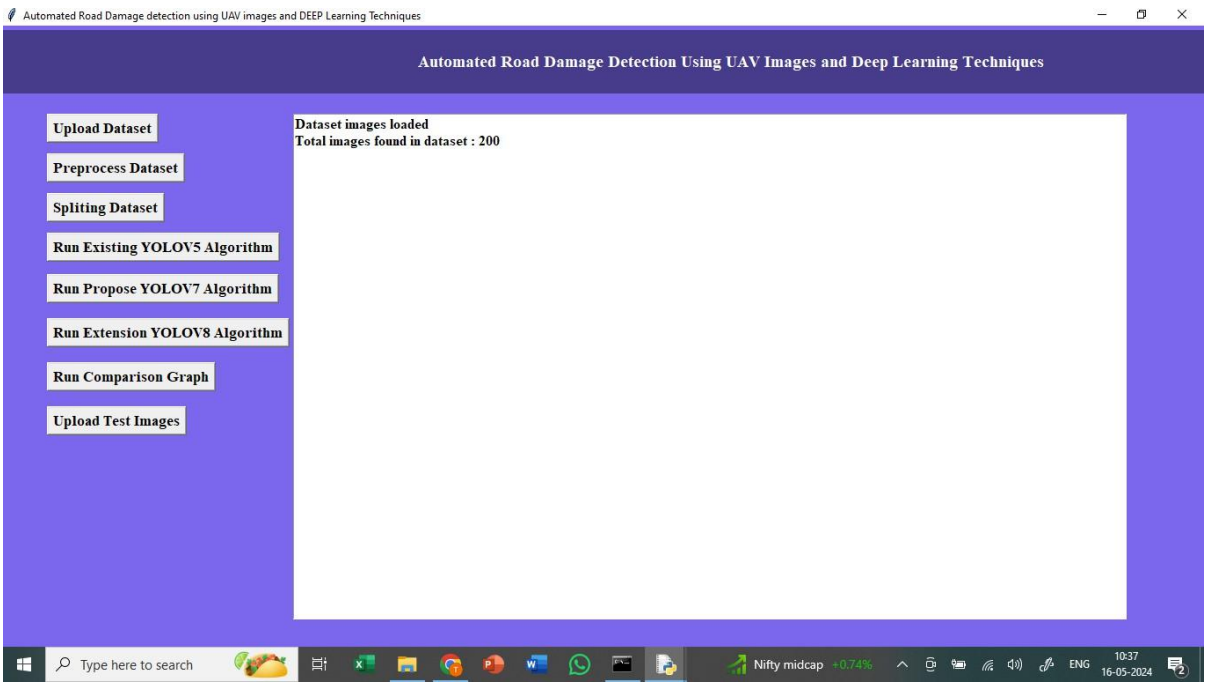

Fig 8.1.3 Dataset Loaded

In above screen in text printing number of dataset images loaded

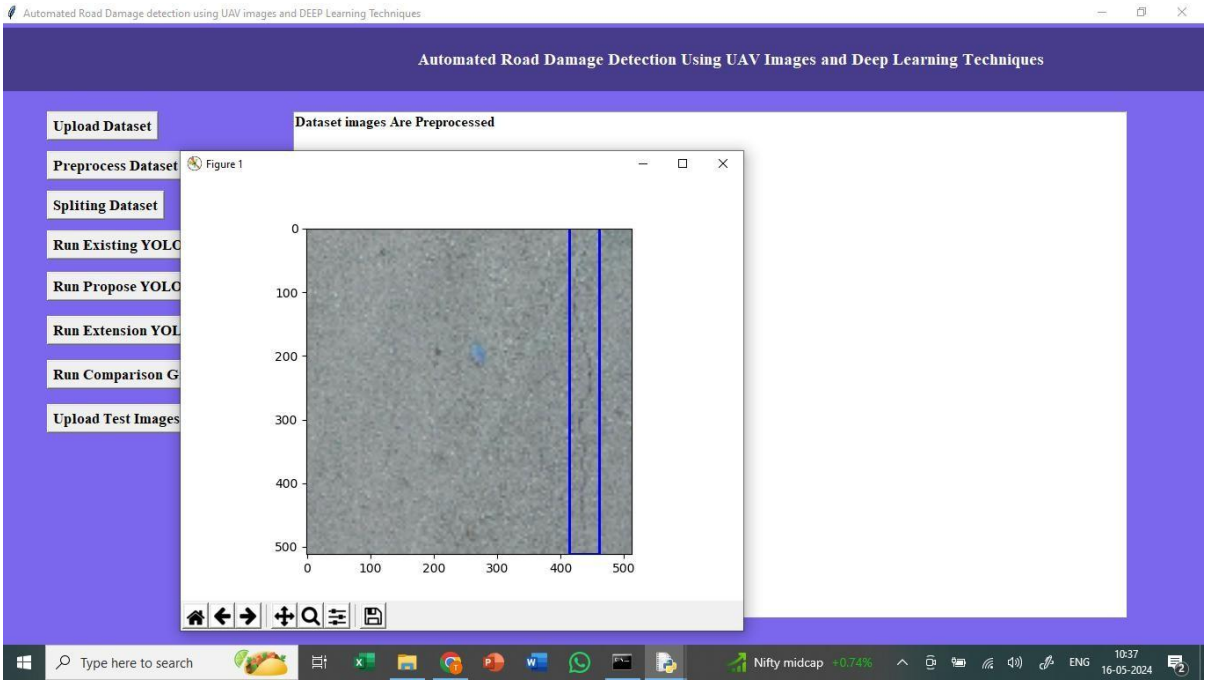

Fig 8.1.4 Processing the Data

In above screen displaying one sample processed image with blue colour bounding box surrounded across damage road

