A

**Project Report**

on

**DETECTION OF POTHOLES USING YOLO ALGORITHM**

Submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU**

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

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

|  |  |
| --- | --- |
| **209E1A05H7** | **P DURGA VAMSI KRISHNA** |
| **209E1A05G4** | **P SRIKANTH** |
| **209E1A05H2** | **P YASWANTH REDDY** |
| **209E1A05G9** | **P MANOJ KUMAR** |

**Under the Esteemed Guidance of**

**Mrs. P.Lakshmi, M.Tech.,**

**Assistant Professor**



**Department of Computer Science and Engineering SRI VENKATESWARA ENGINEERING COLLEGE**

**(Approved by AICTE, New Delhi,NBA & NAAC Accredited Institution with UGC Section 2(f) and 12(b) & Affiliated to JNTUA, Ananthapuramu)**

**Karakambadi Road, TIRUPATI 517507**

**2020-2024**

**SRI VENKATESWARA ENGINEERING COLLEGE**

**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)Karakambadi Road, TIRUPATI** – **517507 DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

***This is to certify that the project report entitled* “DETECTION OF POTHOLES USING YOLO ALGORITHM” *a bonafide record of the project work done and submitted by***

|  |  |
| --- | --- |
| **209E1A05H7** | **P DURGA VAMSI KRISHNA** |
| **209E1A05G4** | **P SRIKANTH** |
| **209E1A05H2** | **P YASWANTH REDDY** |
| **209E1A05G9** | **P MANOJ KUMAR** |

*for the partial fulfillment of the requirements for the award of B.Tech Degree in* **COMPUTER SCIENCE AND ENGINEERING,** *JNTUA, Ananthapuramu during the year 2020-2024.*

# GUIDE: Head of the Department:

**Mrs. P. LAKSHMI Dr. K. SANTHI**

Assistant Professor Professor & HOD

Dept of CSE Dept of CSE

### External Viva-Voce Exam Held on

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION**

We hereby declare that the project report entitled **“DETECTION OF POTHOLES USING YOLO ALGORITHM”** done by us under the guidance of **P Lakshmi,** and is submitted in partial fulfillment of the requirements for the award of the Bachelor’s degree in **Computer Science and Engineering.** This project is the result of our own effort and it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above.

**P DURGA VAMSI KRISHNA (209E1A05H7) P SRIKANTH (209E1A05G4)**

**P YASWANTH REDDY (209E1A05H2)**

**P MANOJ KUMAR (209E1A05G9)**

**ACKNOWLEDGEMENT**

We are thankful to our guide **Mrs. P. Lakshmi** for her valuable guidance and encouragement. Her helping attitude and suggestions have helped us in the successful completion of the project.

We would like to express our gratefulness and sincere thanks to **Dr. K. Santhi,** Head of the Department of **COMPUTER SCIENCE AND ENGINEERING**, for her kind help and encouragement during the course of our study and in the successful completion of the project work.

We have great pleasure in expressing our hearty thanks to our beloved Principal

**Dr. C. Chandrasekhar,** for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank to the **Management** for providing all the necessary facilities during the course of study.

We would like to thank our parents and friends, who have the greatest contributions in all our achievements, for the great care and blessings in making us successful in all our endeavors.

**P DURGA VAMSI KRISHNA (209E1A05H7) P SRIKANTH (209E1A05G4) P YASWANTH REDDY (209E1A05H2) P MANOJ KUMAR (209E1A05G9)**

# CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **DESCRIPTION** | **PAGE NO.** |
|  | **Abstract**  **List of Figures List of Tables**  **List of Abbreviations** | **i ii iii iv** |
| **CHAPTER NO.** | **DESCRIPTION** | **PAGE NO.** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **PROJECT DESCRIPTION** | **2** |
|  | 2.1 PROBLEM DEFINITION | 2 |
|  | 2.2 PROJECT DETAILS | 3 |
| **3** | **COMPUTATIONAL ENVIRONMENT** | **9** |
|  | 3.1 SOFTWARE REQUIREMENTS | 9 |
|  | 3.2 HARDWARE REQUIREMENTS | 9 |
|  | 3.3 SOFTWARE FEATURES | 9 |
|  | 3.3.1 METHODOLOGY AND ALGORITHMS | 20 |
|  | 3.3.2 DATASET PRE-PREOCESSING | 22 |
|  | 3.3.3 TRAINING OF YOLOv8 ALGORITHM | 23 |
| **4** | **FEASIBILITY STUDY** | **25** |
|  | 4.1 TECHNICAL FEASIBILITY | 25 |
|  | 4.2 SOCIAL FEASIBILITY | 25 |
|  | 4.3 ECONOMICAL FEASIBILITY | 26 |

1. SYSTEM ANALYSIS 27
   1. EXISTING SYSTEM 27
   2. PROPOSED SYSTEM 27
2. SYSTEM DESIGN 29
   1. [INPUT DESIGN 29](#_TOC_250008)
      1. OBJECTIVES FOR INPUT DESIGN 30
   2. [OUTPUT DESIGN 30](#_TOC_250007)
      1. OBJECTIVES FOR OUTPUT DESIGN 30
   3. [UML DIAGRAMS 30](#_TOC_250006)
      1. [CLASS DIAGRAM 32](#_TOC_250005)
      2. [USE CASE DIAGRAM 33](#_TOC_250004)
      3. [SEQUENCE DIAGRAM 36](#_TOC_250003)
      4. [COLLABORATION DIAGRAM 38](#_TOC_250002)
      5. [ACTIVITY DIAGRAM 38](#_TOC_250001)
      6. [DEPLOYMENT DIAGRAM 40](#_TOC_250000)
3. SYSTEM IMPLEMENTATION 41
4. TESTING 47
5. SCREEN LAYOUTS 54
6. CONCLUSION 63
7. FUTURE SCOPE 64
8. REFERENCES 65

**ABSTRACT**

The purpose of this project is to develop a pothole identification system that could detect potholes both during the day and at night. This model was created using the YOLO, or You Only Look Once, method for real-time object recognition. The pothole is detected by a pre- trained model using **YOLOv8**. Videos as well as images are collected in real time and processed using the YOLOv8 computational image processing technology. This can help to exercise caution when travelling on roads having potholes, which improves public safety and allows the responsible authorities to address the potholes more quickly. Potholes are important when it comes to road safety. about a period of time in India, there have been about 9300 deaths along with 25000 injuries only as a result of potholes, not taking into account other factors that caused accidents into account. When driving for lengthy periods of time while under stress or strain, the driver loses attention both during the day and at night. The pothole detecting system can be used to avoid accidents from occurring. When YOLO first debuted, It was the first model to combine categorization of objects and restricted space prediction under a single end- to-end distinguishable network. It was created and is maintained under the Darknet framework. YOLOv8 is the initial YOLO model created using the PyTorch framework, and it is a lot simpler and lighter to use. When active, this system comprises of a camera that extracts pictures from live camera recordings in order to detect potholes. As with real-time object detection systems, it will show potholes in real-time emphasize them with boxes. This technique has an accuracy of 80- 90%.

**P DURGA VAMSI KRISHNA (209E1A05H7) P SRIKANTH (209E1A05G4) P YASWANTH REDDY (209E1A05H2) P MANOJ KUMAR (209E1A05G9)**

|  |  |  |  |
| --- | --- | --- | --- |
| **S .NO.** | **FIGURE NO**. | **DESCRIPTION** | **PAGE NO.** |
| **1** | Fig 1 | Accidents Information | 1 |
| **2** | Fig 3.3.1 | Methodology Flowchart | 24 |
| **3** | Fig 5.1 | System block design of this project | 28 |
| **4** | Fig 6.3.1 | Class Diagram | 33 |
| **5** | Fig 6.3.2 | Use Case Diagram | 34 |
| **6** | Fig 6.3.3 | Sequence Diagram | 37 |
| **7** | Fig 6.3.4 | Collaboration Diagram | 38 |
| **8** | Fig 6.3.5 | Activity Diagram | 40 |
| **9** | Fig 6.3.6 | Deployment Diagram | 40 |

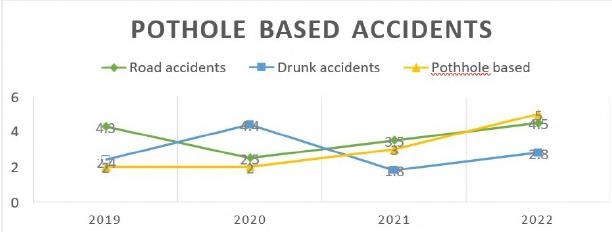
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Table No.** | **Description** | **Page No** |
| **1** | **3.3.1** | Breakdown of the Training and Validation Dataset | **22** |
| **2** | **8.4.1** | Test cases format | **51** |
| **3** | **8.4.2** | Test cases model building | **51** |

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME** | **ABBREVIATION** |
| **1** | YOLO | You Only Look Once |
| **2** | IDE | Integrated Development Environment |
| **3** | IDLE | Integrated Development Learning Environment |
| **4** | CNN | Convolutional Neural Network |
| **5** | UML | Unified Modelling Language |

# CHAPTER 1 INTRODUCTION

In many nations, a sizable amount of passenger traffic is transported on roads, making them an essential method of transportation. Unfortunately, many of the roads in developing nations are congested, narrow, badly maintained, and have poor surface quality, which leads to the production of potholes. These imperfections in the road surface can be hazardous, especially if they are submerged in water during the rainy season and could result in deadly accidents. It can be difficult for authorities to locate and fix potholes.

Many Indian statistics reveal that more than 50% of accidents occur during night time. The prime factors for more accidents during nighttime included poor visibility of the streets to the driver, driver’s visual fatigue & performance, etc.,



**Fig 1:** Accidents Information

Over a period, around 9300 fatalities and 25000 injuries in India were solely attributable to potholes, disregarding any other object that may have contributed to the incidents. Driving for a lengthy period or while under stress or strain can make it difficult for the driver to focus both during the day and at night. The pothole detecting system can be used to stop accidents from happening.

There are numerous technical methods for detecting potholes, including thermal imaging, vibration sensor technology, scanning with 3D reconstruction, and computer vision. The ability of computer vision and image processing-based techniques to replace obsolete human inspection methods for pothole identification has led to an increase in their use in recent years. This is owing to the accessibility, affordability, and practicality of cameras. Using a camera is a practical way to implement a pothole detection system, and the YOLOv8 image processing technique has been created to help with this. Potholes can be recognized more quickly and there is more public safety on roads with potholes thanks to the real-time photographs and videos that are taken. Potholes must therefore be considered while discussing road safety.

# CHAPTER 2 PROJECT DESCRIPTION

## PROBLEM DEFINITION

Potholes are a common issue in road infrastructure, posing risks to motorists and pedestrians while also contributing to increased maintenance costs for road authorities. Due to this it is difficult for the driver to have control of the vehicle when he sees a 10 obstacle (like other vehicles, animals) etc. When crossing a certain speed limit with uneven roads mainly potholes it is most likely that he loses control of the vehicle. This leads to serious injuries or sometimes fatal death. Traditional methods of identifying potholes rely heavily on manual inspections, which are time-consuming, expensive, and often inadequate for effectively addressing the problem at scale. Therefore, there is a pressing need for automated solutions that can detect potholes efficiently and accurately.

This project aims to develop an automated pothole detection system using computer vision techniques, with a focus on implementing the You Only Look Once (YOLO) algorithm. By leveraging machine learning and image processing, the system will analyze images or video streams captured by cameras mounted on vehicles or fixed positions along roadsides. The primary objective is to create a robust algorithm capable of identifying and localizing potholes in real-time, enabling timely repairs and maintenance interventions. Additionally, the system should be scalable and adaptable to various environmental conditions, road types, and infrastructure setups to ensure widespread applicability and effectiveness in mitigating the hazards posed by potholes.

## PROJECT DETAILS

Potholes present a pervasive problem in road infrastructure, posing safety hazards to motorists and pedestrians and incurring substantial maintenance costs for road authorities. Manual methods of pothole detection are labor-intensive, time- consuming, and often insufficient for timely intervention, especially in urban areas with extensive road networks. To address this challenge, this project focuses on developing an automated pothole detection system leveraging the You Only Look Once (YOLO) algorithm, a state-of-the-art object detection algorithm known for its speed and accuracy.

The core of the project revolves around adapting the YOLO algorithm for pothole detection. YOLO offers a unified framework for object detection, providing real-time performance by simultaneously predicting bounding boxes and class probabilities for multiple objects within an image. By training the YOLO model on a dataset of annotated road images or video frames containing potholes, the algorithm can learn to accurately identify and localize potholes in diverse environmental conditions.

A crucial aspect of training the YOLO model is the availability of a high- quality annotated dataset. The project involves collecting a diverse dataset of road images or video footage depicting various road conditions and pothole instances. These images or videos are meticulously annotated to highlight the locations and boundaries of potholes, enabling the algorithm to learn from labeled data during the training process.

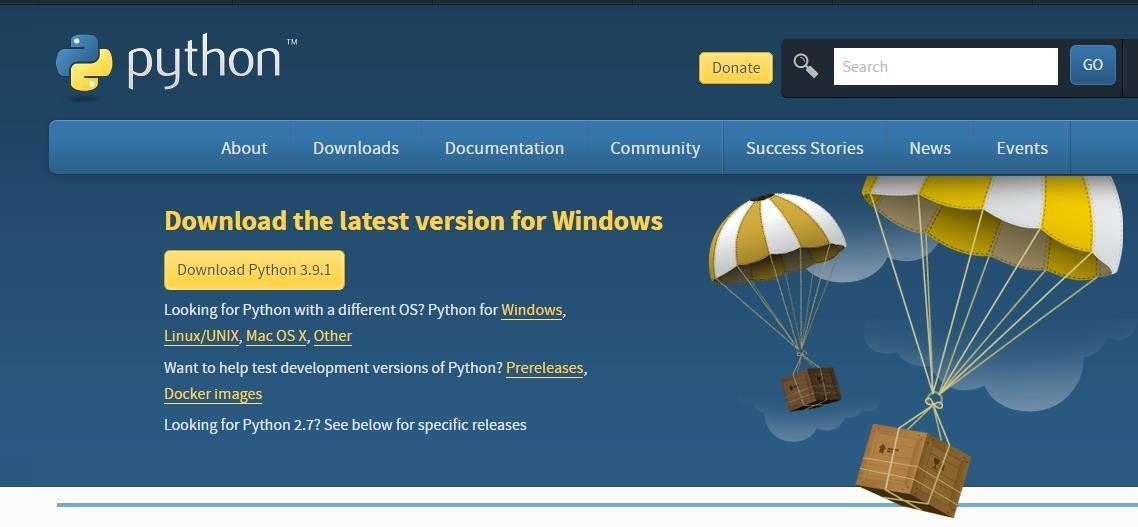
Once the dataset is prepared, the YOLO algorithm undergoes training to learn the characteristics of potholes and optimize its parameters for accurate detection. The training process involves iteratively adjusting the model's weights and biases to minimize detection errors and maximize performance metrics such as precision, recall, and **mean average precision (mAP)**.

The project on pothole detection using the YOLO algorithm represents a significant step towards addressing the challenges associated with manual pothole inspection and maintenance. By leveraging advanced computer vision techniques, the automated system offers a cost-effective and efficient solution for identifying and addressing road defects, ultimately contributing to improved road safety and infrastructure management.

## SOFTWARE INSTALLATION FOR DEEP LEARNING PROJECTS:

### Installing Python:

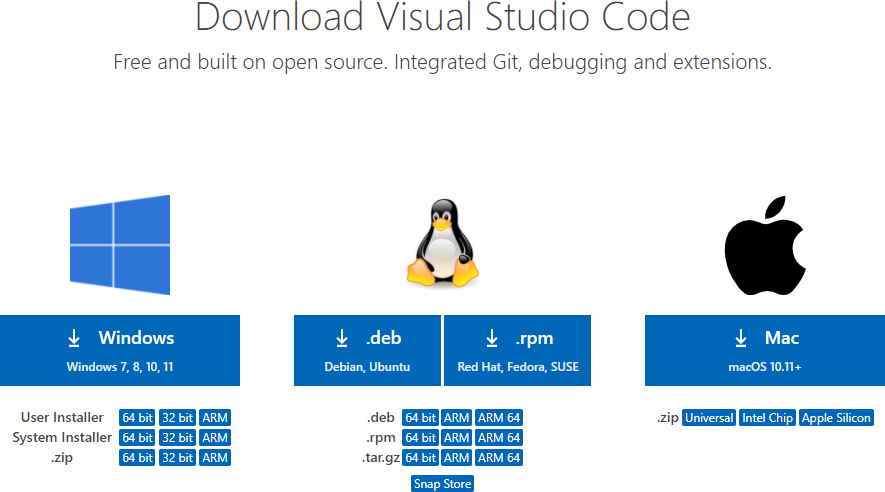
1. To download and install Python visit the official website of python <https://www.python.org/downloads/> and choose your version



1. Once the download is complete, run the **exe** for install Python. Now click on Install Now
2. You can see Python installing at this point
3. When it finishes, you can see a screen that says the Setup was successful. N ow click on "Close"

### Installing Visual Studio Code:

1. Visit the Official Website of the Visual Studio Code using any web browser like Google Chrome, Microsoft Edge, etc.
2. Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application. Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application. Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.
3. Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.



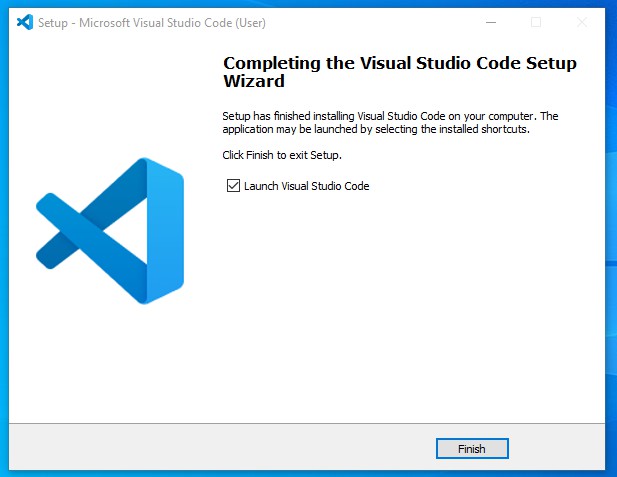
1. When the download finishes, then the Visual Studio Code Icon appears in the downloads folder.



1. Click on the Installer icon to start the installation process of the Visual Studio Code.
2. After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

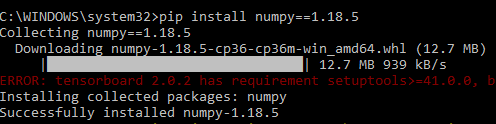


1. Then it will ask to begin the installation setup. Click on the Install button.
2. After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.
3. After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “Launch Visual Studio Code” checkbox and then click Next.



1. After the previous step, the Visual Studio Code window opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!
2. You need to install some packages to execute project in a proper way.
3. Open the command prompt/anaconda prompt or terminal as administrator.
4. The prompt will get open, with specified path, type “pip install package name” which you want to install (like NumPy, pandas, sea born, scikit- learn, Matplotlib, Pyplot, OpenCV2)

Ex: Pip install numpy



## SYSTEM REQUIREMENTS SPECIFICATION

### Functional and non-functional requirements:

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non- functional requirements.

### Functional Requirements:

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily

incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she registers for the first time on some software system.

### Non-functional requirements:

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. They basically deal with issues like:

* + Portability
  + Security
  + Maintainability
  + Reliability
  + Scalability
  + Performance
  + Reusability
  + Flexibility

Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are > 10000

# CHAPTER 3 COMPUTATIONAL ENVIRONMENT

## SOFTWARE SPECIFICATION

* + - Operating system : Windows 10 or above
    - Programming Language : Python 3
    - Platform : Google Colab
    - Technology : Computer Vison

## HARDWARE SPECIFICATION

* + - System : Pentium IV 2.4 GHz (min)
    - Hard Disk : 40 GB.
    - Ram : 512 MB

## SOFTWARE FEATURES

Python, a high-level programming language, offers a plethora of features that contribute to its popularity among developers. One prominent feature is its simplicity and readability, making it an ideal choice for beginners and seasoned programmers alike. Python's syntax emphasizes` code readability, with clear and concise code structures that are easy to understand and maintain.

Another key feature of Python is its versatility and cross-platform compatibility. Python supports multiple operating systems, including Windows, macOS, and Linux, allowing developers to write code once and deploy it across different platforms without modification. This flexibility makes Python suitable for a wide range of applications, from web development to scientific computing and data analysis.

Python's extensive standard library is another notable feature that simplifies development by providing a rich set of pre-built modules and functions. These modules cover a wide range of tasks, including file I/O, networking, database access, and more, enabling developers to leverage existing code and resources to expedite development and reduce time-to-market.

Furthermore, Python's dynamic typing and automatic memory management contribute to its ease of use and productivity. Unlike statically-typed languages, Python does not require variable declarations, allowing developers to focus on solving problems rather than managing data types. Additionally, Python's garbage collection

mechanism automatically deallocates memory when objects are no longer in use, eliminating the need for manual memory management and reducing the risk of memory leaks and other memory-related issues.

Python's support for object-oriented programming (OOP) is another valuable feature that enables developers to write modular, reusable code. Python allows for the creation of classes and objects, encapsulating data and behaviour into cohesive units that can be easily extended and modified. This facilitates code organization, promotes code reuse, and enhances maintainability, particularly for large-scale projects.

Moreover, Python's extensive ecosystem of third-party libraries and frameworks further extends its capabilities, catering to a wide range of development needs. From web frameworks like Django and Flask to data science libraries like NumPy and pandas, Python offers a vast array of tools and resources that empower developers to tackle diverse challenges and build robust, feature-rich applications.

Overall, Python's simplicity, versatility, extensive standard library, dynamic typing, object-oriented programming support, and rich ecosystem of third-party libraries make it a compelling choice for developers across various domains and industries. Its combination of ease of use, flexibility, and powerful features continues to drive its widespread adoption and popularity in the software development community.

### Why Python

Python is favoured by developers for several reasons, including its simplicity, versatility, extensive libraries, and vibrant community support.

One of Python's most appealing aspects is its readability and ease of use. Python's syntax is designed to be clear and concise, making it accessible to beginners and experienced programmers alike. Its simple and straightforward syntax reduces the time required for development and debugging, enabling developers to focus more on solving problems rather than dealing with complex syntax rules.

Python's versatility is another major factor driving its popularity. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, giving developers the flexibility to choose the most suitable approach for their projects. Additionally, Python's cross-platform compatibility allows code to run seamlessly on various operating systems, further enhancing its versatility and practicality.

Furthermore, Python boasts an extensive standard library, providing a wide range of modules and functions for tasks such as file I/O, networking, database

access, and more. This rich set of built-in tools accelerates development by eliminating the need to reinvent the wheel and simplifying common programming tasks.

Python's dynamic typing and automatic memory management contribute to its ease of use and productivity. Unlike statically-typed languages, Python does not require variable declarations, allowing developers to focus on solving problems rather than managing data types. Additionally, Python's garbage collection mechanism automatically deallocates memory when objects are no longer in use, reducing the risk of memory leaks and other memory-related issues.

Moreover, Python boasts a vibrant and supportive community of developers, educators, and enthusiasts who actively contribute to its growth and development. The Python community is known for its collaborative spirit, extensive documentation, and wealth of online resources, including tutorials, forums, and user groups. This thriving community fosters knowledge sharing, innovation, and continuous improvement, making Python an attractive choice for developers seeking community support and collaboration.

Overall, Python's simplicity, versatility, extensive libraries, dynamic typing, automatic memory management, and vibrant community support make it a compelling choice for a wide range of applications, from web development and data analysis to scientific computing and artificial intelligence. Its combination of ease of use, flexibility, and powerful features continues to drive its widespread adoption and popularity in the software development community.

### The Python Programming Language

Python is a high-level programming language known for its simplicity and readability. It's widely used for various purposes including web development, data analysis, artificial intelligence, and scientific computing. One of the key features of Python is its straightforward syntax, which makes it accessible to beginners and experts alike.

Python code is executed line by line, which makes it easy to understand and debug. Let's take a look at some basic Python code snippets:

```python

### # Example 1: Hello World

print("Hello, World!")

### # Example 2: Variables and Data Types

x = 5

y = "Hello" print(x) print(y)

### # Example 3: Basic Arithmetic Operations

a = 10

b = 5

sum = a + b difference = a - b product = a \* b quotient = a / b print("Sum:", sum)

print("Difference:", difference) print("Product:", product) print("Quotient:", quotient)

### # Example 4: Control Structures - if-else

num = 10 if num > 0:

print("Positive number") elif num == 0: print("Zero")

else:

print("Negative number")

**# Example 5: Loops - for loop** fruits = ["apple", "banana", "cherry"] for fruit in fruits:

print(fruit)

### # Example 6: Functions

def greet(name):

print("Hello, " + name) greet("Alice") greet("Bob")

# Example 7: Lists my\_list = [1, 2, 3, 4, 5]

print("Length of list:", len(my\_list)) print("First element:", my\_list[0]) print("Last element:", my\_list[-1])

```

These examples cover some of the fundamental aspects of Python, including printing to the console, variable declaration, basic arithmetic operations, control structures (if- else), loops (for loop), defining functions, and working with lists. Python's versatility and simplicity make it an excellent choice for both beginners and experienced programmers alike.

### The Python Platform

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware.

Python is a versatile programming language that can be executed on various platforms, including Windows, macOS, and Linux. It's an interpreted language, meaning that Python code is executed by an interpreter, which converts the code into machine-readable instructions.

Python is supported by a wide range of integrated development environments (IDEs), text editors, and online platforms. Some popular Python development environments include:

1. PyCharm: Developed by JetBrains, PyCharm is a powerful IDE with features like code completion, debugging, and version control integration.
2. Visual Studio Code (VS Code): A lightweight and extensible code editor from Microsoft, VS Code offers excellent support for Python development through extensions.
3. Jupyter Notebook: Ideal for data analysis and scientific computing, Jupyter Notebook allows you to create and share documents containing live code, equations, visualizations, and narrative text.
4. Spyder: This Python IDE is designed for scientific computing and provides features such as a variable explorer, a profiler, and integration with scientific libraries like NumPy and Matplotlib.
5. IDLE: IDLE comes bundled with the Python installation and offers a simple integrated development environment for writing and executing Python code.

In addition to these development environments, Python code can also be executed directly from the command line or terminal using the Python interpreter. This provides a quick and easy way to run Python scripts without the need for a dedicated IDE or text editor.

Python's platform-independent nature, combined with its extensive standard library and vast ecosystem of third-party packages, makes it a popular choice for a wide range of applications, from web development and automation to machine learning and artificial intelligence.

### Install the Python Desktop IDE

We do have many operating systems. Namely:

* + Windows
  + Mac OS X
  + Linux
  + Portable IDE (Windows and Linux)
  + ChromeOS for Individuals and for Education

Choose the operating system and start the procedure of installing any Python IDE. We are using Windows 10 operating system.

## WHAT IS AN IDE?

IDE stands for Integrated Development Environment, which is a type of software that combines many common software developer tools in one single, user-friendly interface. On top of the common features of a code editor, IDEs typically include additional features such as code autocompletion and debugging.

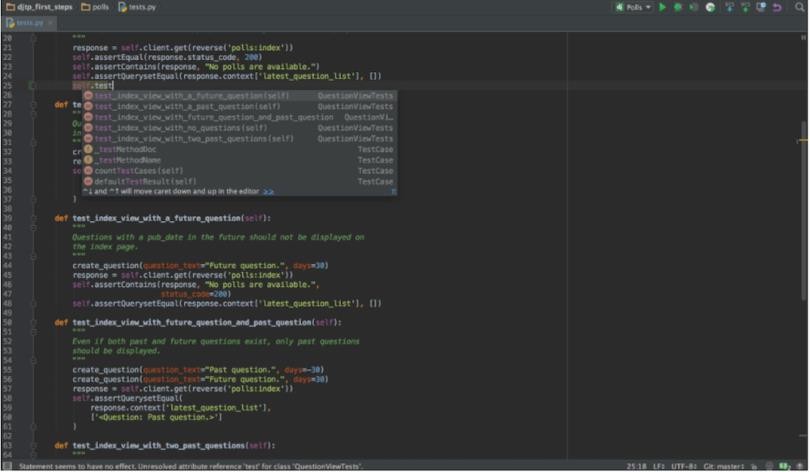
An IDE is a comprehensive tool that you can use throughout the complete process of software development. IDEs provide code building, editing, testing and running capabilities into one central application, so you rarely have to use other programs.

IDEs can also include features like syntax highlighting, code autocompletion and more to improve workflows.

A code editor is typically just a software application used for editing source code, which means that you will likely need some other program to compile/interpret the code and run it. This does give the code editor some advantages though. It can be more lightweight and customizable through plugins and add-ons. A code editor can also be utilized as an individual application or implemented as part of an IDE application.

Here, I have gathered a list of the most widely used and popular Python IDEs and code editors. They aren’t necessarily in best-to-worst order. Instead, I give each a short description to make it easy for you to choose the one that suits your needs best!

1. **PYCHARM (IDE)**



**Fig 3.5: PyCharm IDE**

PyCharm is a Python IDE developed by JetBrains. It is one of the most popular Python IDEs, used by many professional developers. PyCharm comes in two flavors. It is available as a free Apache-licensed community version and as a paid, proprietary professional version.

The community version is likely the better version for you. It comes with features such as syntax highlighting, very robust and intelligent code auto-completion, on-the-fly error highlighting and quick fixes as well as execution and debugging of Python code. PyCharm’s code inspection is one of the most advanced among Python

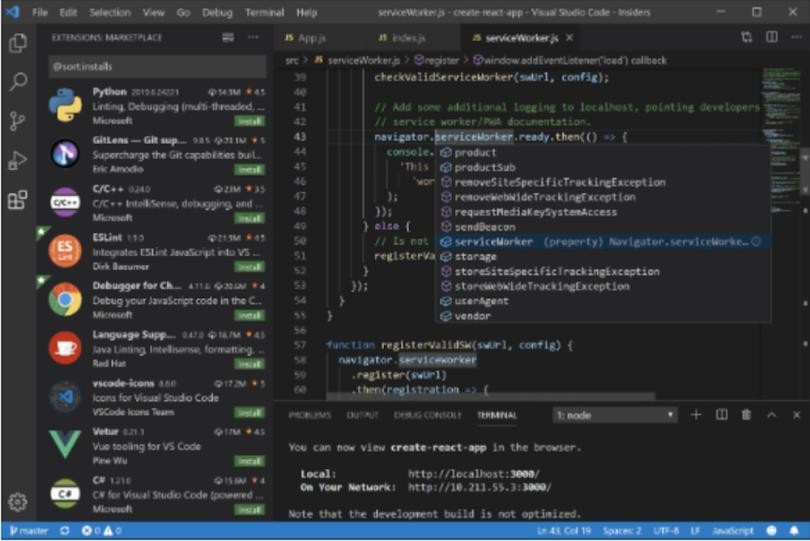
IDEs. Additionally, PyCharm has a very active community, so you are likely to find help when you need it.

The professional version comes with a couple of added advanced features. These include database management and Python web frameworks like Django, Google App Engine, and Pyramid.

The downsides of PyCharm come from its comprehensiveness. It has somewhat long loading times, and you might have to tweak some settings to run existing projects.

All in all, PyCharm is a very good choice for anyone looking for a comprehensive Python development tool. It helps with the quality of the code you write and boosts your efficiency.

## VISUAL STUDIO CODE (CODE EDITOR)

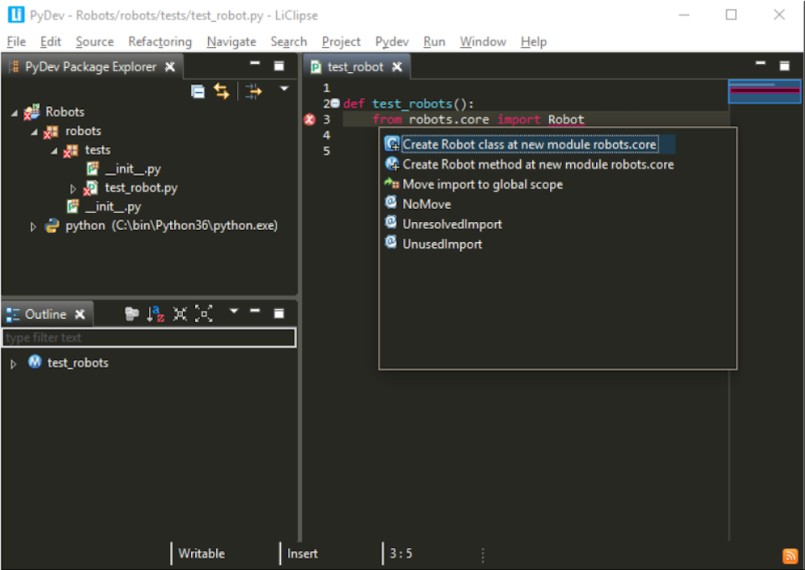


### Fig 3.6: Visual Studio Code

Visual Studio Code (or VS Code for short) is a free and open-source code editor developed by Microsoft. It supports many programming languages, including Python, through an extension. It is relatively lightweight and comes with many useful features like syntax highlighting, very good code autocompletion, Git integration, and advanced code debugging. Visual Studio Code is often confused with Visual Studio. These are different programs, however. VS Code is also very customizable through a

huge number of extensions. It is a good choice if you are looking for a lightweight, fast, and customizable code editor for Python.

## PYDEV (IDE)



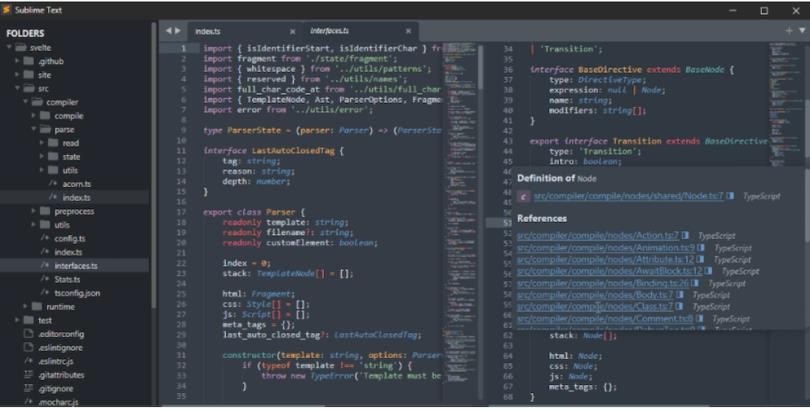
### Fig 3.7 Pydev

Eclipse is a popular IDE designed for software development in Java. Through an extension, however, you can use it for other languages as well, including Python. PyDev is the plugin that allows you to use Eclipse as a Python IDE. Eclipse and PyDev are both free and open-source.

Notable features include syntax highlighting, code autocompletion, refactoring, debugging, code analysis, interactive console, and good support for Python web development.

Eclipse + PyDev is a good choice if you are looking for a complete IDE that is free and open-source. This combination is also great if you need to work on multiple languages. You can use the same IDE for the different languages and have robust features for all of them.

## SUBLIME TEXT (CODE EDITOR)



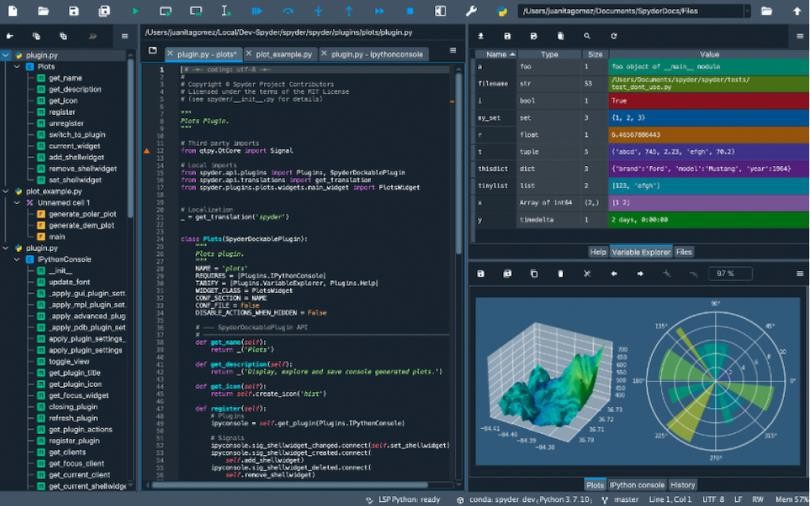
### Fig 3.8 Sublime Text

Sublime Text is a popular code editor with support for multiple programming languages, including Python. According to the site; “Text may be downloaded and evaluated for free, however, a license must be purchased for continued use. There is currently no enforced time limit for the evaluation.”

Features of Sublime Text include high customizability through plugins, speed, minimal, discreet, and powerful user interface, syntax highlighting, code auto- completion, and powerful text editing features.

Sublime Text can be a good choice for you if you are looking for a lightweight code editor that you can customize, and that also has a minimal and powerful interface and text editing features.

## SPYDER (IDE)



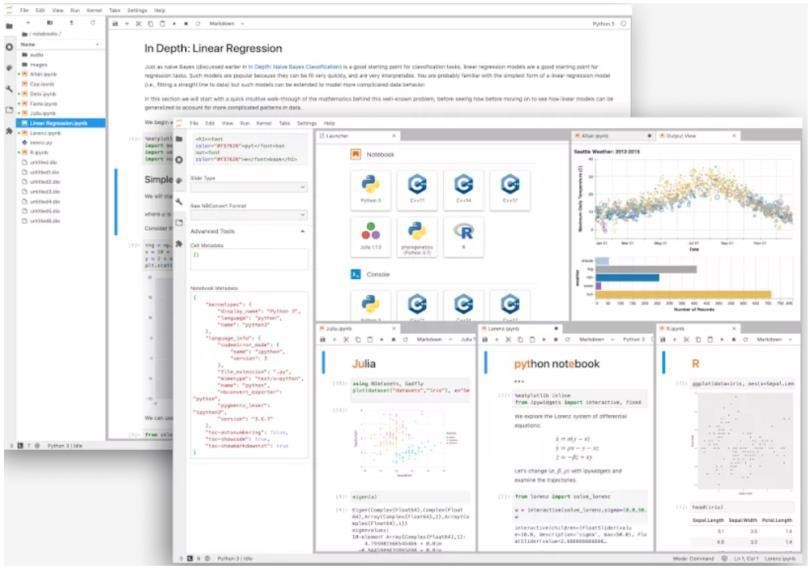
**Fig 3.9** Spyder

Spyder is a free and open-source IDE for Python development. It is commonly used for scientific development. Spyder is also known as Scientific Python Development IDE and is great for data science and machine learning applications. Spyder is also one of the most lightweight IDEs for Python.