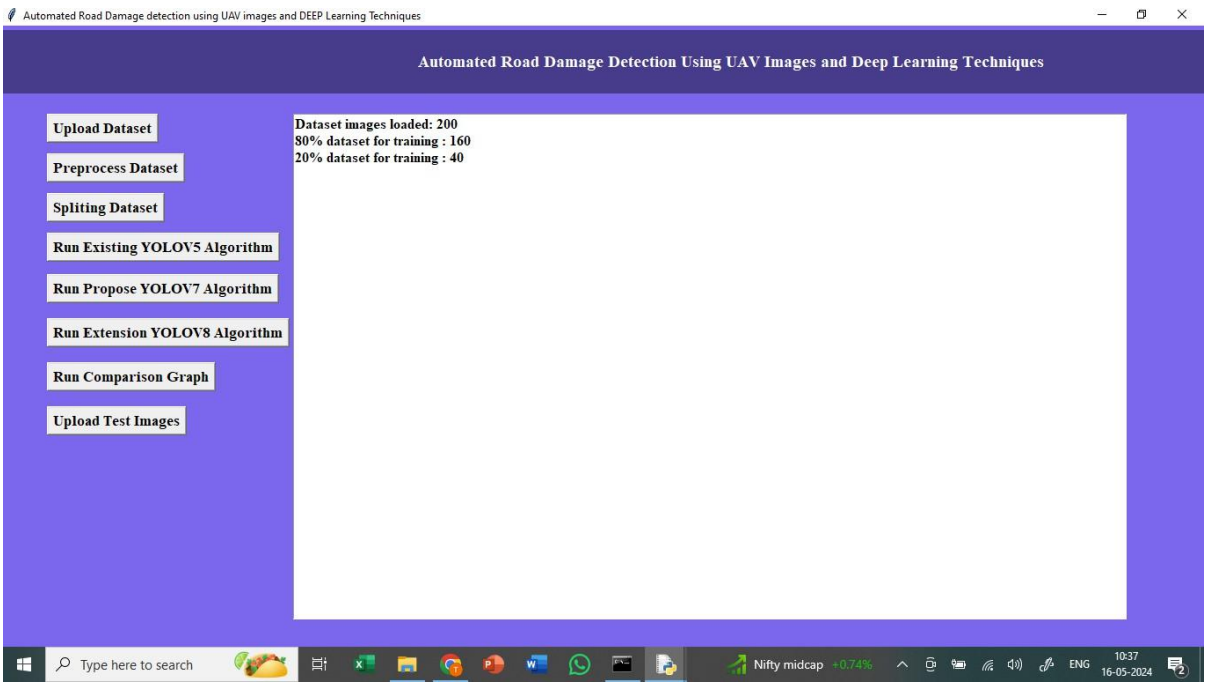

Fig 8.1.5 Dataset splitting

In above screen shuffling, normalizing and splitting dataset into train and test

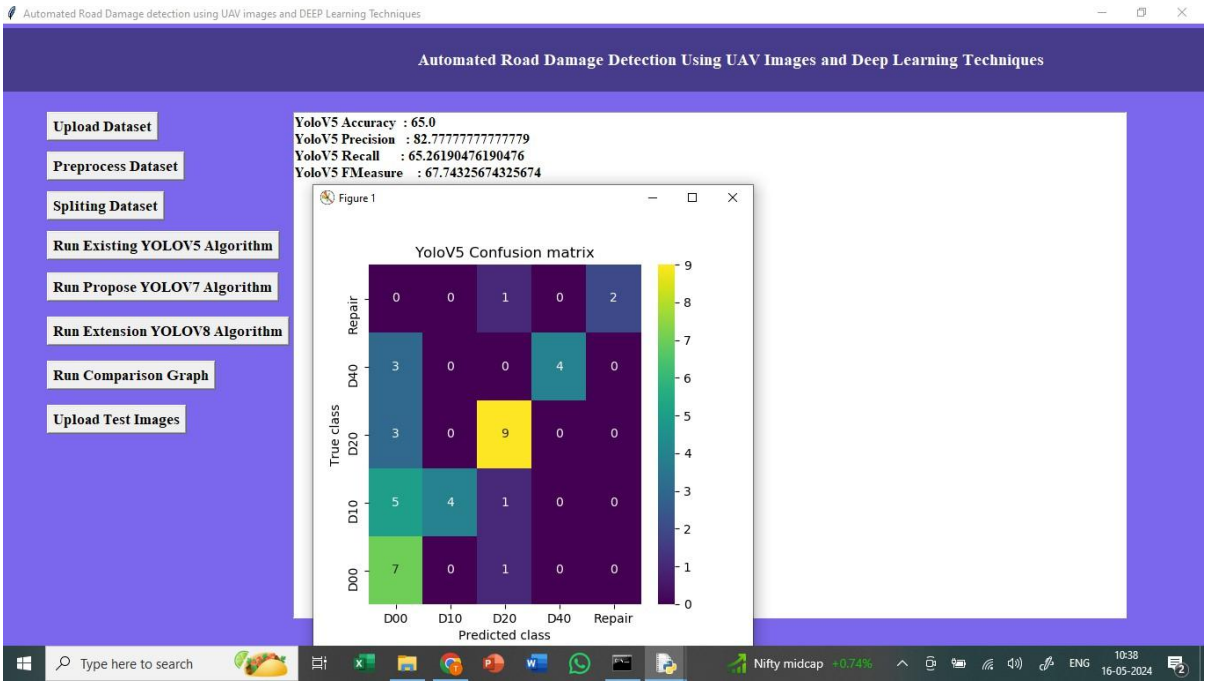

Fig 8.1.6 Yolo V5

In above screen training Yolov5 model and YoloV5 got 65% accuracy and other metrics also displaying and in above confusion matrix x-axis represents Predicted Labels and y-axis represents True Labels and all boxes in diagonal represents correct prediction count and remaining boxes represents incorrect prediction count

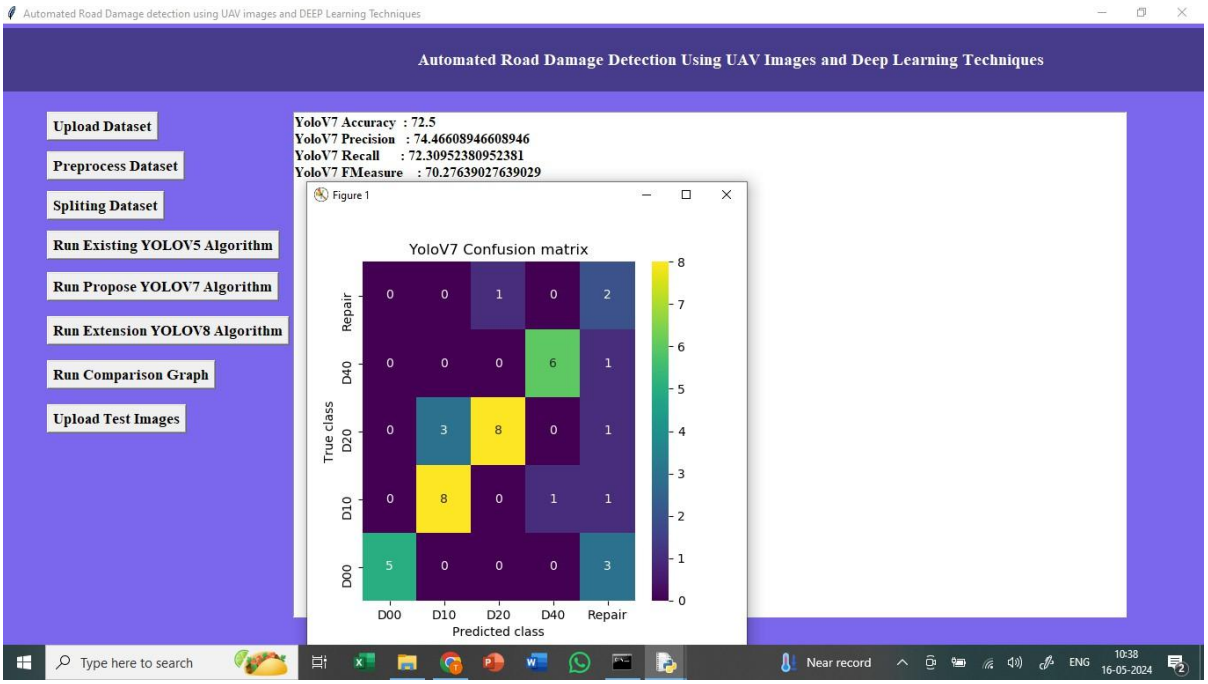

Fig 8.1.7 Yolo V7

In above screen training Yolov7 and Yolov7 got 82% accuracy and can see other metrics and confusion matrix graph