Features of Spyder include syntax highlighting, code autocompletion, debugging, code analysis, interactive console, plotting all kinds of charts and graphs, data manipulation, and integration of many data science libraries such as NumPy, Pandas, Scipy, and Matplotlib. Spyder also has great community support.

Spyder is a comprehensive IDE that especially shines in the fields of machine learning or data science.

## JUPYTER NOTEBOOK



### Fig 3.10 Jupyter Notebook

Jupyter Notebook is a web-based interactive development environment. Therefore, you can access it from almost anywhere, from any computer, and continue working on your project as long as you have an internet connection. For this same reason, it is also a great tool for presenting information and sharing your work.

Jupyter Notebook has support for multiple programming languages, including Python. It is also easy to use and open source.

Jupyter Notebook is well known in the data science community for analyzing, sharing, and presenting information. It is great for machine learning, simulation, and data science applications as well as visualizations. Jupyter Notebook is also great if you just need a quick tool to run some Python since you don’t need to set up anything on your computer.

### Methodology and Algorithms

1. Determine potential uses such as pothole detection, road maintenance condition reporting, and many others.
2. Capture images or get images from the internet for the above applications. These photographs will be captured by the camera in real-time. But for initial implementation, already available images will be used.
3. Image processing task: Method or algorithm to detect or identify frames that satisfy the task or goal. For ex, if pothole is the task to be identified, then the algorithm should be able to differentiate/separate images into ones with pothole and neglect other images. This image processing task can be done by image classification, object detection and other techniques. In this work, we are employing YOLO object detection for our tasks. To facilitate the initial training of our object detector, we must employ bounding box annotations. Everything that we desire the detector to see is contained within a box, and each box is labelled with the type of object which we wish it to infer.
4. YOLO object detection and Implementation: Using YOLO has two steps namely

(1) training and (2) detection.

1. Training is giving the YOLO algorithm with sample images and training the algorithm to identify the task. Detection is when you give a new image (not present in the sample image), it should be able to perform the identification task on this unknown image. We are prepared to enter the YOLOv8 training code now that we have a dataset ready. First, we must clone the YOLOv8 repository.
2. Since we can use YOLOv8, we must first install any required dependencies. We are going to prepare the programming surroundings for executing of commands during object identification training by doing so.
3. inference. Once the data is uploaded, you can select preprocessing and augmentation processes, as well as the YOLOv8 PyTorch format.
4. The export generates a data.yaml YOLOv8 file. yaml including details about our custom classes as well as where to find of a YOLOv8 pictures folder, a YOLOv8 tags

folder, and a YOLOv8 labels folder. That is now necessary. To begin training, use yaml files to specify the input image dimensions, batch size, the amount of epochs, and data location.yaml file, provide model parameters, a custom route to weights, and 21 cache pictures for swiftness training.

1. Finally run YOLOV8 Inference on test images and real-time images, verify the result.

### Dataset Pre-processing

* 1. To train the YOLOv8 algorithm for pothole detection, a large dataset of images is needed. All the photos in this collection were taken in the daytime because it is difficult and impractical to take road images at night. As a result, the same testing set has been given an artificial low-lighting effect.
  2. To construct the dataset, images are labelled with the potholes using the Makesens labelling AI tool. A bounding box was used to identify each pothole, and the class label was set to "Pothole." As it will be explained in much more detail below, the dataset was then divided into a training set and a validation set, with 57% of the photos used for training and 43% for validation. Details for the training dataset, validation dataset, and photos taken under various lighting situations are shown in Table 1.
  3. To augment the dataset and improve the model's ability to detect potholes under different conditions, various data augmentation techniques were applied to the training set. These techniques include random cropping, flipping, and color adjustments.

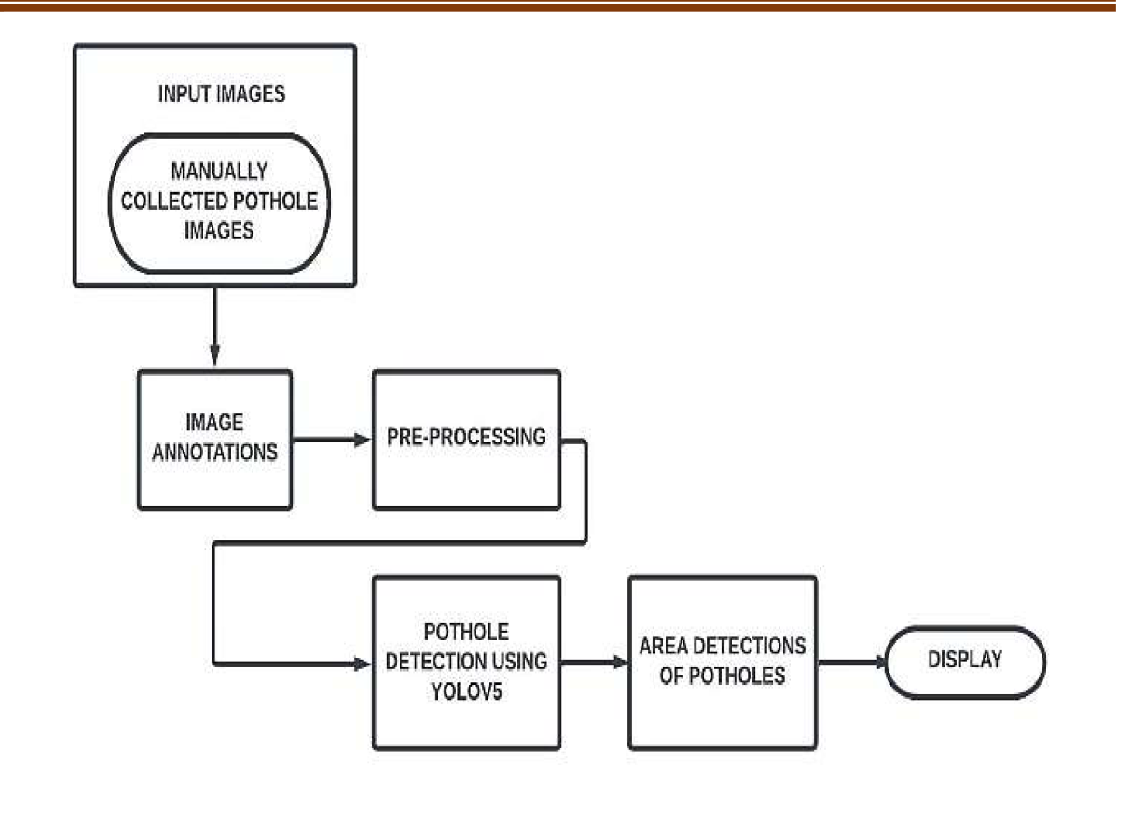
|  |  |
| --- | --- |
| **Index** | **Description** |
| Training Dataset | 200 images |
| Validation Dataset | 150 images |
| Size of image | 640x640 |
| Size of Dataset | 5 MB |
| Lightning level | Normal: 170, Low Light:  180 |
| Total Images | 350 images |

### Table 3.3.1: Breakdown of the Training and Validation Dataset

**Training of YOLOv8 Algorithm**

* The YOLOv8 algorithm was trained on the collected and pre-processed dataset of annotated images that includes potholes as a class.
* for pothole detection. The training process involved the following steps:
* Initialization: Compose and annotate a pothole-detecting image dataset. The annotations contain the class labels and bounding boxes for the potholes.
* Data Augmentation: The training dataset was subjected to data augmentation techniques such as random cropping, rotation, and flipping to avoid overfitting.
* Hyperparameter Tuning: The model's performance is optimized by the hyperparameter's learning rate, batch size, and many epochs to reduce the loss function.
* Training: Using a deep learning framework such as PyTorch, the training was carried out on the annotated dataset for a sufficient number of epochs until the loss function approached convergence.
* Evaluation: Based on the validation dataset, the trained model was assessed to assess its expected accuracy according to Eq. (1) and safeguard against overfitting.
* Accuracy = No of potholes detected in the image / Total no of potholes in the image. (1)
* Fine-tuning: If the model's performance on the validation dataset was not satisfactory, fine-tuning was done by adjusting the hyperparameters and retraining the model.

The YOLOv8 model processes each image, performing object detection and bounding box prediction on each frame of the image. Class labels and their boundary boxes were used to help the YOLOv8 algorithm find potholes. A bounding box is predicted around a pothole if one is found. The model's flow chart is depicted in Fig.3 below. The bounding box is made up of two components: a Class Label and a Predicted Accuracy Percentage, where Accuracy percentage accuracy with which a pothole is identified. The result reflects the number of potholes that were found in each image.



**Fig 3.3.1: Methodology Flowchart**

# CHAPTER 4 FEASIBILITY STUDY

Feasibility study is the initial design stage of any project, which brings together the elements of knowledge that indicate if a project is possible or not. A feasibility study includes an estimate of the level of expertise required for a project and who can provide it, quantitative and qualitative assessments of other essential resources, identification of critical points, a general timetable, and a general cost estimate. Whether a project is viable or not, i.e. whether it can generate an equal or a higher rate of return during its lifetime requires a thorough investigation of the investment per se as well as the level of current expenditure. The preliminary design is the simple description of the conceived idea with an indication of the main factors to be considered in the study.

The feasibility of a proposed solution is evaluated in terms of its components. These components are

* + TECHNICAL FEASIBILITY
  + SOCIAL FEASIBILITY
  + ECONOMICAL FEASIBILITY
  + OPERATIONAL FEASIBILITY

## TECHNICAL FEASIBILITY

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

## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of

confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## ECONOMICAL FEASIBILITY

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

## OPERATION FEASIBILITY

The purpose of this project is to develop software named Criminal Activity Detection in Social Network which facilitates quick allocation process. The activities of the system such as data entry, information retrieval, updating and deletion of records from various tables etc. are made easily. All the operations of this project are trained in this area, so this project is operationally feasible.

# CHAPTER 5 SYSTEM ANALYSIS

## EXISTING SYSTEM

A pothole detection technique is introduced using Location-Aware Convolutional Neural Networks. It highlights the limitations of traditional methods and the potential for deep learning approaches in improving pothole detection. The study proposes a novel location-aware convolutional neural network (LACNN) that considers the spatial relationships of potholes in the surrounding environment. The study's findings demonstrate that the LACNN outperformed other cutting-edge techniques in pothole detection, achieving an F1 score of 0.932. The LACNN was also shown to be robust to variations in lighting and road surface conditions.

The methodology used was FRCNN (faster region based Convolutional neural network). But all the papers entirely focused their work on detection of potholes during daylight. The accidents that are taking place in night are quite high in number, so we decided to focus our work to be detecting potholes during night and alerting drivers of vehicles to avoid accidents using YOLOV8 algorithm

### Drawbacks of Existing System

* Existing system is a two staged algorithm. i.e. Object detection and recognition are two separate processes
* It is very time consuming for real time applications

## PROPOSED SYSTEM

The proposed model for pothole detection in this study utilizes the YOLOV8 algorithm for training and testing. This algorithm involves creating a labelled training set for object detection, which is captured by a camera and processed frame by frame. Each captured image is labelled using a labelling tool, with each object of interest marked by a boundary box.

The conversion of the Darknet research framework to the PyTorch framework is the main contribution of YOLOv8. We need to use bounding box annotations to supervise the learning of our pothole detector. Each object that we want the detector to recognize is enclosed in a box, and each box is labelled with the object class that we want the detector to identify.

For each grid cell, boundary boxes and box ratings of confidence are projected. The bounding boxes are made up of X and Y coordinates that reflect the bounding box's centre, as well as confidence predictions that are used to find and

recognise objects in the image. After labelling, we will separate our data into sets for training and validation.

Pothole diversity, lighting levels, pothole distance from camera, shot angle, image instrument, and meteorological circumstances are just a few of the many variables that might affect pothole detection. For this investigation, we have collected 350 images of 640 x 640 resolution in real-time and tailored the dataset to our requirements. The images are enhanced for better detection of potholes. Few pothole samples of both daytime and nighttime are shown in Fig. 1 and Fig. 2 respectively.

Camera

Image Extraction

Real Time

Image pre-processing using YOLO V8 algorithm

Comparing with custom weights

Display of Potholes

Creates bounding box around the detected region

**Fig 5.1: System block design of this project**

Pothole Detection

### Advantages of Proposed System

This project presents the review on Computer vision based YOLO algorithm to overcome the drawbacks of the traditional initial system like FRCNN. The advantages are as follows:

* Single stage algorithm. i.e. Object detection and recognition happen together
* High accuracy
* Less time consuming

# CHAPTER 6 SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development, then design is the act of taking the marketing information and creating the design of the product to be manufactured.

Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

### Elements of a System

* Architecture: This is the conceptual model that defines the structure, behaviour and more views of a system. We can use flowcharts to represent and illustrate the architecture.
* Modules: These are components that handle one specific task in a system. A combination of the modules make up the system

### Input Design

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well- designed input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding –
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

### Objectives for Input Design:

The objectives of input design are:

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

### Output Design

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

The objectives of input design are:

* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end user’s requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

## UML DIAGRAMS

The Unified Modelling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns, and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things

simpler, has exacerbated these architectural problems. The Unified Modelling Language (UML) was designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

### Contents of UML

In general, a UML diagram consists of the following features:

* **Entities:** These may be classes, objects, users, or systems behaviors.
* Relationship Lines that model the relationships between entities in the system.
* **Generalization --** a solid line with an arrow that points to a higher abstraction of the present item.
* **Association --** a solid line that represents that one entity uses another entity as part of its behavior.
* **Dependency --** a dotted line with an arrowhead that shows one entity depends on the behavior of another entity.

In this project six basic UML diagrams have been explained

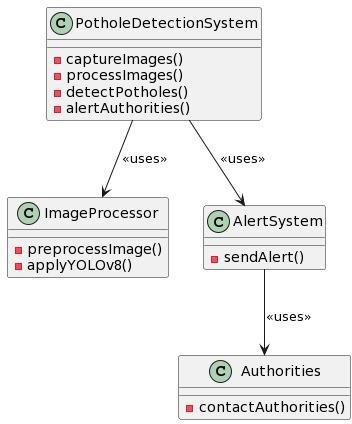
1. Class Diagram
2. Use Case Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. Activity Diagram
6. Deployment Diagram

## CLASS DIAGRAM

UML class diagrams model static class relationships that represent the fundamental architecture of the system. Note that these diagrams describe the relationships between classes, not those between specific objects instantiated from those classes. Thus, the diagram applies to all the objects in the system.

A class diagram consists of the following features:

* + - * **Classes:** These titled boxes represent the classes in the system and contain information about the name of the class, fields, methods and access specifies. Abstract roles of the Class in the system can also be indicated.
      * **Interfaces:** These titled boxes represent interfaces in the system and contain information about the name of the interface and its methods. Relationship Lines that model the relationships between classes and interfaces in the system.
      * **Dependency:** A dotted line with an open arrowhead that shows one entity depends on the behavior of another entity. Typical usages are to represent that one class instantiates another or that it uses the other as an input parameter
      * **Aggregation:** Represented by an association line with a hollow diamond at the tail end. An aggregation models the notion that one object uses another object without "owning" it and thus is not responsible for its creation or destruction.
      * **Inheritance:** A solid line with a solid arrowhead that points from a sub-class to a super class or from a sub-interface to its super-interface.
      * **Implementation:** A dotted line with a solid arrowhead that points from a class to the interface that it implements
      * **Composition:** Represented by an association line with a solid diamond at the tail end. A composition models the notion of one object "owning" another and thus being responsible for the creation and destruction of another object.



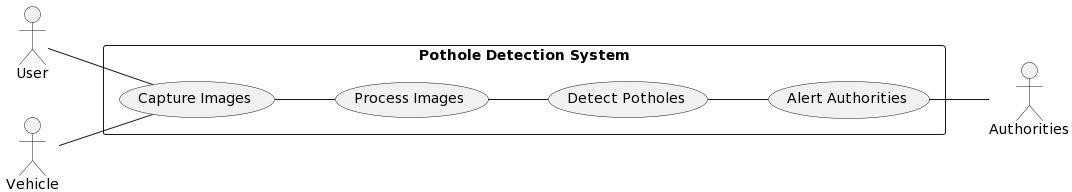
**Fig 6.3.1: Class Diagram**

## USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms.

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



### Fig 6.3.2: Use case Diagram

**Parts of Use cases**

A use case describes a sequence of actions that provide something ofmeasurable value to an actor and is drawn as a horizontal ellipse.

### Actors

An actor is a person, organization, or external system that plays a role in oneor more interactions with the system.

### System boundary boxes (optional)

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not **Relationships.**

### Include

In one form of interaction, a given use case may include another. “Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case.”

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use

case, with the label "«include»". This usage resembles a macro expansion where the included use case behavior is placed inline in the base use case behavior. There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

### Extend

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". The notes or constraints may be associated with this relationship to illustrate the conditions under which this behavior will be executed. Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case. Depending on the modeler’s approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal".

### Generalization

In the third form of relationship among use cases, a generalization/ specialization relationship exists. A given use case may have common behaviours, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation).

### Associations

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modelled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicate the direction of the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads imply control flow and should not be confused with data flow.

## STEPS TO DRAW USE CASE DIAGRAMS

* Identifying Actor
* Identifying Use cases
* Review your use case for completeness

## SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) 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 depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behaviour of a system.

### Elements of sequence diagram

The sequence diagram is an element that is used primarily to showcase the interaction that occurs between multiple objects. This interaction will be shown over certain period of time. Because of this, the first symbol that is used is one that symbolizes the object.

### Lifeline

A lifeline will generally be generated, and it is a dashed line that sits vertically, and the top will be in the form of a rectangle. This rectangle is used to indicate both the instance and the class. If the lifeline must be used to denote an object, it will be underlined.

### Messages

To showcase an interaction, messages will be used. These messages will come in the form of horizontal arrows, and the messages should be written on top of the arrows. If the arrow has a full head, and it’s solid, it will be called a synchronous call. If the solid arrow has a stick head, it will be an asynchronous call. Stick heads with dash arrows are used to represent return messages.

### Objects

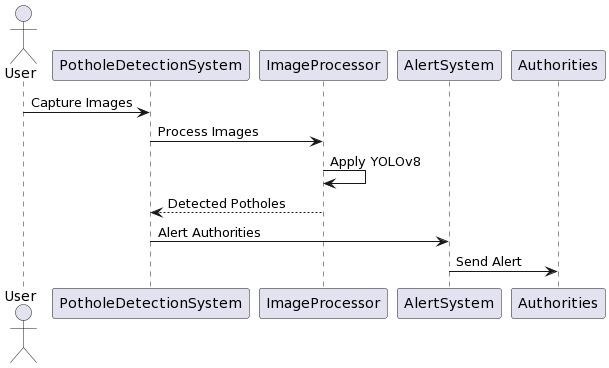
Objects will also be given the ability to call methods upon themselves, and they can add net activation boxes. Because of this, they can communicate with others to show multiple levels of processing. Whenever an object is eradicated or erased

\

from memory, the "X" will be drawn at the lifeline's top, and the dash line will not be drawn beneath it. This will often occur as a result of a message. If a message is sent from the outside of the diagram, it can be used to define a message that comes from a circle that is filled in. Within a UML based model, a Super step is a collection of steps which result from outside stimuli.

### Steps to Create a Sequence Diagram

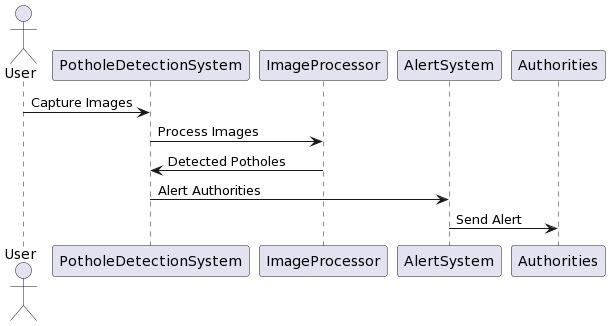
* Set the context for the interaction, whether it is a system, subsystem, operationor class.
* Set the stage for the interaction by identifying which objects play a role in interaction.
* Set the lifetime for each object.
* Start with the message that initiates the interaction.
* Visualize the nesting of messages or the points in time during actual computation.
* Specify time and space constraints, adorn each message with timing mark and attach suitable time or space constraints.
* Specify the flow of control more formally, attach pre and post conditions to each message.



**Fig 6.3.3: Sequence Diagram**

## COLLABORATION DIAGRAM

A Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Interaction diagrams address the dynamic view of a system.



**Fig 6.3.4: Collaboration Diagram**

## ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

### How to draw Activity Diagram?

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram is not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim lane etc. Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated

with constraints and conditions. So before drawing an activity diagram we should identify the following elements.

* Activities
* Association
* Conditions
* Constraints

The following are the basic notational elements that can be used to make up a diagram:

### Initial state

An initial state represents a default vertex that is the source for a single transition to the default state of a composite state. There can be at most one initial vertex in a region. The outgoing transition from the initial vertex may have a behavior, but not a trigger or guard. It is represented by Filled circle, pointing to the initial state.

### Final state

A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed. It is represented by Hollow circle containing a smaller filled circle, indicating the final state.

### Rounded rectangle

It denotes a state. Top of the rectangle contains a name of the state. Can contain a horizontal line in the middle, below which the activities that are done in that state are indicated.

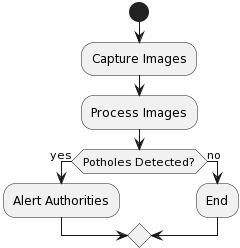
### Arrow

It denotes transition. The name of the event (if any) causing this transition

labels the arrow body.

### Steps To Construct Activity Diagram

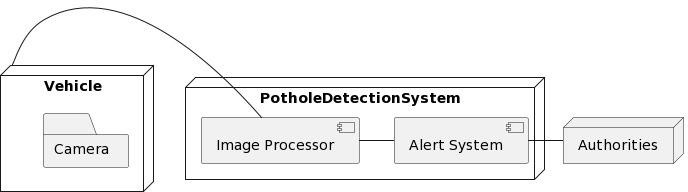
* Identify the preconditions of the workflow
* Collect the abstractions that are involved in the operations
* Beginning at the operation’s initial state, specify the activities and actions.
* Use branching to specify conditional paths and iterations
* Use forking & joining to specify parallel flows of control.



**Fig 6.3.4: Activity Diagram**

## DEPLOYMENT DIAGRAM

The Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.



**Fig 6.3.6: Deployment Diagram**

# CHAPTER 7 SYSTEM IMPLEMENTATION

## IMPLEMENTATION PROCESS

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

### Step 1: Set Up Development Environment

1. Install Python and required libraries: Python, ultralytics, OpenCV, scikit-learn, Pandas.
2. Set up a development environment using your preferred text editor or IDE.

### Step 2: Collect Training Data

1. Collect a dataset of pothole images for training the pothole detection model from Roboflow or Kaggle website.
2. Ensure diversity in the dataset to account for variations in lighting, environment conditions.

### Step 3: Train Face Recognition Model

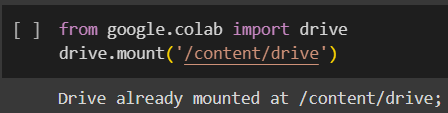
1. Preprocess the training data (e.g., pothole detection, normalization).
2. Train the pothole detection model using deep learning YOLOv8 object detection algorithm.
3. Evaluate the model's performance using validation data and fine-tune as necessary.

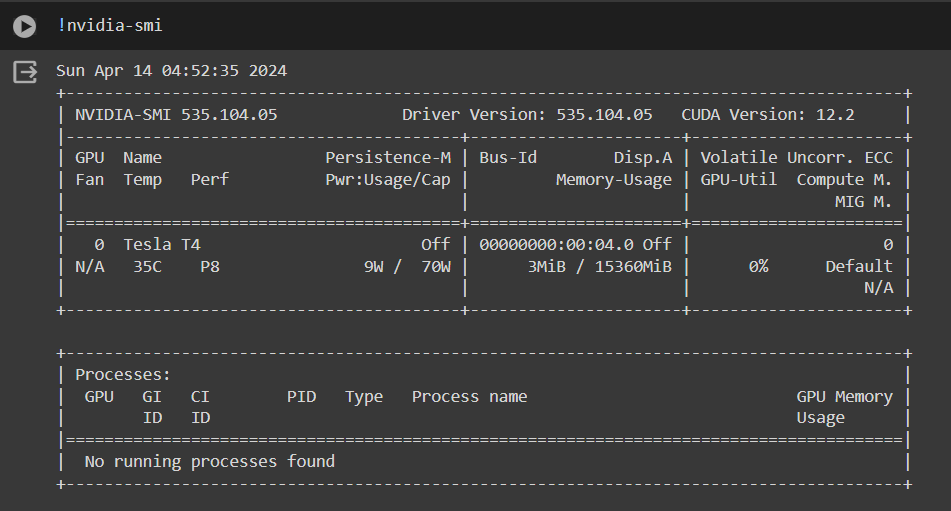
### Step 4: Testing and Validation

1. Write unit tests and integration tests to ensure the correctness of each component.
2. Conduct user acceptance testing (UAT) to validate the system's functionality and usability.

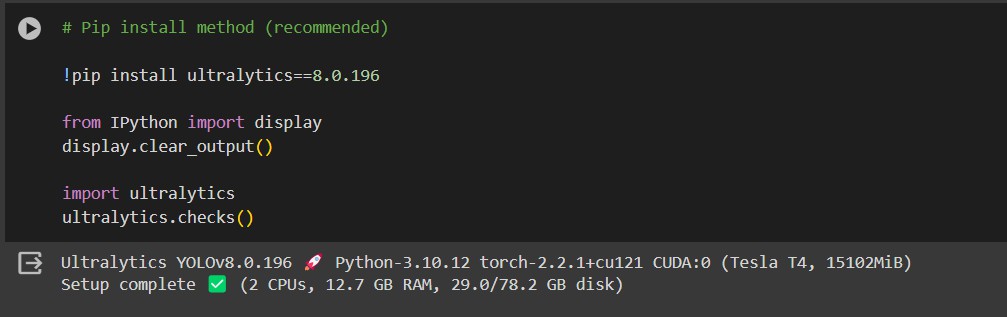
## SOURCE CODE

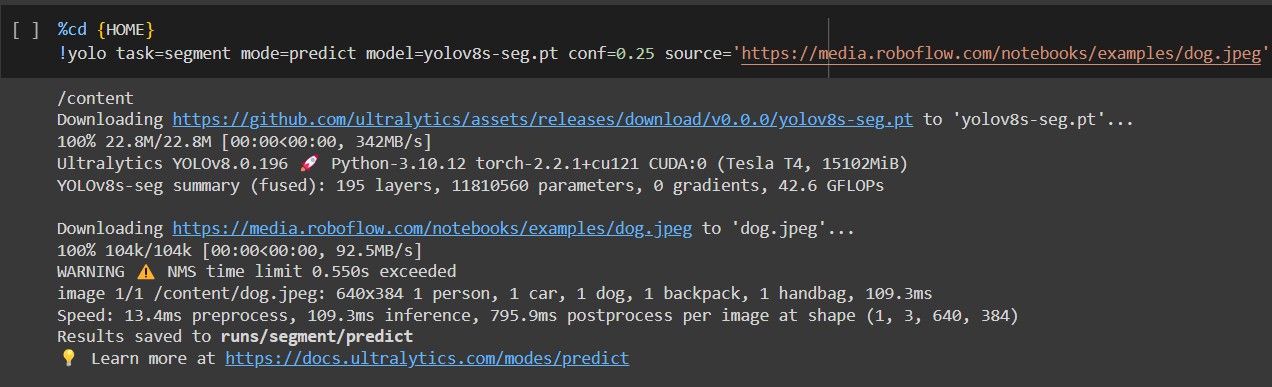
### Model Training Source Code



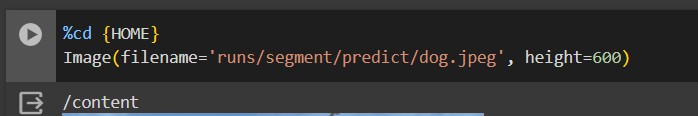


**Installing YOLOv8:**



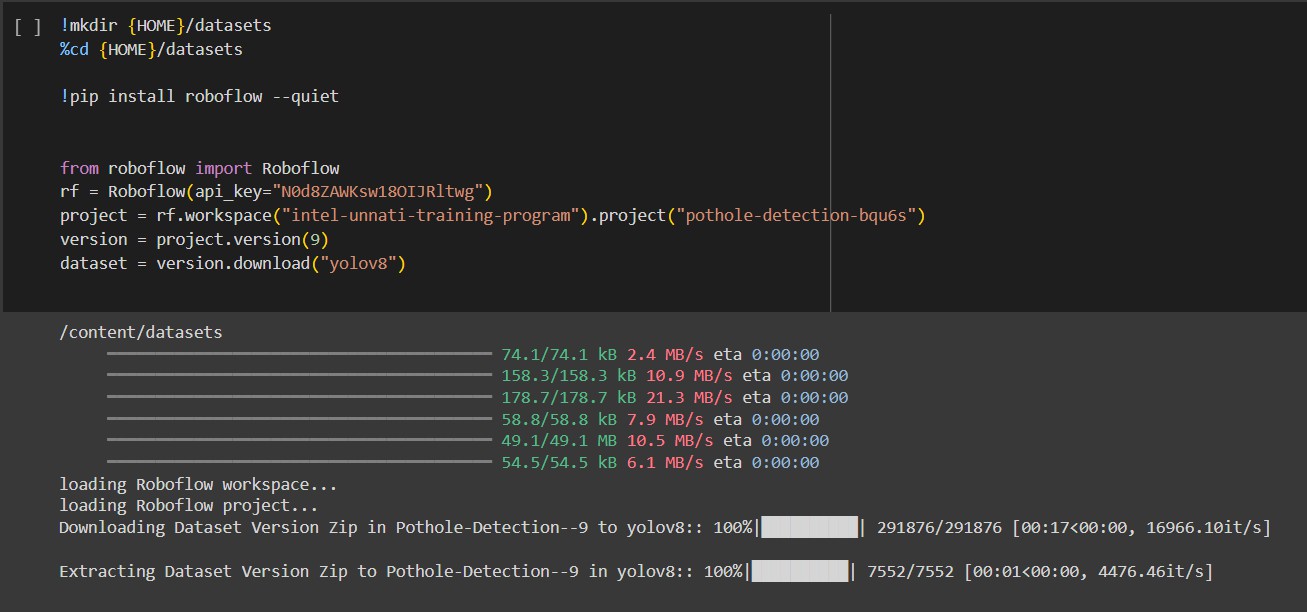


**yolo mode=predict runs** YOLOv8 inference on a variety of sources, downloading models automatically from the latest YOLOv8 release, and saving results to **runs/predict**.