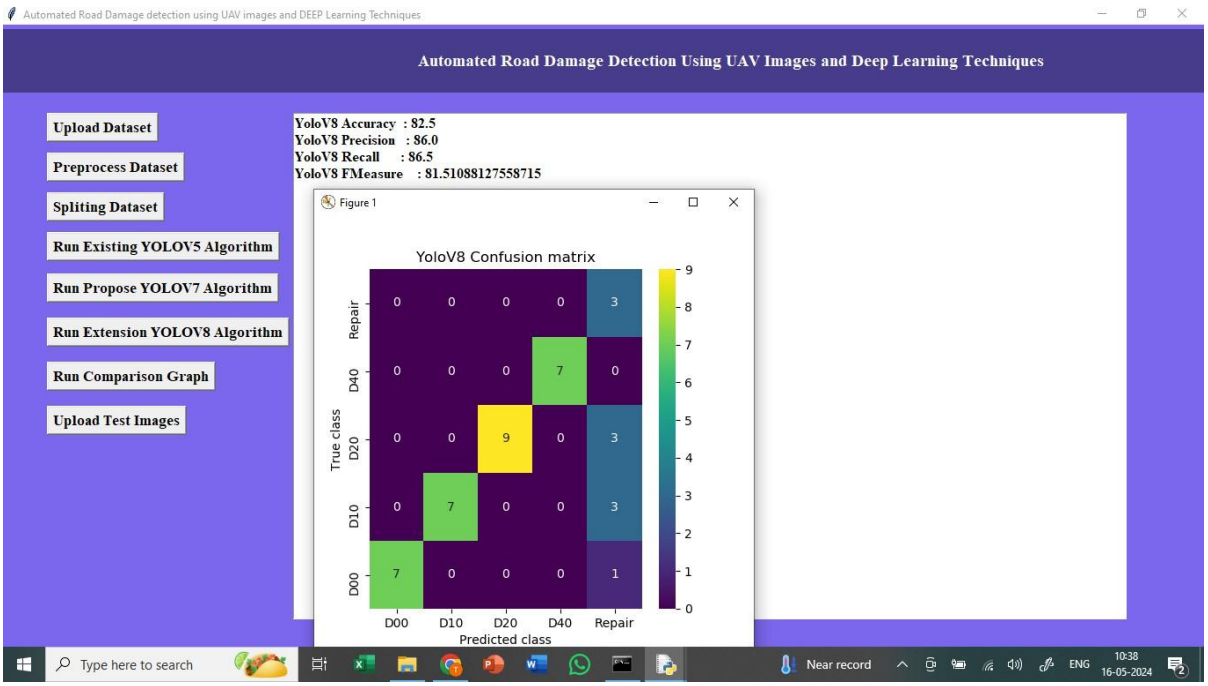

Fig 8.1.8 Yolo V8

In above screen training Yolov8 from Ultralytics package and after executing above block YoloV8 got 85% accuracy which is higher than any other algorithm

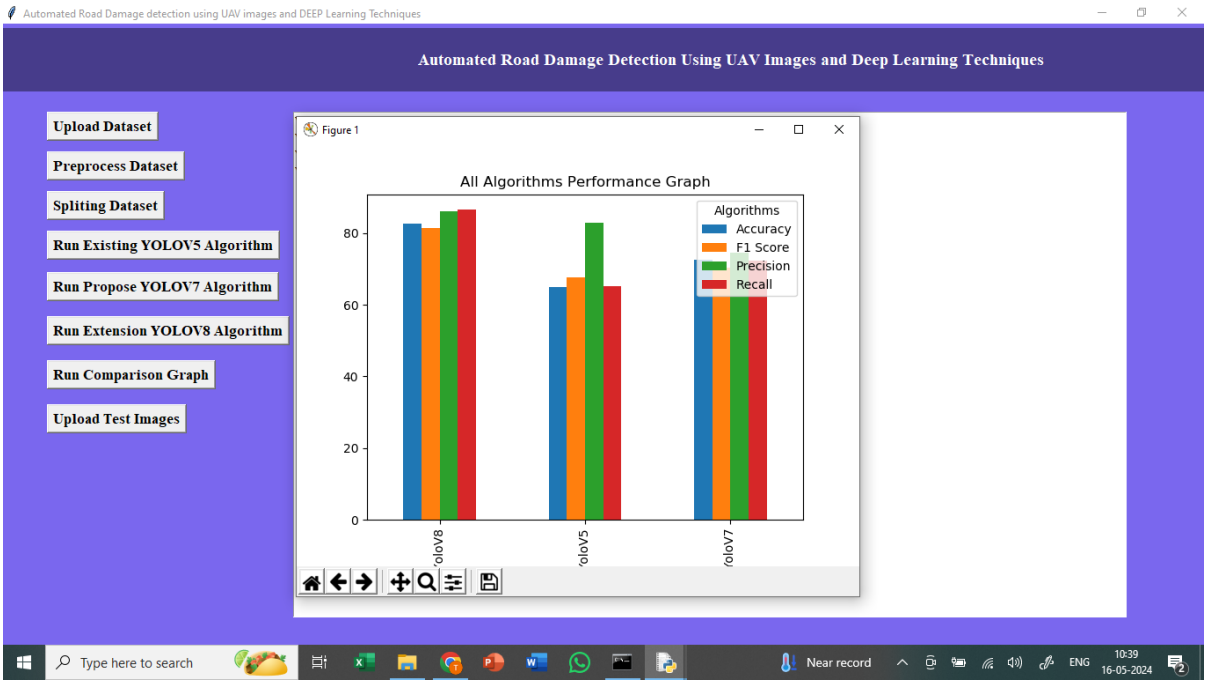

Fig 8.1.9 Comparison graph

In above graph displaying comparison between all algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars