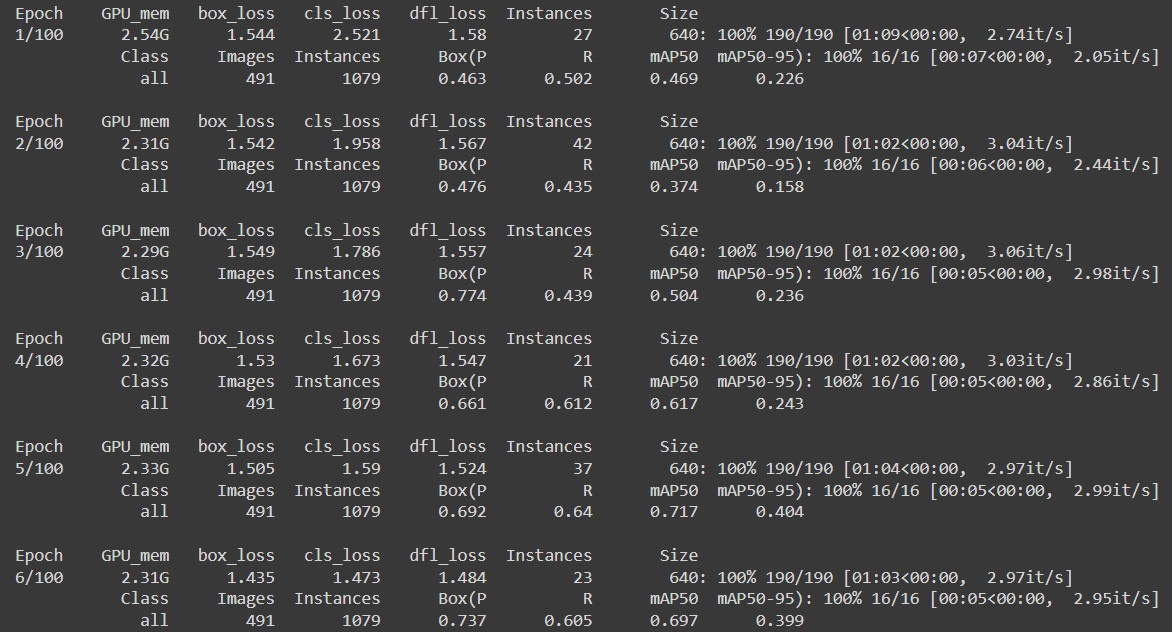


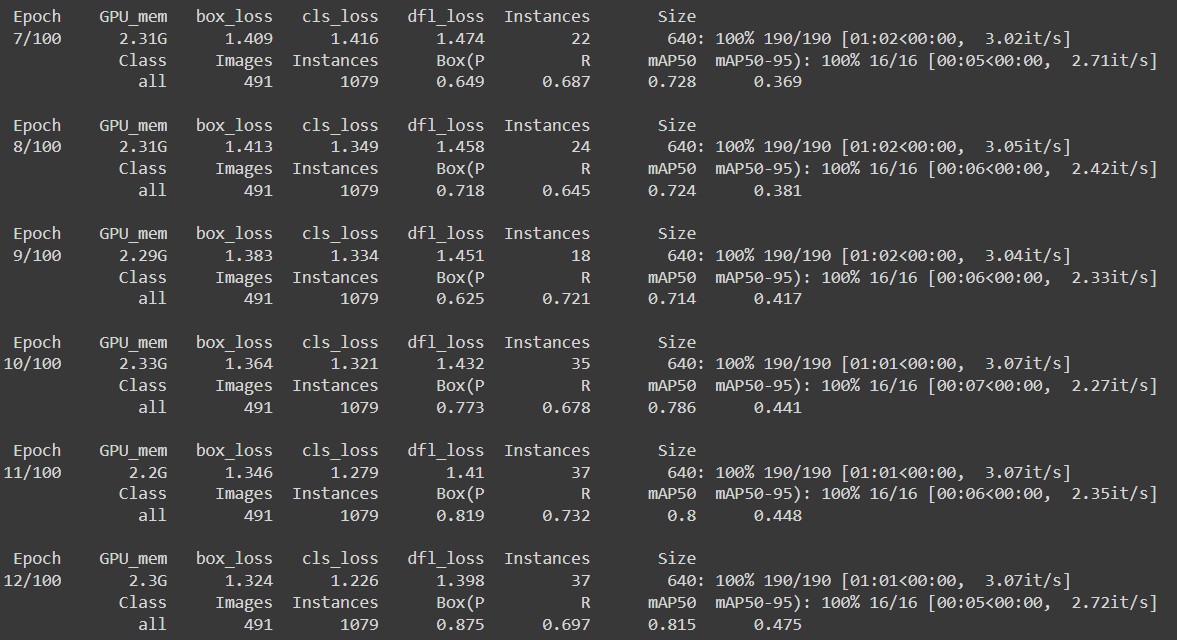


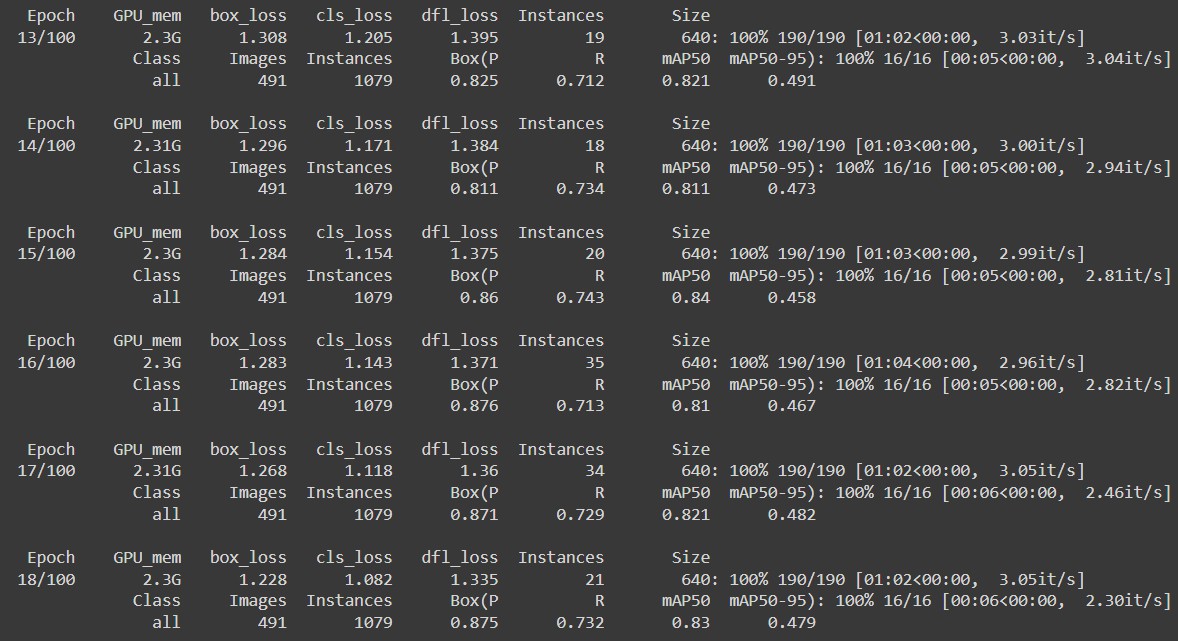
**Collecting Dataset**



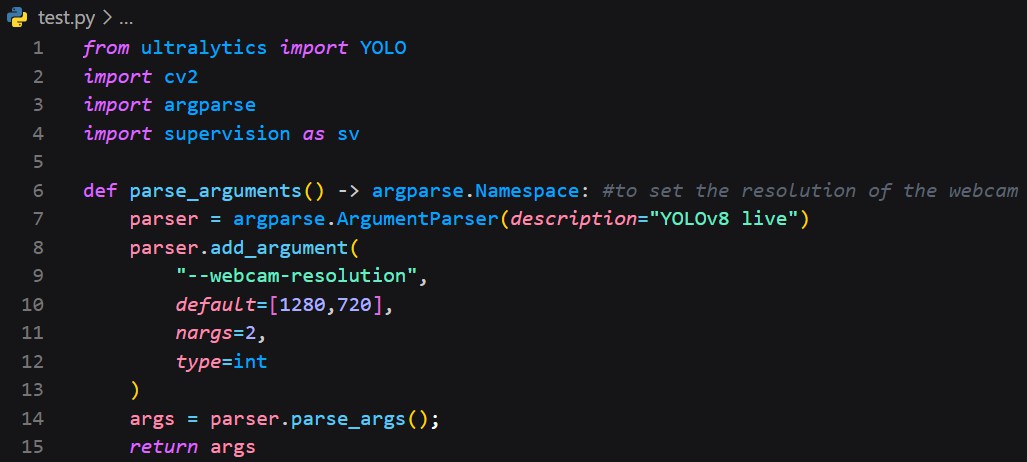
**Training Pothole Detection Model**

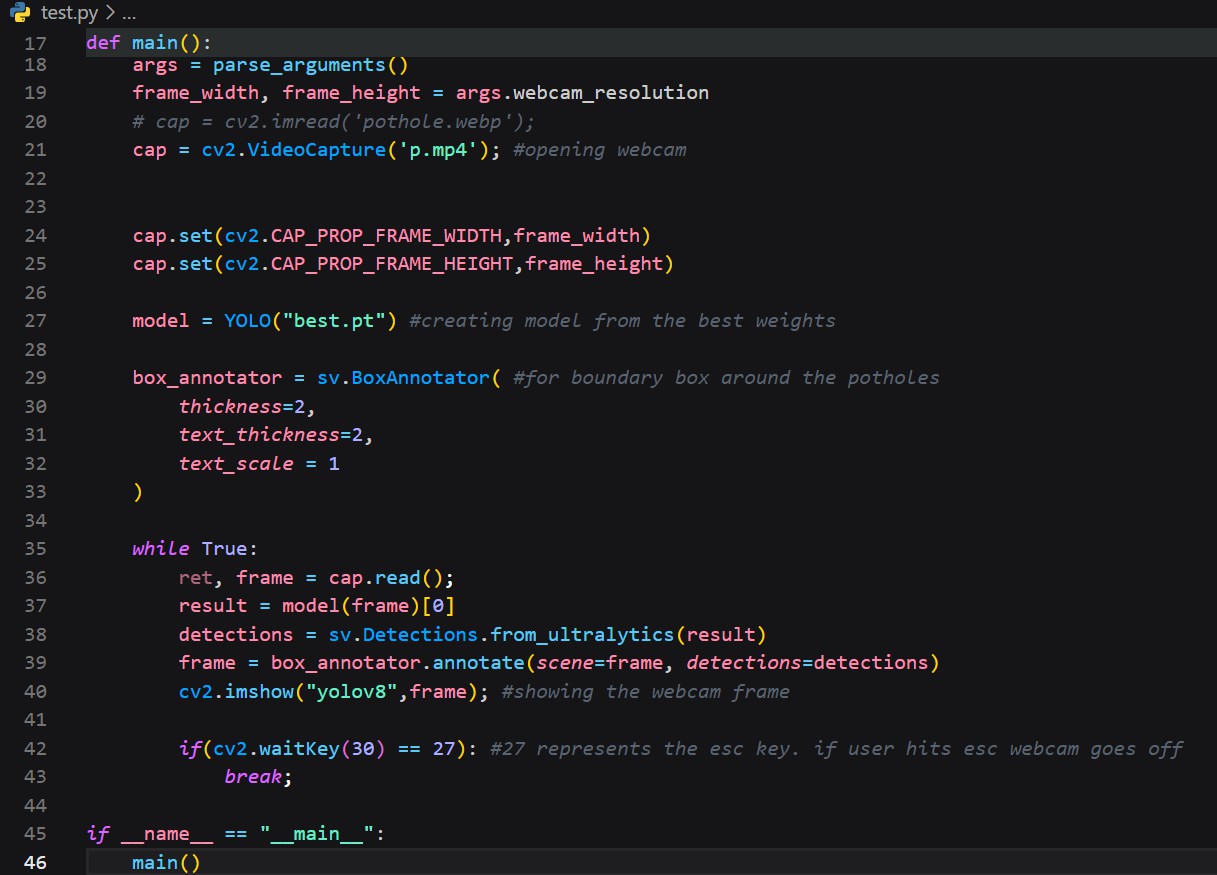






**File: test.py**





# CHAPTER 8 TESTING

A “program unit” stands for a routine or a collection of routines implemented by an individual programmer. It might even be a stand-alone program or a functional unit a larger program.

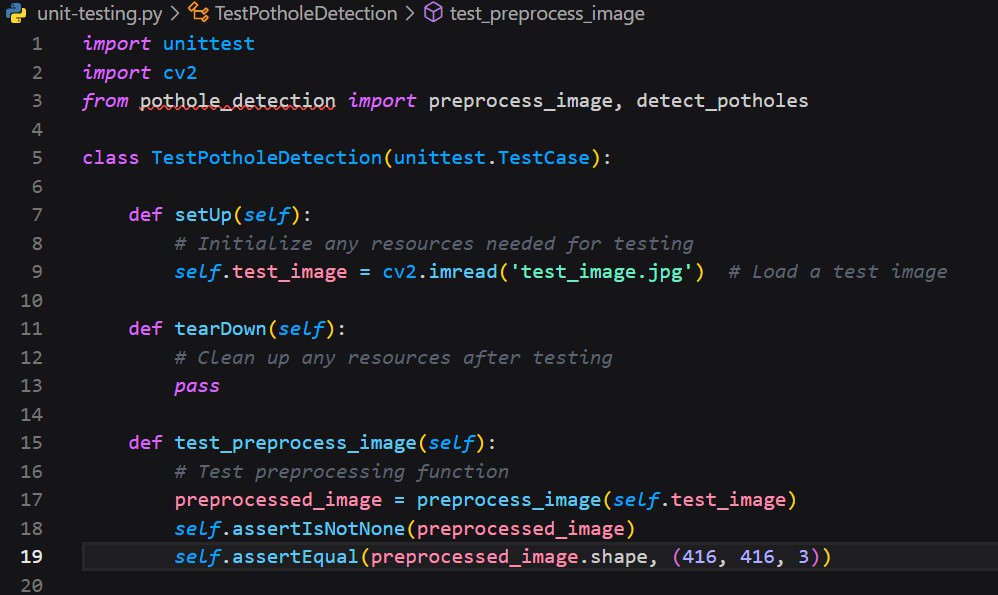
## UNIT TESTING

Unit testing is performed prior to integration of the unit into a larger system. It is like coding and debugging ->unit testing ->integration testing. A program unit must be tested for functional tests, performance tests, stress tests and structure tests.

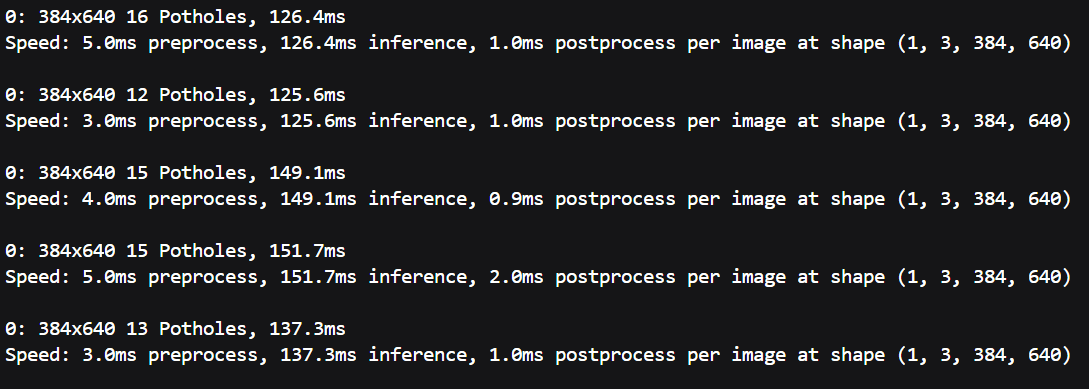
Functional tests refer to executing the code with standard inputs, for which the results will occur within the expected boundaries. Performance test determines the execution time spent in various parts of the unit, response time, device utilization and throughput. Performance testing will help the tuning of the system.

Stress tests drive the system to its limits. They are designed to internationally break the unit. Structure tests verify logical execution along different execution paths. Functional, performance and stress tests are collectively known as “black box testing”. Structure testing is referred to as “white box” or “glass box” testing. Program errors can be classified as missing path errors, computational errors and domain errors.

Even if it looks like all possible execution paths have been tested, there might still exist some more paths. A missing path error occurs, when a branching statement and the associated computations are accidentally omitted. Missing paths can be detected only by functional specifications. A domain error occurs when a program traverses the wrong path because of an incorrect predicate in a branching statement. When a test case fails to detect a computational error there is said to be a coincidental error.







## DEBUGGING

Debugging is eliminating the cause of known errors. Commonly used debugging techniques are induction, deduction, and backtracking. Debugging by induction involves the following steps:

* + - Collect all the information about test details and test results
    - Look for patterns
    - Form one or more hypotheses and rank/classify them.
    - Prove/disprove hypotheses. Re examine
    - Implement appropriate corrections
    - Verify the corrections. Re run the system and test again until satisfactory
    - Debugging by deduction involves the following steps:
    - List possible causes for observed failure.
    - Use the available information to eliminate various hypotheses.
    - Prove/disprove the remaining hypotheses.
    - Determine the appropriate correction.
    - Carry out the corrections and verify.

Debugging by backtracking involves working backward in the source code from point where the error was observed. Run additional test cause and collect more information.

## INTEGRATION TESTING

Integration testing strategies include bottom-up (traditional), top-down and sandwich strategies. Bottom-up integration consists of unit testing, followed by testing entire system. Unit testing tries to discover errors in modules. Modules are tested independently in an artificial environment known as “test harness”. Test harnesses provide data environments and calling sequences for the routines and subsystem that are being tested in isolation.

Disadvantages of bottom-up testing include that harness preparation, which can sometimes take about 50% or more of the coding and debugging effort for a smaller product. After testing all the modules independently and in isolation, they are

linked and executed in one single integration run. This is known as “Big bang” approach to integration testing. Isolating sources of errors is difficult in “big bang” approach.

Top-down integration starts with main routine and one or two immediately next lower-level routines. After a through checking the top level becomes a test harness to its immediate subordinate routines. Top-down integration offers the following advantages.

System integration is distributed throughout the implementation phase. Modules are integrated as they are developed.

* + - Top-level interfaces are first test.
    - Top-level routines provide a natural test harness for lower-level routines.
    - Errors are localized to the new modules and interfaces that are being added.

Though top-down integrations seem to offer better advantages, it may not be applicable in certain situations. Sometimes it may be necessary to test certain low- level modules first. In such situations, a sandwich strategy is preferable. Sandwich integration is mostly top-down, but bottom-up techniques are used on some modules and sub systems. This mixed approach retains the advantages of both strategies.

## SYSTEM TESTING

System testing involves two activities: Integration testing and Acceptance testing. Integration strategy stresses on the order in which modules are written, debugged and unit tested. Acceptance test involves functional tests, performance tests and stress tests to verify requirements fulfillment. System checking checks the interface, decision logic, control flow, recovery procedures and throughput, capacity and timing characteristics of the entire system.

**Test Cases**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input text | Tested for whether potholes  are detected or not | Success |

**Table 8.4.1: Test cases format**

**Test Cases Model building:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. N O** | **Test cases** | **I/O** | **Expecte d O/T** | **Actual O/T** | **P**  **/ F** |
| 1 | Read the dataset. | Dataset path. | Dataset need to read  successfu lly. | Dataset fetched successfully. | P |
| 2 | Performing  pre-  Processing on the dataset | Pre- processi ng part  tak es place | Pre- processin g  Should be performe  d on dataset | Pre- processing successfully completed. | P |
| 3 | Model Building | Model Buildin g for the clean data | Need to create model using required  algorithm | Model Created Successfully. | P |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Potholes detection | Input Features provide d. | Output should be whether potholes detected  or not | Model detected potholes successfully | P |

**Table 8.4.2: Test cases model building**

## ACCEPTANCE TESTING

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests in order to check whether the system implemented satisfies the requirements specifications. Quality assurance people as well as customers may simultaneously develop acceptance tests and run them. In additional to functional and performance tests, stress test are performed to determine the limits/limitations of the system developed. For example, a compiler may be tested for its symbol table overflows or a real-time system may be tested for multiple interrupts of different/same priorities.

Acceptance test tools include a test coverage analyzer, and a coding standards checker. Test coverage analyzer records the control paths followed for each test case. A timing analyzer reports the time spent in various regions of the source code under different test cases. Coding standard are stated in the product requirements. Manual inspection is usually not an adequate mechanism from detecting violations of coding standards.

## TESTING OBJECTIVES

Testing is a process of execution a program with the intent of finding on errors. A good test is on that has a high probability of finding an undiscovered errors. Testing is vital to the success of the system. System testing is the state of implementation, which ensures that the system works accurately before live operations commence. System testing makes a logical assumption that the system is correct and that the system is correct and that the goals are successfully achieved.

## EFFECTIVE TESTING PREREQUISITES

### Integration testing

An overall test plan for the project is prepared before the start of coding.

### Validation testing

This project will be tested under this testing sample data and produce the correct sample output.

### Recovery testing

This project will be tested under this testing using correct data input and its product and the correct valid output without any errors.

### Security testing

This project contains password to secure the data.

### Test Data and Input

Taking various types of data we do the above testing. Preparation of test data plays a vital role in system testing. After preparing the test data the system under study is treated using the test data. While testing the system by using the above testing and correction methods. The system has been verified and validated by running with both.

* Run with live data
* Run with test data

### Run with test data

In the case the system was run with some sample data. Specification testing was also done for each conditions or combinations for conditions.

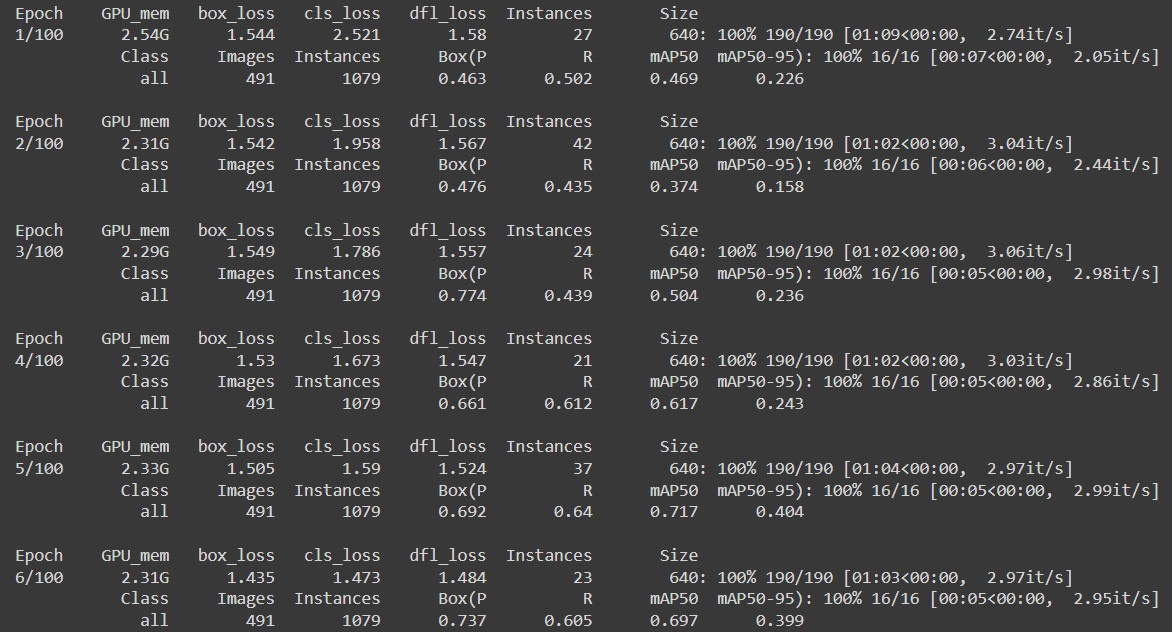
### Run with live data

The system was tested with the data of the old system for a particular period.

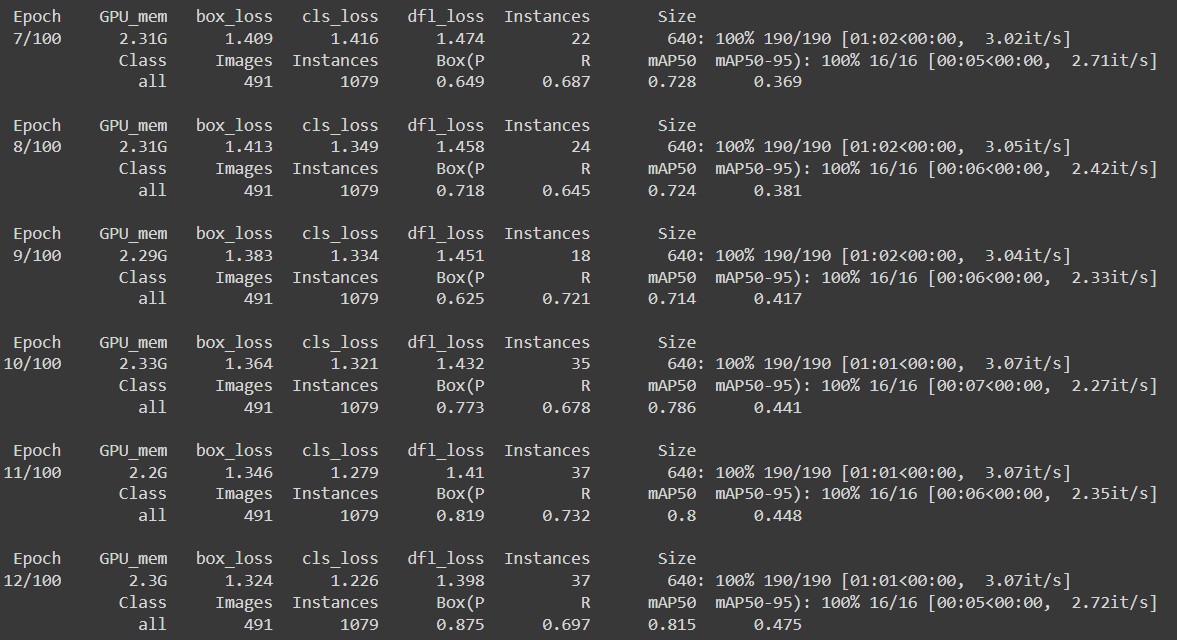
Then the new reports were verified with the old one.