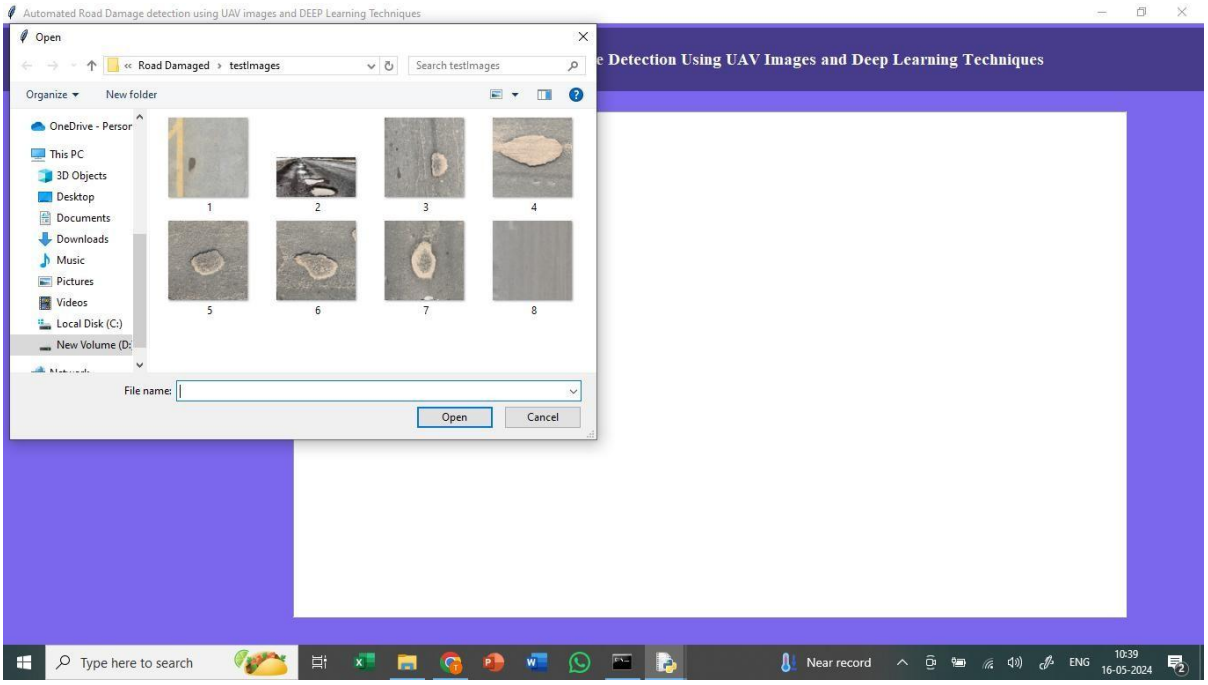

Fig 8.1.10 Upload Image

In above screen uploading road damaged image from test images folder

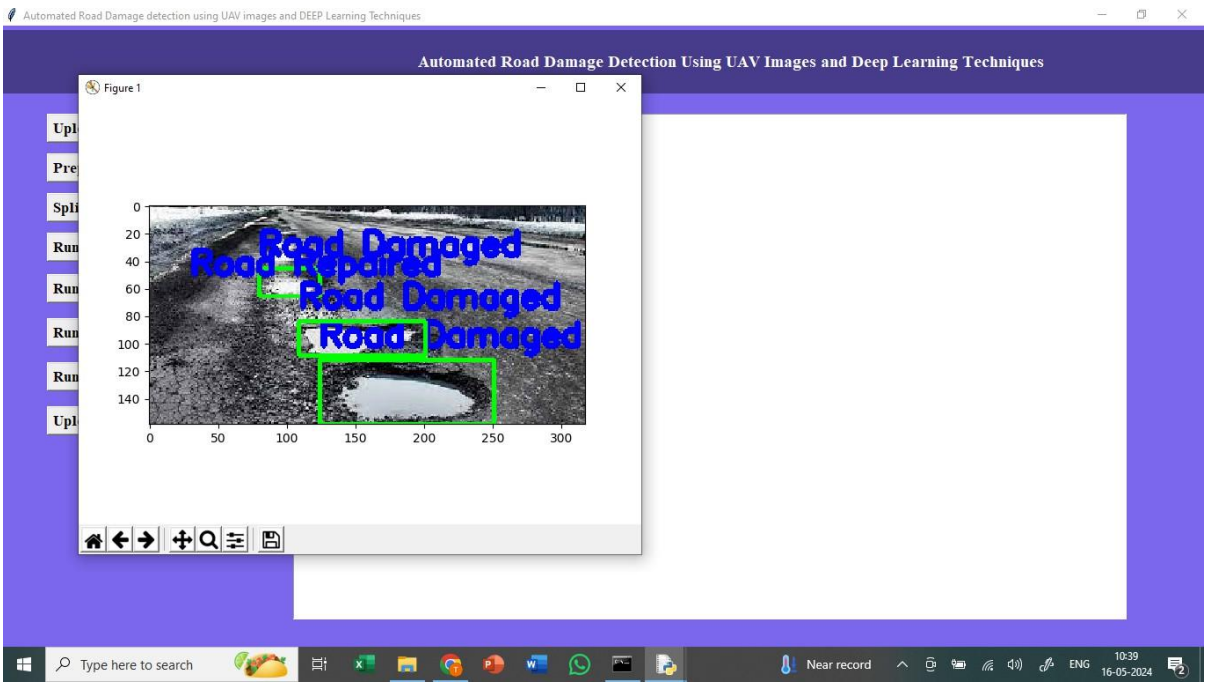

Fig 8.1.11 Pithole detection

In above screen with bounding boxes damage road detected from given test input images