# CHAPTER 9 SCREEN LAYOUTS

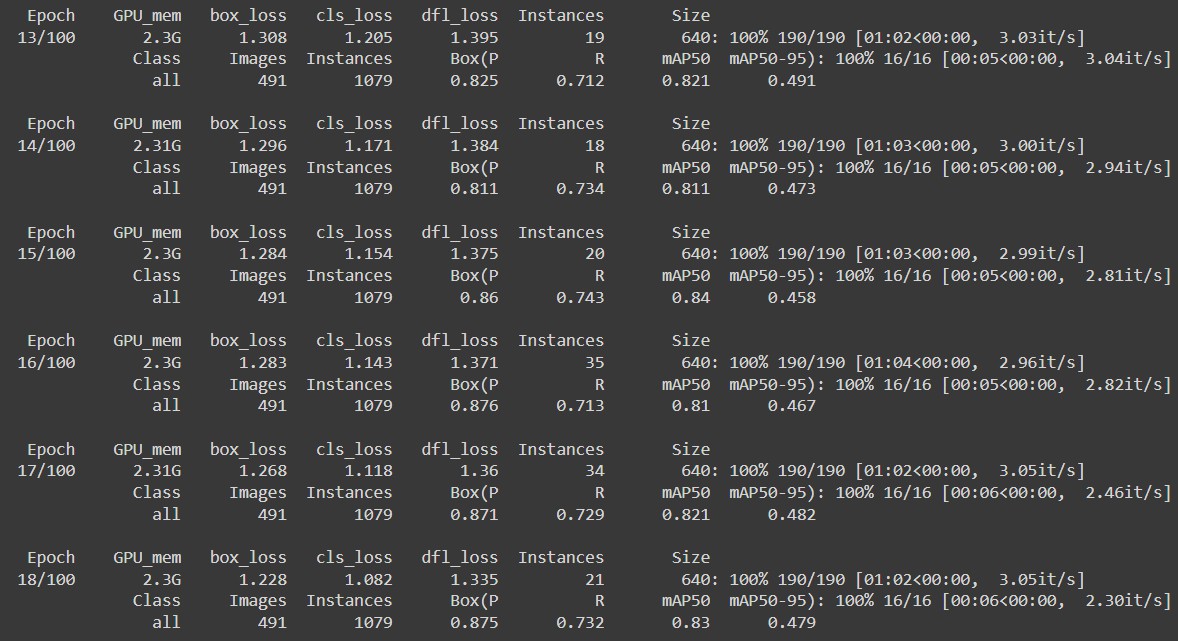
**Model Building:**



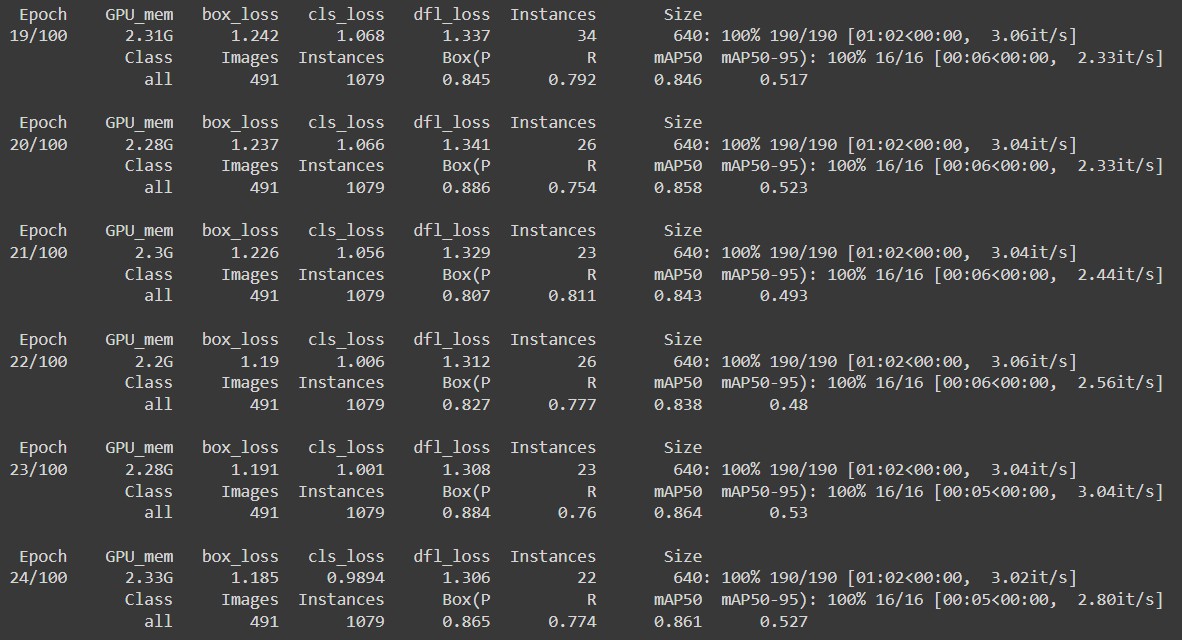
**Fig: Model building-1**



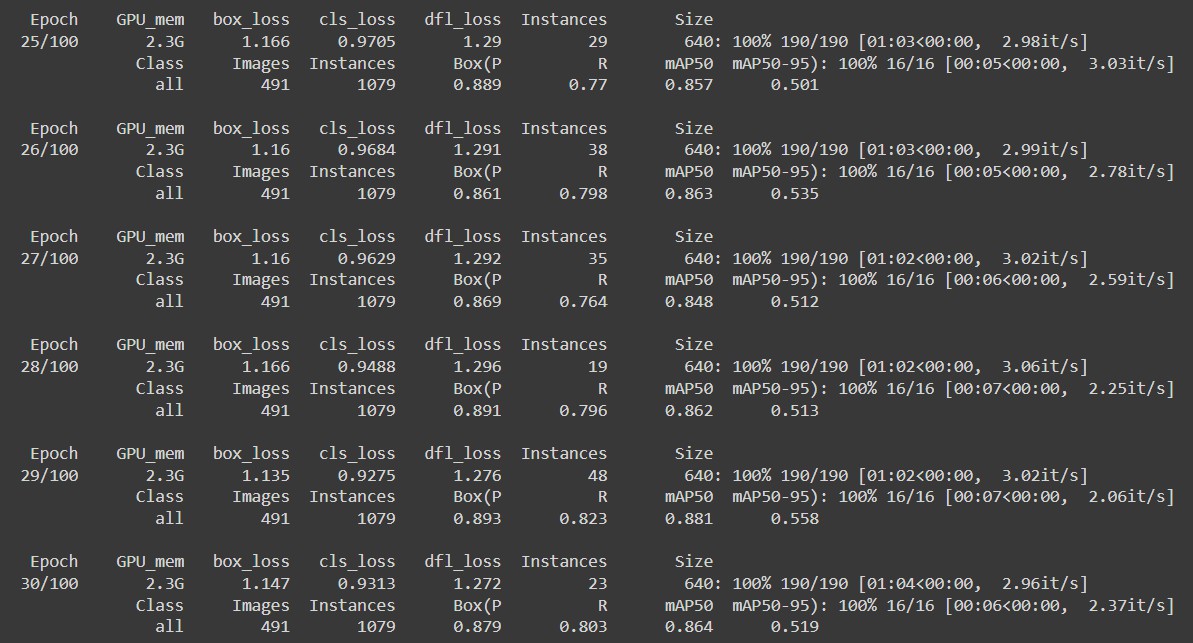
**Fig: Model building-2**



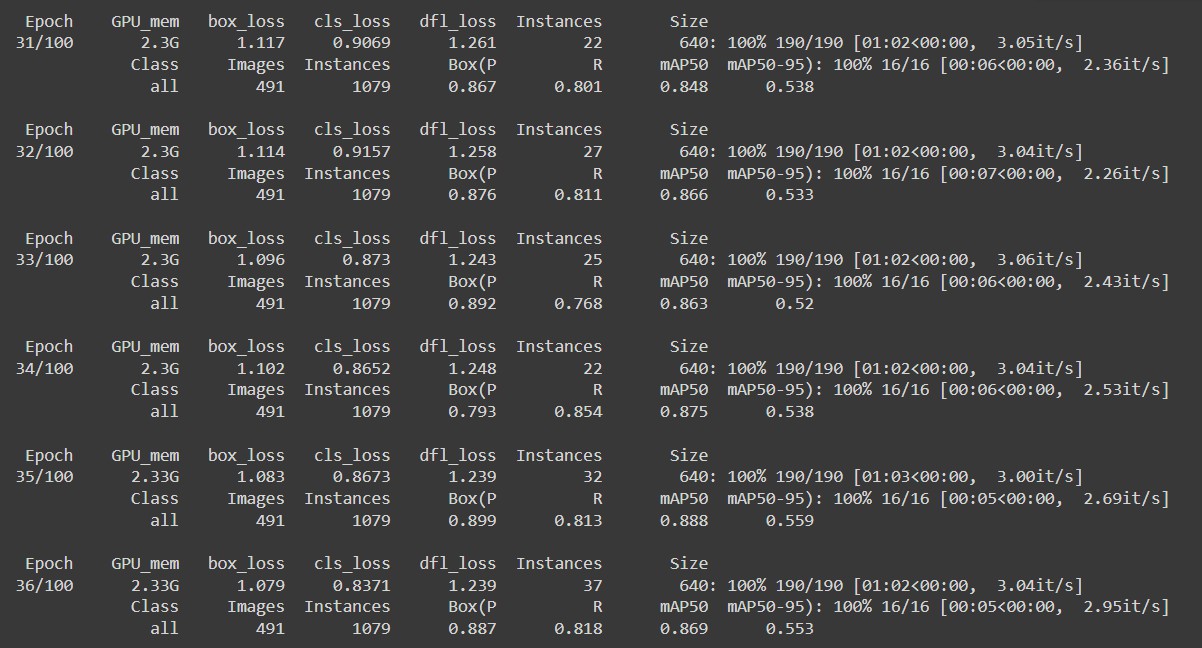
**Fig: Model building-3**



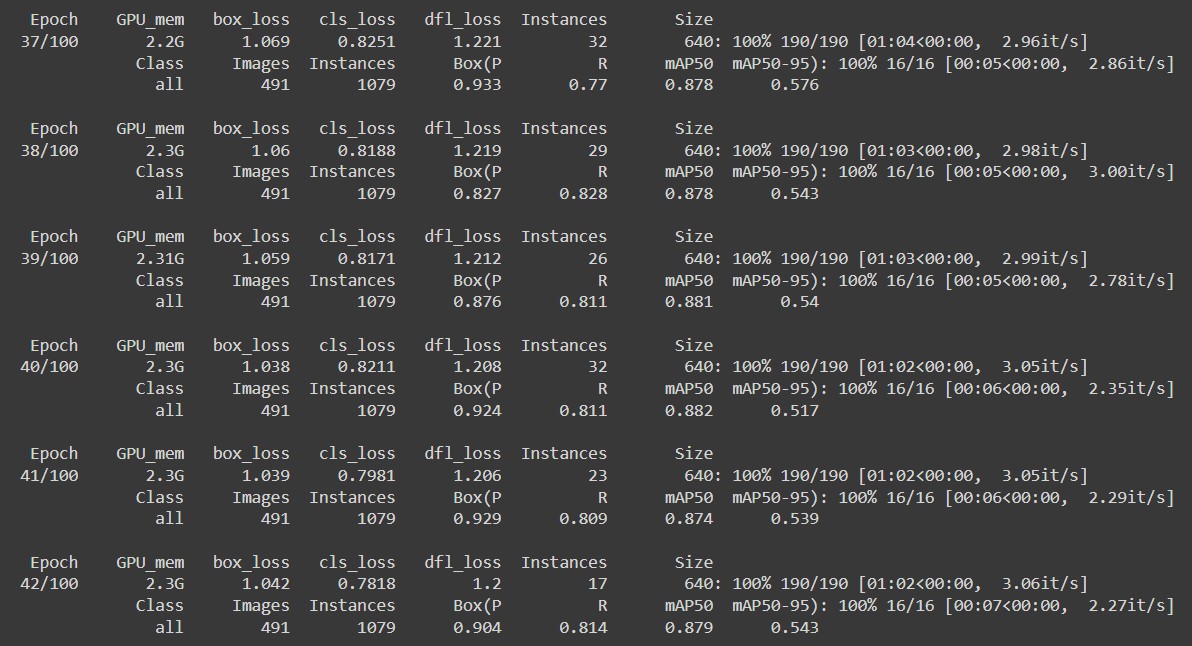
**Fig: Model building-4**



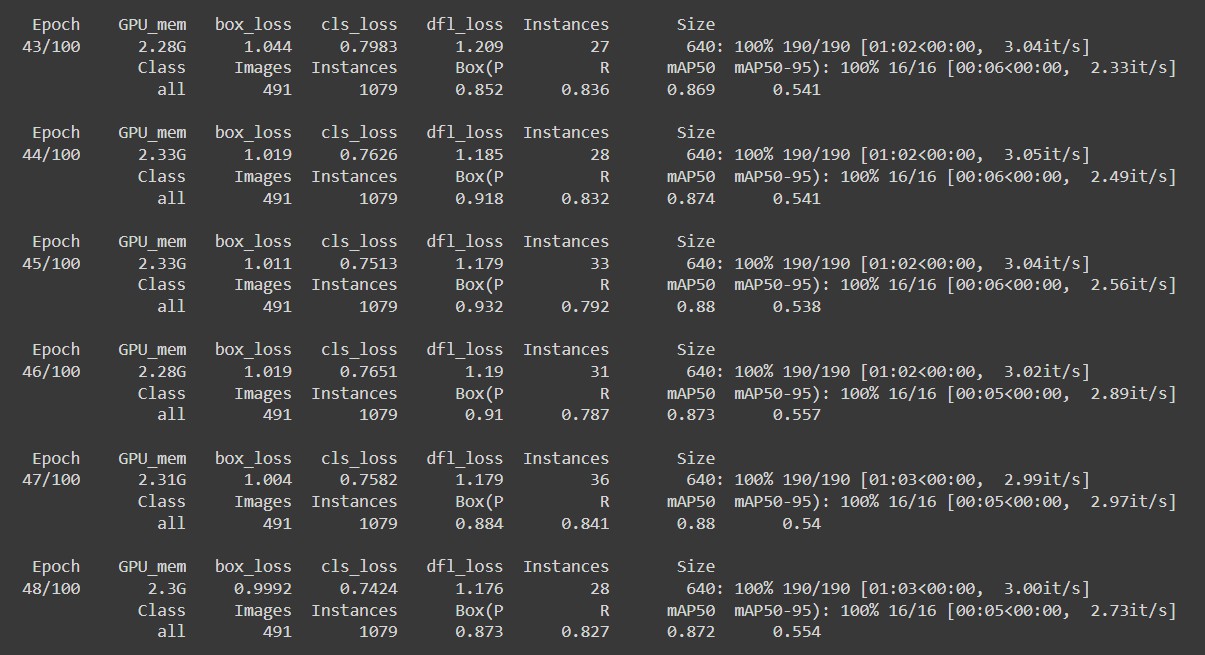
**Fig: Model building-5**



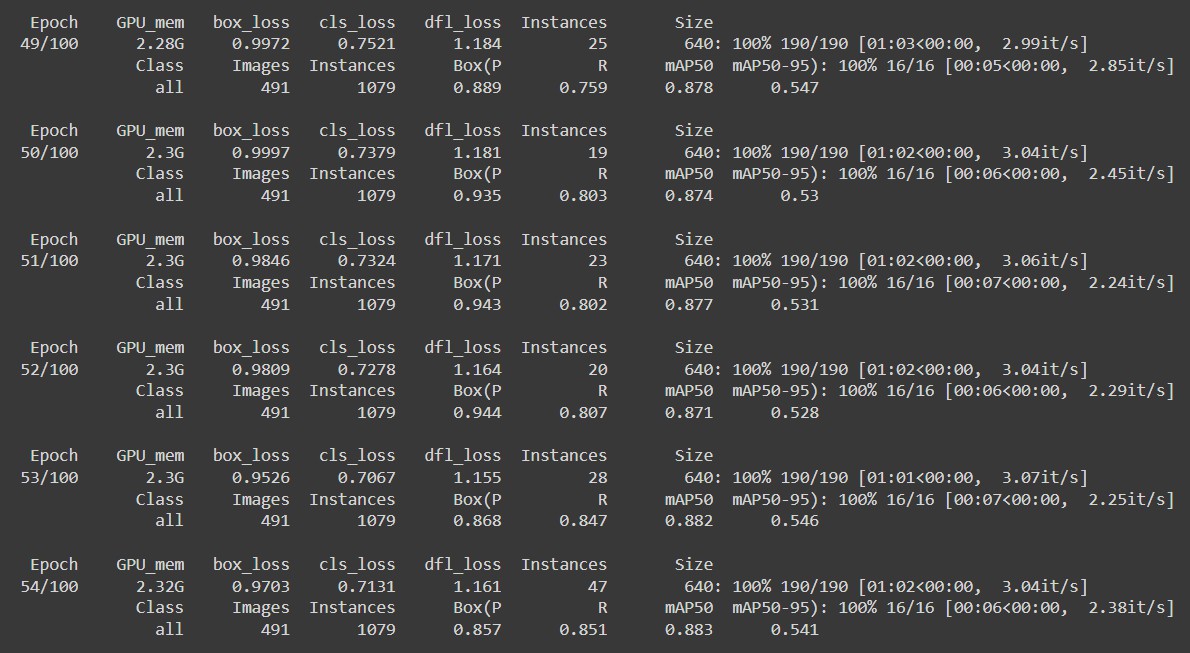
**Fig: Model building-6**



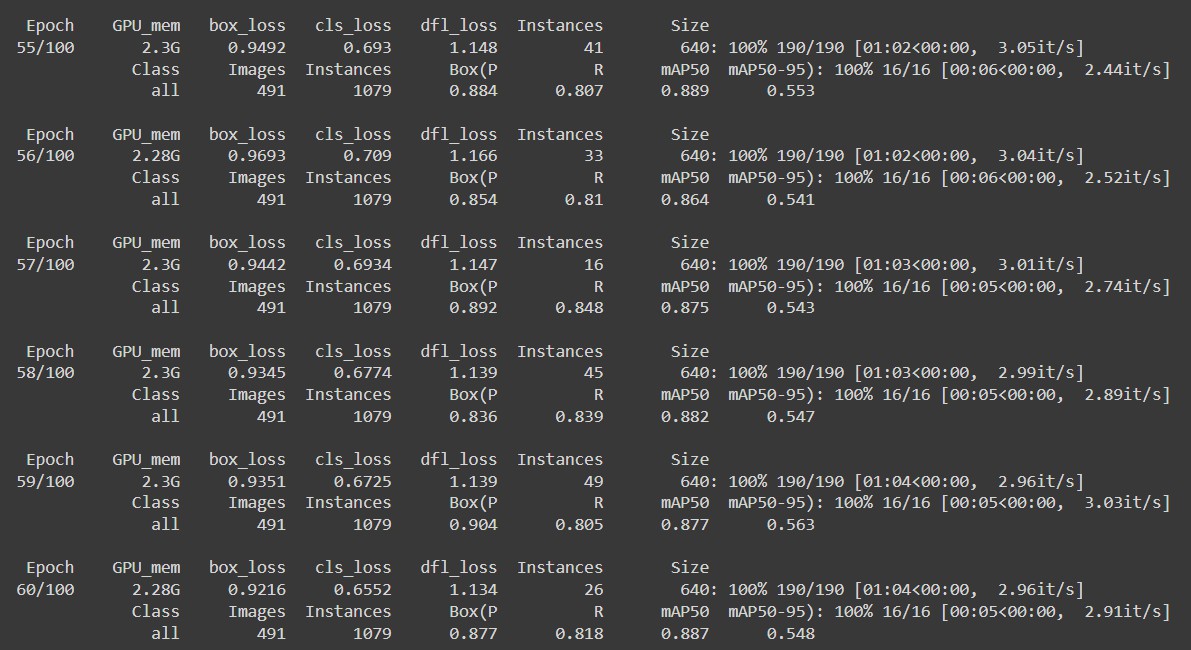
**Fig: Model building-7**



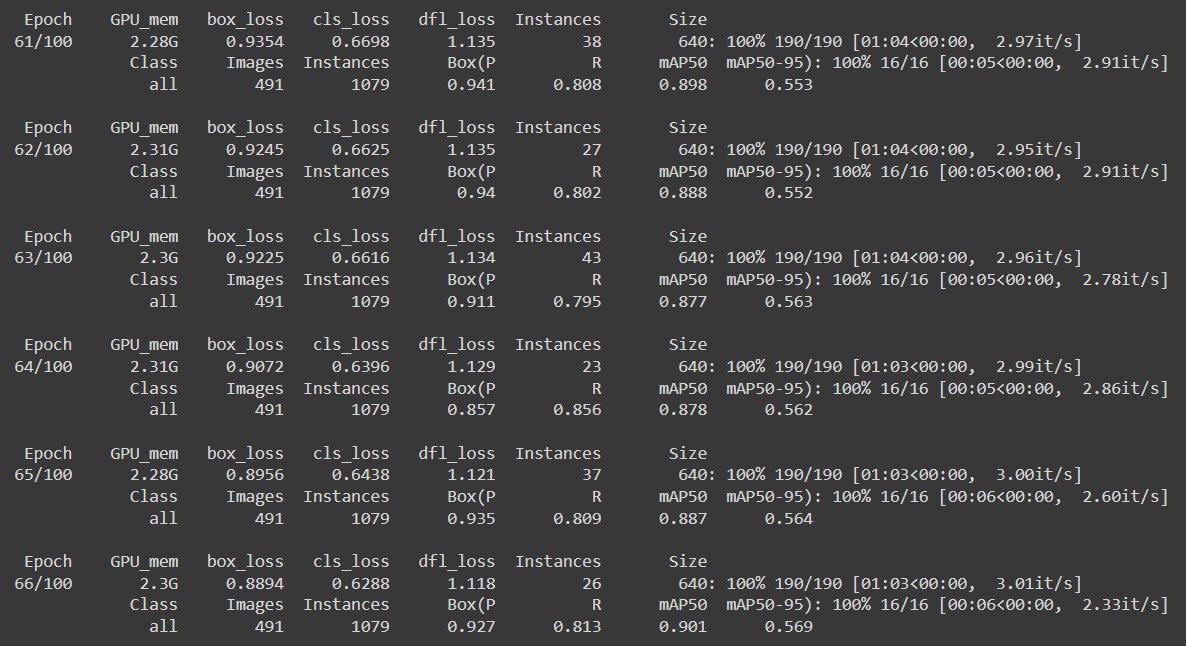
**Fig: Model building-8**



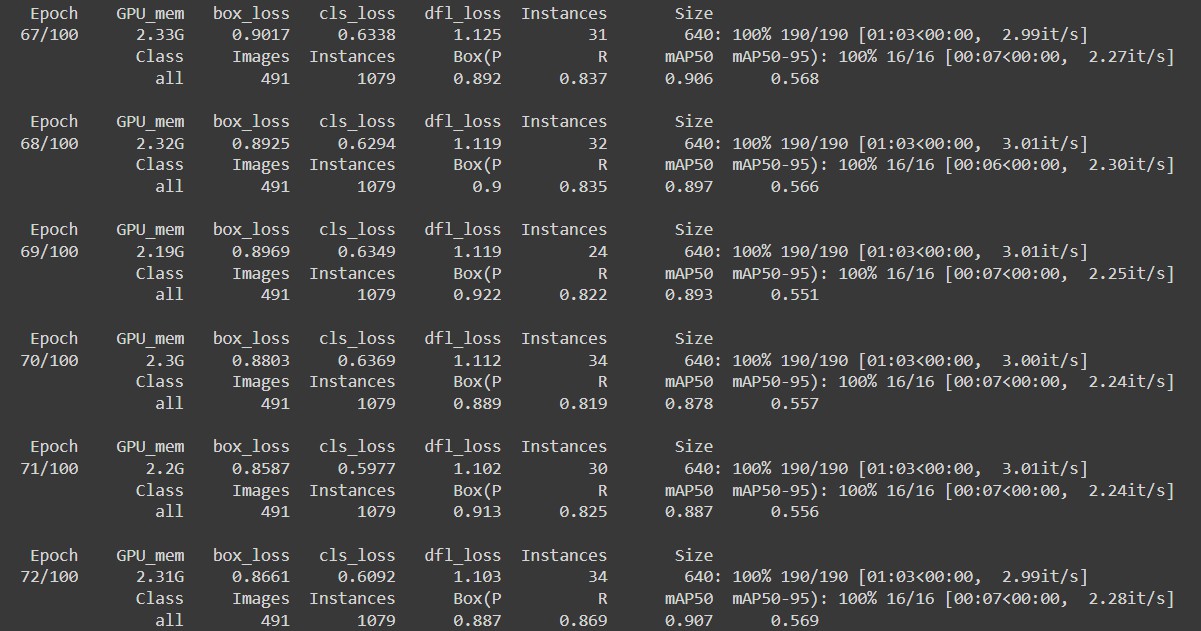
**Fig: Model building-9**



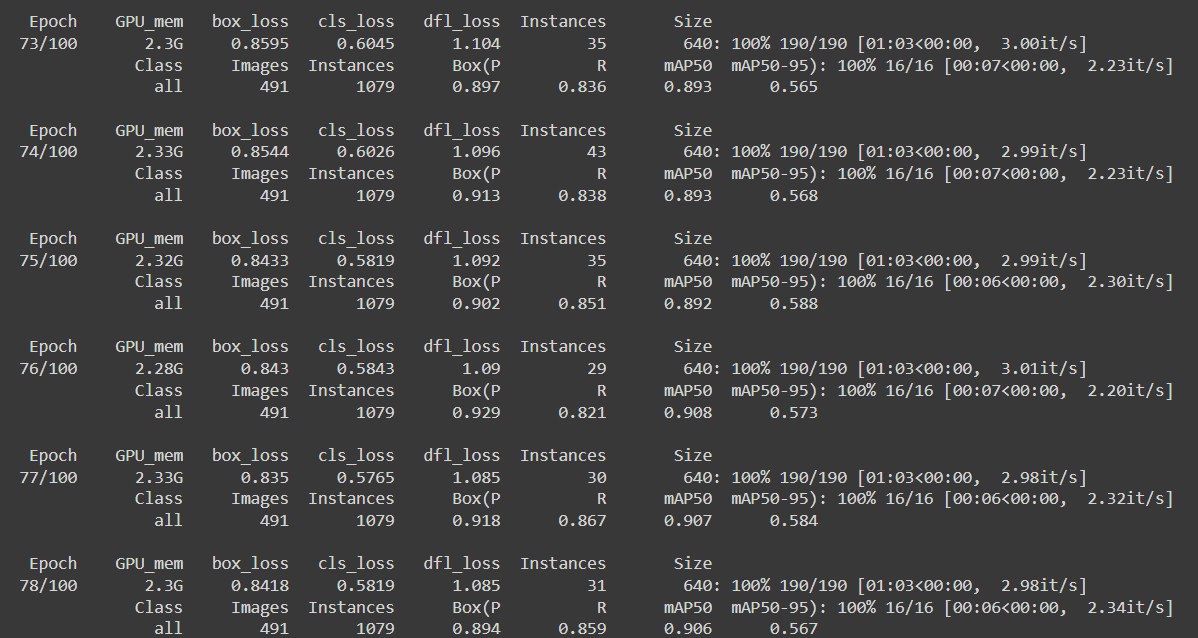
**Fig: Model building-10**



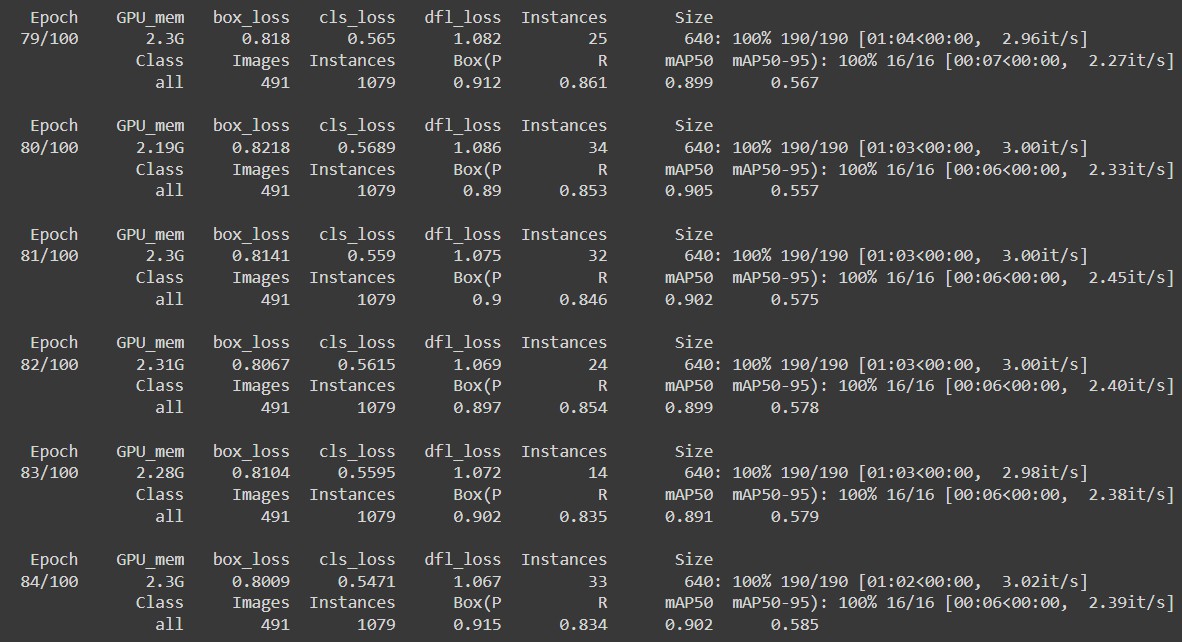
**Fig: Model building-11**



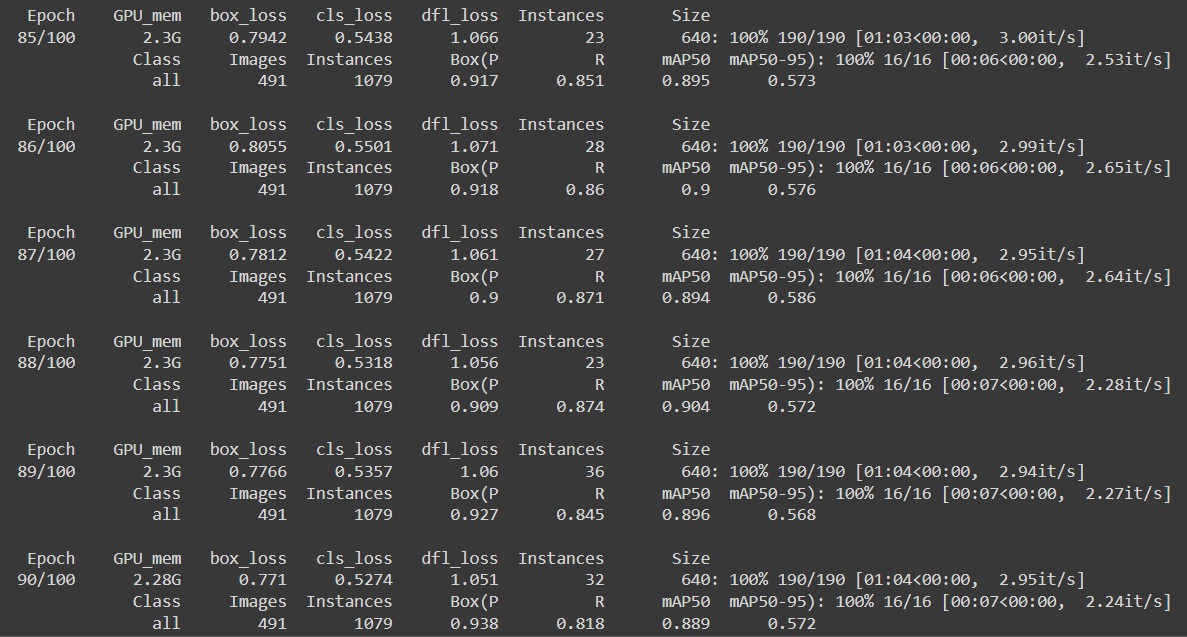
**Fig: Model building-12**



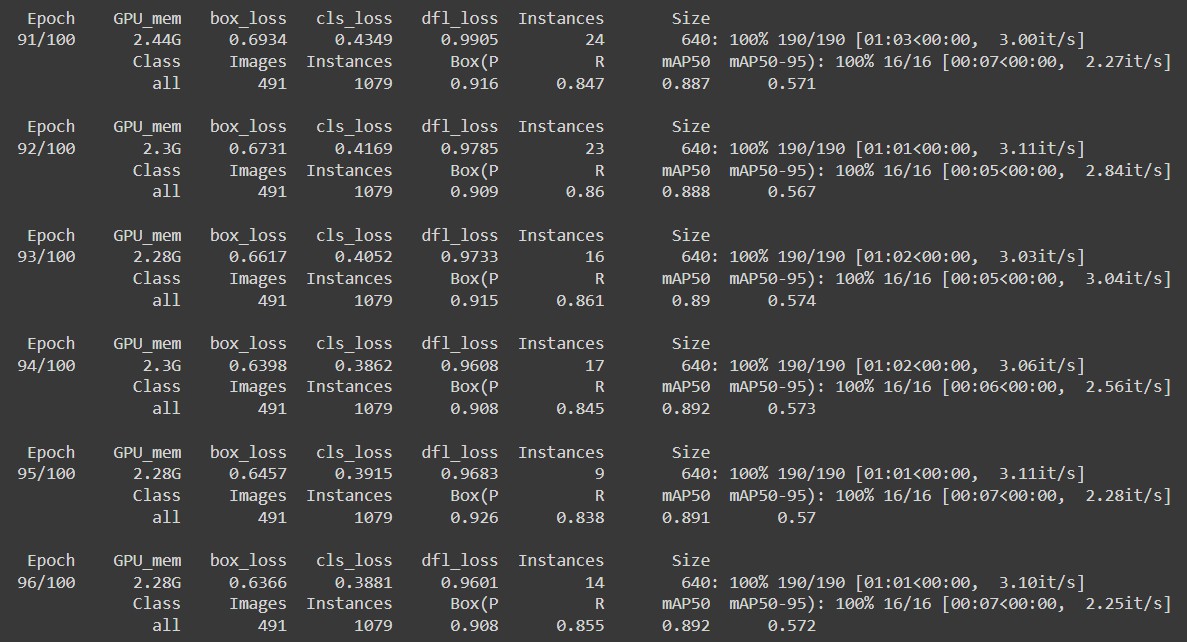
**Fig: Model building-13**



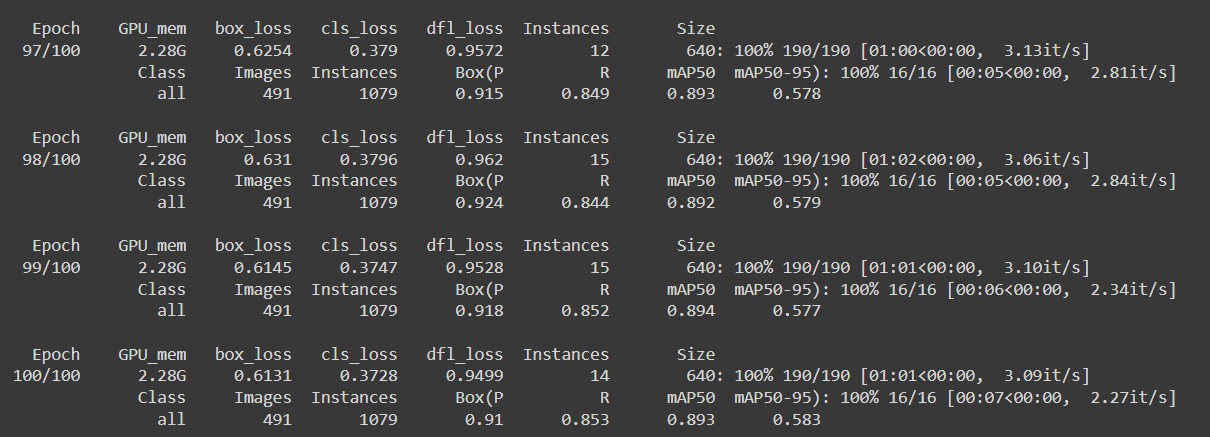
**Fig: Model building-14**



**Fig: Model building-15**



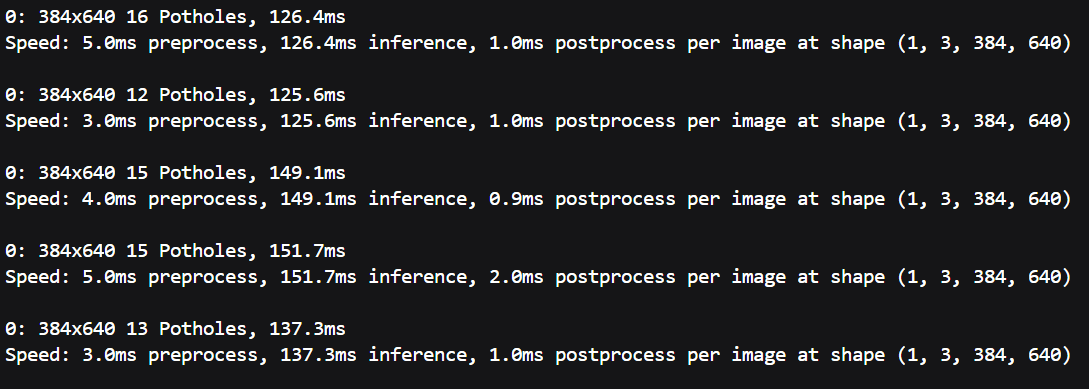
**Fig: Model building-16**



**Fig: Model building-17**



**Fig: Output Screen**



**Fig: Testing Output**

# CHAPTER 10

**CONCLUSION AND FUTURE ENHANCEMENTS**

In conclusion, the presented project delves into the realm of Object Detection, a crucial aspect in enhancing machines' ability to detect the potholes within visual content. The fusion of Convolutional Neural Networks (CNN) and YOLOv8 models stands out as a key enabler in this transformative process.