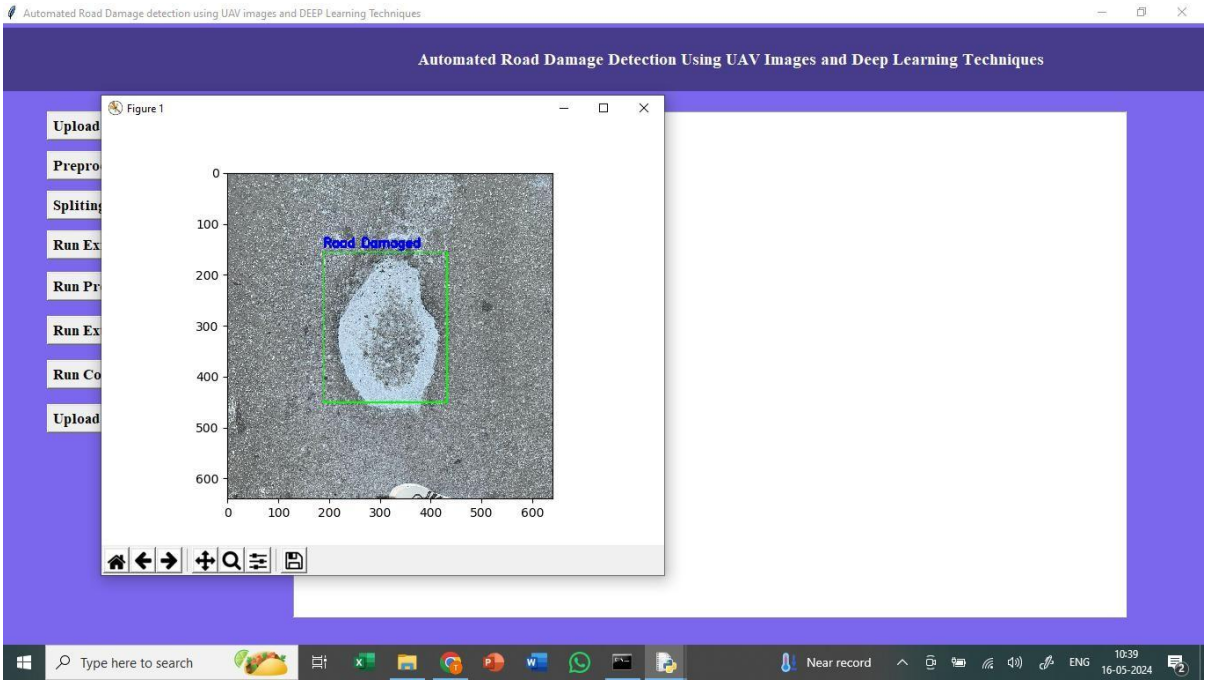

Fig 8.1.12 Road damage detection

In above screen we can see other tested images

CHAPTER-9
CONCLUSION

9.1 CONCLUSION

This project demonstrated the effectiveness of advanced YOLO architectures for road damage detection using UAV images. Through a comparative analysis of YOLOv4, YOLOv5, YOLOv7, and YOLOv8, alongside an implementation of YOLOv5 with Transformer networks, significant improvements were observed in detection accuracy. YOLOv8 outperformed the previous versions, achieving a **mAP.5 of 78.5%**, highlighting its superior capability in identifying road damage patterns. The integration of a UAV image database with the **RDD2022 dataset** played a crucial role in refining model performance, particularly in addressing class imbalances for potholes and alligator cracks.

Despite these advancements, opportunities for further refinement remain. One promising direction involves **multispectral imaging and LIDAR sensors**, which could provide richer data for damage classification. The fusion of such technologies with **embedded computing** may enable real-time processing, reducing the need for manual inspections and increasing operational efficiency. Another potential approach is the use of **fixed-wing UAVs**, which may offer broader coverage and faster data collection for large-scale infrastructure monitoring.

This research lays a strong foundation for future advancements in road damage detection. By leveraging deep learning and UAV-based imaging, municipalities and infrastructure maintenance teams can enhance their ability to assess and address road damage proactively. Continued research into hybrid architectures, specialized datasets, and real-time computing will further elevate detection capabilities, contributing to **safer and more sustainable road systems** worldwide.

CHAPTER-10
FUTURE ENHANCEMENTS

10.1 FUTURE ENHANCEMENTS

Future enhancements for the road damage detection system include integrating advanced sensors like LIDAR and multispectral imaging for enriched data, adopting emerging architectures such as Transformer-based YOLO models for improved accuracy, and expanding the UAV image database with diverse datasets across varied conditions to enhance model robustness. Exploring real-time processing capabilities on embedded systems, combining UAV data with ground-based sensors, and utilizing fixed-wing UAVs for broader coverage could optimize scalability and efficiency. Further advancements include addressing class imbalance with innovative techniques, using GANs for rare damage type generation, and incorporating sustainability measures like energy-efficient models to reduce costs. Collaborating with policymakers and researchers can foster community-based contributions and promote adoption in infrastructure maintenance globally.

1. Integration of Advanced Sensor Data

- **Incorporating Multispectral and Hyperspectral Images**: These can provide additional spectral information that may improve damage detection and classification accuracy.
- **Using LIDAR Data**: LIDAR sensors can add depth information, aiding in the precise identification of cracks, potholes, and other surface deformations.
- **Sensor Fusion**: Combining data from UAV cameras, multispectral sensors, and LIDAR could further improve detection by leveraging complementary features.

2. Experimenting with Advanced Architectures

- **Transformer-based YOLO Versions**: YOLO models with attention mechanisms (e.g., Transformers) could be explored for even better performance in complex road damage scenarios.
- **Application of YOLOv9 (if available)**: As the YOLO family evolves, newer versions with potentially better capabilities could enhance detection accuracy.
- **Integration with GANs (Generative Adversarial Networks)**: GANs can be used for data augmentation, particularly for rare damage types, to address class imbalance issues.

3. Enhanced Dataset Development

- **Expanding the UAV Image Database**: Collecting more UAV images under varying weather, lighting, and geographical conditions will improve the robustness of the model.
- **Crowdsourced Damage Reporting**: Engaging local communities to contribute images of road damage for training and validation datasets.
- **Including Diverse Road Types**: Extending datasets to include rural, urban, and highway roads across various terrains globally.

4. Real-time Damage Detection System

- **Onboard Processing with Edge Computing**: Deploying models on embedded systems, such as NVIDIA Jetson or Raspberry Pi, can enable real-time analysis directly on UAVs.
- **Low-latency Systems**: Optimizing the architecture for faster inference, ensuring real-time detection during UAV flights.

5. Scalability and Deployment

- **Integration with Public Infrastructure Systems**: Collaborating with local governments to deploy UAV-based road inspection for public works maintenance.
- **Automated Repair Assistance**: Combining the detection system with robotics to automate road repairs, starting with simple damage types like potholes.

6. Adapting to Diverse Use Cases

- **Monitoring Other Infrastructure**: The architecture could be adapted for bridges, railways, or airport runway inspection, broadening its applicability.
- **Flood and Disaster Damage Assessment**: Extending the model's capabilities to identify and quantify damage caused by natural disasters.
- **Use of Fixed-wing UAVs**: Fixed-wing drones can cover larger areas more efficiently, which would be beneficial for highway or rural road monitoring.

7. Collaborative Research Directions

- **Open Benchmarking Platforms**: Creating an open platform where researchers can benchmark various YOLO architectures on standardized road damage datasets.

- **Partnership with AI and UAV Innovators**: Collaborating with tech companies and UAV manufacturers to develop customized hardware-software solutions.

8. Improved Handling of Edge Cases

- **Addressing Rare Damage Types**: Enhancing model accuracy for less common road damages, such as surface shearing or subsidence.
- **Adverse Weather Conditions**: Training the model to perform effectively in rain, fog, or other challenging conditions.
- **Advanced Class Balancing Techniques**: Developing innovative methods to tackle class imbalance issues without sacrificing accuracy.

9. Sustainability and Cost-Effectiveness

- **Energy-efficient Models**: Optimizing architectures to reduce the computational cost for real-time, scalable deployments.
- **Reducing UAV Dependency**: Exploring alternatives, such as ground-based robotic systems or hybrid UAV-ground systems, for cost-effective implementation.

10. Community and Policy Integration

- **Policy Advocacy**: Promoting the use of such systems for sustainable infrastructure maintenance in urban and rural development policies.
- **Community Awareness Programs**: Educating the public about the importance of timely road damage detection and the role of UAV technology.

This roadmap sets the stage for a robust, innovative future in automated road damage detection and management. Incorporating these advancements can enhance the reliability, accuracy, and scalability of such systems, further solidifying their role in global infrastructure improvement.

CHAPTER-11
REFERENCES

11.1 REFERENCES

[1] H. S. S. Blas, A. C. Balea, A. S. Mendes, L. A. Silva, and G. V. González, ‘‘A platform for swimming pool detection and legal verification using a multi-agent system and remote image sensing,’’ Int. J. Interact. Multimedia Artif. Intell., vol. 2023, pp. 1–13, Jan. 2023.

[2] A. Safonova, Y. Hamad, A. Alekhina, and D. Kaplun, ‘‘Detection of Norway spruce trees (Picea abies) infested by bark beetle in UAV images using YOLOs architectures,’’ IEEE Access, vol. 10, pp. 10384–10392, 2022.

[3] L. A. Silva, A. S. Mendes, H. S. S. Blas, L. C. Bastos, A. L. Gonçalves, and A. F. de Moraes, ‘‘Active actions in the extraction of urban objects for information quality and knowledge recommendation with machine learning,’’ Sensors, vol. 23, no. 1, p. 138, Dec. 2022, doi: 10.3390/ s23010138.

[4] M. Guerrieri and G. Parla, ‘‘Flexible and stone pavements distress detection and measurement by deep learning and low-cost detection devices,’’ Eng. Failure Anal., vol. 141, Nov. 2022, Art. no. 106714, doi: 10.1016/j.engfailanal.2022.106714.

[5] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, K. Michael, J. Fang, C. Wong, D. Montes, Z. Wang, C. Fati, J. Nadar, V. Sonck, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, ‘‘Ultralytics/YOLOv5: V7.0—YOLOv5 SOTA realtime instance segmentation,’’ Zenodo, Tech. Rep., Nov. 2022. [Online]. Available: https://zenodo.org/record/7347926

[6] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, and Y. Sekimoto, ‘‘RDD2022: A multi-national image dataset for automatic road damage detection,’’ 2022, arXiv:2209.08538.

[7] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, ‘‘YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,’’ 2022, arXiv:220.

[8] L. A. Silva, H. S. S. Blas, D. P. García, A. S. Mendes, and G. V. González, ‘‘An architectural multi-agent system for a pavement monitoring system with pothole recognition in UAV images,’’ Sensors, vol. 20, no. 21, p. 6205, Oct. 2020, doi: 10.3390/s20216.

[9] Y. Bhatia, R. Rai, V. Gupta, N. Aggarwal, and A. Akula, ‘‘Convolutional neural networks based potholes detection using thermal imaging,’’ J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 3, pp. 578–588, Mar. 2022, doi: 10.1016/j.jksuci.2019.02.004.