The incorporation of YOLOv8 adds a linguistic dimension to the project, enabling the model to detect potholes in the images as well as in the videos.

# CHAPTER 11 FUTURE ENHANCEMENTS

**Dataset Augmentation**: Increasing the diversity and size of the training dataset through techniques like data augmentation can help improve the model's ability to generalize to different environmental conditions, road surfaces, lighting conditions, and types of potholes.

**Integration with GIS Data**: Integrating geographical information system (GIS) data, such as road maps and pavement condition surveys, can provide additional contextual information to the pothole detection system, helping to improve accuracy and enable better decision-making.

**Real-time Feedback Mechanisms**: Implementing real-time feedback mechanisms that provide immediate alerts or notifications to relevant stakeholders (such as road maintenance authorities or navigation apps) when potholes are detected can help expedite repair efforts and improve road safety.

**User Interface Improvements**: Enhancing the user interface of the pothole detection system to make it more intuitive and user-friendly for both end-users (such as drivers or road maintenance crews) and administrators (such as system operators or data analysts).

# CHAPTER 12 REFERENCES

1. W. health, Statistics, “Injury Deaths Rise in Rank,” 2008, [Online]. Available:

https:/[/www.who.int/viol](http://www.who.int/violence_injury_prevention/key_facts/VIP_key_fact_3.pdf/)e[nce\_injury\_prevention/key\_facts/VIP\_key\_fact\_3.pdf/.](http://www.who.int/violence_injury_prevention/key_facts/VIP_key_fact_3.pdf/)

1. A. Glowacz, “Ventilation diagnosis of angle grinder using thermal imaging,” Sensors, vol. 21, no. 8, p. 2853, 2021.
2. B. X. Yu and X. Yu, “Vibration-based system for pavement condition evaluation,” in Proceedings of the Applications of Advanced Technology in Transportation,

pp. 183–189, Chicago, IL, USA, August 2006.

1. K.T. Chang, J.R. Chang, and J.K. Liu, “Detection of pavement distresses using 3d laser scanning technology,” in Proceedings of the Computing in Civil Engineering, pp. 1–11,Cancun, Mexico, May 2005.
2. L. Huidrom, L. K. Das, and S.K. Sud, “Method for automated assessment of potholes, cracks and patches from road surface video clips,” Procedia-Social and Behavioural Sciences, vol.104, pp. 312–321, December 2013.
3. Kamaljit Kaur Sandhu, "Over 9,300 deaths, 25,000 injured in 3 years due to potholes," India Today, Jul. 23, 2018. [Online]. Available: https:/[/www.indi](http://www.indiatoday.in/india/story/over-9300-deaths-25000-injured-in-3-years-)a[today.in/india/story/over-9300-deaths-25000-injured-in-3-years-](http://www.indiatoday.in/india/story/over-9300-deaths-25000-injured-in-3-years-) due-to-potholes-1294147-2018-07-23. [Accessed Jan 9,2023]
4. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified, real-time object detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788, Las Vegas, NV, USA, June 2016.
5. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal Speed and Accuracy of Object Detection,” April 2020, doi: 10.48550/arXiv. 2004.10934.
6. Z. Jiang, L. Zhao, S. Li, and Y. Jia, “Real-time Object Detection Method Based on Improved Yolov4-Tiny,” 2020, doi: 10.48550/arXiv.2011.04244v2.
7. Makesense AI labelling tool, [Online]. Available: https:/[/www.makesense.ai/](http://www.makesense.ai/). [Accessed Jan 10, 2023].

*A Modern Pothole Detection technique using Deep Learning*

Abhishek Kumar

*Department of Ele. & Electronics Engineering*

*Gaya College of Engineering Gaya, Bihar, India* [Abhisheksingh73017@gmail.com](mailto:Abhisheksingh73017@gmail.com)

Vibhav Prakash Singh *Department of Computer Science and Engineering*

*National Institute of Technology, Allahabad*

*UP, India*

[vibhav@mnnit.ac.in](mailto:vibhav@mnnit.ac.in)

Chakrapani

*Department of Computer Science and Engineering*

*Gaya College of Engineering Gaya-Bihar, India* [chakrapani3513.a@gmail.com](mailto:chakrapani3513.a@gmail.com)

Dhruba Jyoti Kalita

*Department of Computer Science and Engineering*

*Gaya College of Engineering Gaya-Bihar, India* [dhruba.nitjsr18@gmail.com](mailto:dhruba.nitjsr18@gmail.com)

***Abstract*- Road accident detection and avoidance are a more difficult and challenging problem in India as poor quality of construction materials get used in road drainage system construction. Due to the above problems, roads get damaged early and potholes appear on the roads which cause accidents. According to a report submitted by the Ministry of Road Transport and Highways transport research wing New Delhi in 2017, approximately 4,64,910 accidents happen per year in India. This paper proposed a deep learning-based model that can detect potholes early using images and videos which can reduce the chances of an accident. This model is basically based on Transfer Learning, Faster Region-based Convolutional Neural Network(F-RCNN) and Inception-V2. There are many models for pothole detection that uses the accelerometer (without using images and videos) with machine learning techniques, but a less number of pothole detection models can be found which uses only machine learning techniques to detect potholes. The results of this work have shown that our proposed model outperforms other existing techniques of potholes detection.**

***Keywords – Transfer Learning, Deep Learning, Tensor flow API, Accelerometer, Image Labelling, F-RCNN, inception-v2***

1. INTRODUCTION

India is the second-largest road network in the world [1]. Therefore, the road network plays an important role in Indian economic development and social functioning. According to the report [2] in the last ten years, the Road Transport sector GDP grew at an annual average rate close to 10% compared to the overall annual GDP growth of 6%. Nowadays road construction is done very rapidly by the government of India. But road maintenance is a challenging task because of the poor drainage system and overloaded vehicles. Due to poor

maintenance of roads, potholes get appeared on the road which causes road accidents. According to statistics submitted by the [3] government of India from 2013 to 2016 potholes claimed 11,836 lives and 36,421 people got injured. Pothole problems cannot be resolved easily because every year almost all the places suffer from floods, disasters, heavy rainfall, etc. Though we cannot maintain the road, we can reduce the number of accidents which are getting increased every year. This paperwork has proposed a novel approach to detect potholes approximately 100 meters ahead using deep learning. Many authors are working on this problem. In [4] Artis M. et.al. have proposed an accelerometer-based model that uses Z-THRESH, Z-DIFF, STDEV(Z) and G-ZERO algorithm and that can be deployed on any Android OS-based devices with limited hardware and software resources. They analyse the performance of the model with a 90% true positive value. Lin, J. et. al. in [5] have used a non-linear SVM model classification tool with a Gaussian radial basis function for the detection of a pothole. In this model, each image of 64×64 gets converted into grayscale and used for the experiment. Here for each grayscale eigenvalue is calculated and an average range of eigenvalue lie between 60 to 100 is searched before applying the SVM model. After this normalization of each image in the range [1,10] is done using eigenvector. Pereira, V. et. al. in [6] have used Convolution Neural Network and compared the performance of their model with SVM and found that their model out formed SVM with 99.80% accuracy. They deployed the model using CNN, pooling, ReLU activation function, Adam Optimizer, and Sigmoid function. Here, convolution and pooling have been used for feature extraction. Adam optimizer is used to reduce the

cost function and sigmoid function for output predicted values between 0 and 1.

1. MODERN BASED POTHOLE DETECTION METHOD

The proposed modern based pothole detection method using the transfer learning technique detects potholes in videos/images in real-time. The method uses common techniques like “use of F- RCNN”, “inception v2 model” which are described below. The main advantage of this method is small training time with an easy training process and higher accuracy.

1. *Tensorflow Object-detection API*

Tensorflow’s object detection API is a very powerful tool that can quickly enable anyone to build and deploy a powerful image recognition system. It provides many pre-trained models (trained on different datasets) which can be used to build customized classifiers/detector/recognizer after fine-tuning. We’ve selected the model named “F-RCNN inception v2”.

1. *Transfer Learning*

Transfer Learning makes use of knowledge gained while solving one problem and apply it to a different but related problem. With this technique, we can save a lot of time. In this technique below first, we select any pre-trained model (In which all the parameters are trained), then we perform fine- tuning. We fetch the new dataset to fine-tune the pre-trained CNN. If the new dataset is similar to the original dataset (with which the model has been trained), the same weights can be used for extracting the features from the new dataset. In our case, the dataset is very different from the original dataset. The Earlier layers of CNN contain more generic features (edge detector, colour blob detectors), but the later layers of CNN become progressively more specific to the details of the classes contained in the original dataset. So, earlier layers can help to extract the features of the new data. We fixed the earlier layers and Re-train the rest of the layers (because of the small amount of data).

1. *F-RCNN (Faster Region-based Convolutional Neural Network)*

It stands for Faster Region-based Convolutional Neural Network. Before talking about Faster R- CNN we should know about Fast R-CNN. Fast R- CNN is a detector that uses an external proposal or

external selective search. It consists of External selective search, CNN with max pooling, ROI (Region of Interest) pooling layer, fully connected layers, and output layers. Fast R-CNN takes an input image and then with the help of CNN & max- pooling layers, a convolutional feature map is extracted from the image. The ROI pooling layer performs a very important task here. We know that fully connected layers can accept only certain sizes, So ROI pooling layers converts the output of the CNN into certain fixed sizes. When we put Fast R- CNN with the RPN (Region proposal network), it becomes Faster R-CNN. So, basically, the difference between Fast R-CNN and Faster R-CNN is the Region proposal. In Fast R-CNN, there is an external selective search whereas in Faster R-CNN RPN is combined with the Architecture.

RPN is the Architecture that makes Fast R-CNN a Faster R-CNN. In order to reduce the computational complexity and requirement, RPN decides where to look in the image. It scans the image and gives k output boxes each with 2 scores indicating the probability of availability of an object. Different size and different aspect ratio of boxes are selected in order to accommodate different types of objects.

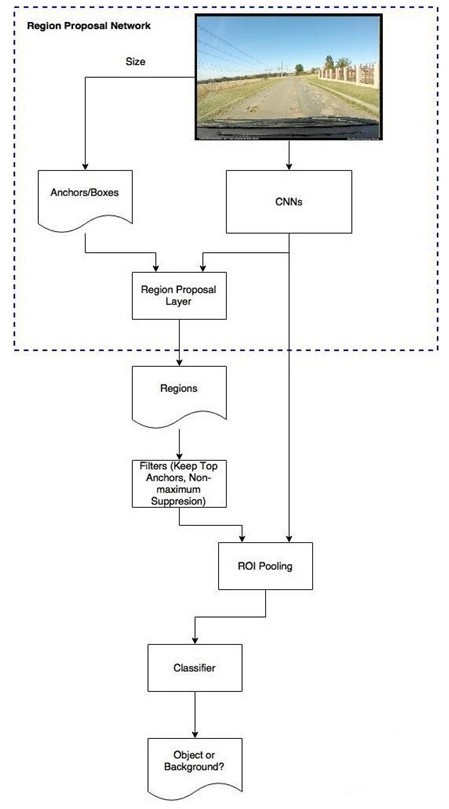


Fig.1 The Architecture of F-RCNN

In order to get a trained working Faster R-CNN architecture, we need to perform the following steps –

* + Training of RPN – first of all, we need to Train the RPN architecture with the dataset so that it can propose the expected region.
  + Training of Fast R-CNN – As we know, Faster R-CNN is a combination of RPN and Fast R-CNN. So, we’ve to train a Fast R-CNN with the proposals obtained by RPN (after training) in order to make a Faster R-CNN.
  + Fixing Convolutional layers, fine-tuning unique layers to RPN.
  + Fixing Convolutional layers, fine-tuning fully connected layers of Fast R-CNN.

1. *Inception – V2*

Inception v2 is an upgraded version in the inception network series. It is a complex architecture of CNNs. Inception v1 is the first version of this series. Inception v1 performs convolution on input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The Outputs are concatenated and sent to the next inception module. Fig.2a shows a typical inception v1 module. As the deep learning networks are computationally expensive, an extra 1x1 convolution is added before the 3x3 and 5x5 convolutions to limit the number of input channels (Fig. 2b). Using this inception module, GoogLeNet neural network architecture (having 9 such inception module) was built. Furthermore, to improve computational speed inception v2 architecture is introduced as shown in figure 03. In this architecture 5×5 convolution is factorized to two 3x3 convolution operations. A 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. Moreover, a new method is introduced to improve performance. In the new method, the convolution of filter size n×n is factorized to a combination of 1×n and n×1 convolutions. This method is 33% cheaper than the single 3×3 convolution.

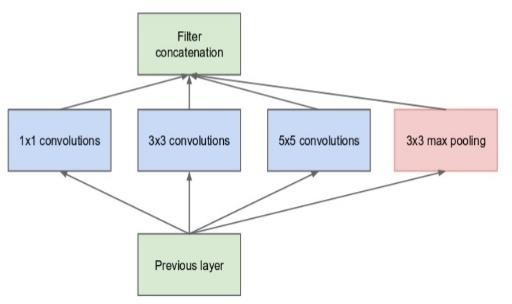


Fig.2a Inception-v1 Module, naïve version

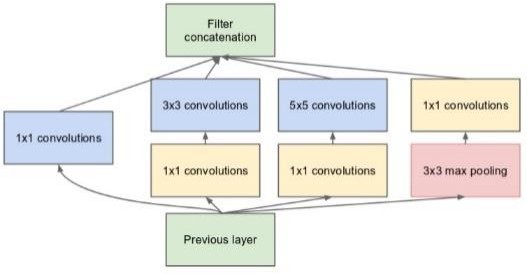


Fig.2b Inception-v1 module with dimension reduction

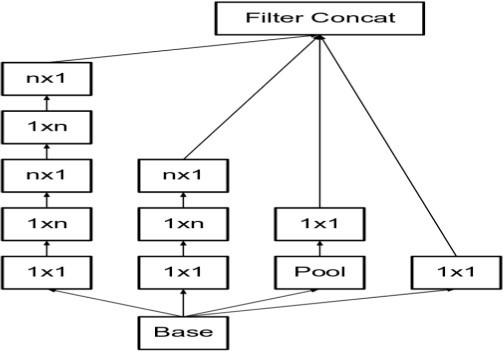


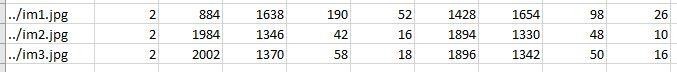
Fig.3 Inception-v2 version

1. Data Pre-processing

We had the pothole dataset consisting of more than 1500 images. This dataset has been compiled at the Electrical and Electronic Department, Stellenbosch University in 2015. The entire dataset consists of two different sets, one was considered to be simple and the other more complex as shown in figure 04. The pothole detection task was found to be much easier if only the region in the images that contained the road was cropped and then used in detection but it takes more time to crop 1500 images. We perform labelling in all images. In the task, we create a CSV file containing images name, path, image dimension and coordinate of the pothole in the image. After that, we create TfRecords for the dataset for feeding it to the Inception of architecture and fine-tuning. The first column in the annotation file contains the path of images. The second column contains a number of potholes in that image and consecutive 4 columns indicate the coordinates of potholes in the image. The size of the image is 3680 × 2760.

Fig.04 Images from Pothole Dataset



Sample Annotations (csv)

1. Detection of Pothole in image/Video

After pre-processing the data, we created a system for the detection of potholes in video/images so that the driver can get an early alert about the pothole on the road. For the system, we have used inception v2 architecture and perform fine-tuning. We fixed the earlier layers of inception v2 network (consisting of 22 layers excluding pooling layers) and re-train only the later layers. For the training, a special configuration file has been created. There is only one class in our configuration file (pothole). Training of the model takes some time depending upon the hardware configuration. After training, the testing process has been done and some extra code has been written to capture video and to detect pothole in the video as well as in image (we actually need “detection of the pothole in videos”). Testing the system shows good performance but it can be improved further. Figure 05, shows the result in the testing process.



Fig.05 Performance of the model

1. Evaluation in real-time

To Evaluate the system, we perform the following set of tasks:

* Setting up a laptop and camera on a vehicle (as we’ve not deployed the system in any android device or raspberry pi yet).
* Region-based
* Detecting potholes in real-time using the system in videos captured by the camera.
* Matching the captured data with real data of road.

1. CONCLUSION

In this paper, we proposed a system to detect potholes in real-time in images/videos captured by a camera mounted on the vehicle and to give an alert to the driver about the pothole on road in front of the vehicle. Further, our system will detect the location of the pothole and upload the same on map (reflected in android app developed by us) so that other users who have no camera mounted on their vehicle can get alert about the pothole using the app only (however this is our future work). In this system, we used famous and complex CNN architectures like Inception v1 (GoogLeNet), inception v2 and finally select inception v2 in our system. In the experiment data, the system shows extremely good results. Our Future works include experiments for improving the system and making it simple for common/general use (Read more in future works section).

1. Future Work

Our system successfully detects pothole in videos/images but there are some works yet to be done. Those are our future works regarding pothole detection. We’ll try another good CNN architectures like the latest versions of inception (inception v3 and inception ResNet) for improving the speed and accuracy. Additionally, a GPS system has to be created for updating the location of potholes on the map for all users. In near future, we’ll create a GPS enabled system and an android app (with GPS and google map enabled) with which we will update the location of the pothole to the map when a user finds it on the road so that other users get the location of potholes earlier without any detection. We also have to deploy our system to raspberry pi or Android so that common people can use that.

REFERENCES

1. https://[www.ibef.org/industry/roads-](http://www.ibef.org/industry/roads-) presentation
2. https://sodhganga.inflibnet.ac.in/bitstream/10603/747 47/10/10/ chapter-3.pdf
3. https://[www.thehindu.com/opinion/op-](http://www.thehindu.com/opinion/op-) ed/fix-the- pothole-problem/article24436287.ece
4. Artis Mednis, Girts strazdins, reinholds zviedris, georgijs kanonirs, leo selavo Digital Signal processing laboratory, Institute of Electronics and computer science, *“Real time pothole detection using android smartphones with accelerometers”*, 14 December str., Riga, LV 1006, Latvia
5. Lin, J., & Liu, Y. (2010, August). Potholes detection based on SVM in the pavement distress image. In *2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science* (pp. 544-547). IEEE**.**
6. Pereira, V., Tamura, S., Hayamizu, S., & Fukai, H. (2018, July). A Deep Learning-Based Approach for Road Pothole Detection in Timor Leste. In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)* (pp. 279-284). IEEE.
7. S. Nienaber, M.J. Booysen, R.S. Kroon, “*Detecting potholes using simple image processing techniques and real-world footage*”, SATC, July 2015, Pretoria,

South Africa

1. S. Nienaber, M.J. Booysen, R.S. Kroon, “*A comparison of Low-cost Monocular Vision Techniques for Pothole Distance Estimation*”, IEEE CIVTS, December 2015, Cape Town, South Africa