# Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

| | |
|---|---|
| **Course code and name:** | F20PA Research Methods and Requirements Engineering |
| **Type of assessment:** | **Individual** |
| **Coursework Title:** | Collabarative Road Condition Monitoring Application for Pothole Detection |
| **Student Name:** | Teo Say Yong |
| **Student ID Number:** | H00390718 |

**Declaration of authorship. By signing this form:**

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

**Student Signature** *(type your name):* *Adison*

**Date**: *15/04/2024*

# Collaborative Road Condition Monitoring Application for Pothole Detection

April 15, 2024

BSc (Hons) Computer Science

Final Year Dissertation

*Supervised by* John See

Heriot-Watt University

School of Mathematical and Computer Sciences

Department of Computer Science

April 2024

# Abstract

The goal of this dissertation is to revolutionize road maintenance and safety through the development of a novel pothole detection model. With the help of state-of-the-art machine learning techniques, like YOLO (You Only Look Once), and an intuitive graphical user interface built with Tkinter, the model can identify potholes in real time from live video streams. The main goal of the project is to develop an efficient and user-friendly system that enables users of all skill levels to take proactive measures to address the problem of potholes on roads. The model shows a great deal of promise for boosting road safety and infrastructure management through thorough research, development, and testing. The dissertation provides a thorough implementation and future development roadmap by outlining the model's methodology, implementation process, evaluation metrics, and outcomes. With broad ramifications for road safety and maintenance procedures, this dissertation establishes the groundwork for a more proactive and effective approach to pothole detection and infrastructure management.

## Declaration

I, Teo Say Yong confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Adison

Date: 15/04/2024

# Table of Contents

# List of Figures

# Chapter 1 Introduction

Our roads' condition has become a major concern in a time of rapid urbanization and technological advancement. A common hazard, potholes not only jeopardize traffic safety but also result in high repair and upkeep costs for infrastructure and automobiles. With the advent of Collaborative Road Condition Monitoring Applications, a promising solution to the ubiquitous problem of pothole detection has become apparent, demonstrating the need for creative solutions.

The way we approach road maintenance has changed dramatically with the release of the Collaborative Road Condition Monitoring Application for Pothole Detection. Road monitoring has historically been the province of local government, which frequently relies on irregular inspections and reports from the public. The goal of the collaborative application is to transform this procedure by utilizing real-time data analysis, community involvement, and technological advancements.

This application uses modern sensor technologies (GPS and accelerometers) that are built into cars and smartphones (preferrably dashcam) to automatically detect and capture image of the road conditions. Through the use of these widely available devices, the application turns every individual using the road into a potential participant in the group's effort to monitor the state of the roads. Beyond individual contributions, collaboration fosters a network effect that improves pothole detection's accuracy and comprehensiveness.

The foundation of this application is real-time data transmission, which allows for immediate updates on road conditions. Municipal authorities can then prioritize and quickly fix potholes because they have access to a dynamic and constantly changing dataset. It is because the application is collaborative in nature, users' devices actively report road anomalies and contribute to the general betterment of their communities, which in turn fosters a sense of civic responsibility.

The effectiveness of this cooperative approach is largely dependent on machine learning algorithms. These algorithms can accurately detect patterns and anomalies that point to potholes

because they were trained on large datasets. The algorithms constantly improve their models as the application gathers more data, guaranteeing an ever-improving capacity to identify and forecast the occurrence of potholes.

## 1.1. Motivation

Potholes are a prevalent issue on roads worldwide, and they can have detrimental effects on traffic flow, road safety, and infrastructure upkeep. Image detection technology has become a viable tool for locating and monitoring potholes on roadways in recent years. The motivation of this report is to let workers who do maintenance of road surfaces acknowledge the location of the potholes that was sent by whoever used this prototype.

## 1.2. Aim

The goal of this research is to create a reliable system that can detect and classify potholes even when they are partially hidden from view by using computer vision techniques and deep learning algorithms. The main goal is to put into practice a model that can detect potholes on roadways, with a special emphasis on situations in which the potholes could be partially hidden by debris or masked by surrounding elements. The purpose of this system is to be an essential part of a driver assistance system by giving drivers real-time alerts to help them drive safely and prevent possible damage to their cars. Considering how crucial it is to find potholes as soon as possible to avoid collisions and reduce road damage.

## 1.3. Objectives

Objective 1: Gather road photo datasets with different kinds of potholes, lighting conditions, and road conditions.

Objective 2: Pre-process the gathered data by resizing, normalizing, and augmenting it to guarantee ideal training conditions for the image detection model

Objective 3: Examine and select an appropriate deep learning architecture for image identification, taking into account variables like precision, computational effectiveness, and ease of adaptation to real-world scenarios.

Objective 4: Use the prepared dataset to train the model, adjusting its parameters to achieve high pothole detection precision and recall.

Objective 5: Assess the developed model's performance using appropriate metrics, including the area under the ROC curve, precision, recall, and F1-score.

Objective 6: Evaluate the efficacy and efficiency of the suggested image detection system by conducting comparative studies with current models or techniques.

Objective 7: Construct a real-time image processing system that uses the trained model to detect potholes instantly.

Objective 8: Evaluate the system's functionality in actual use, taking into account variables like accuracy, latency, and environment adaptability.

## 1.4. Document Overview

The document has 5chapters:

Chapter 1 is the introduction of the project containing the motivation, aim and objective of the project.

Chapter 2 is the background, risk of pothole and literature review regarding the research of object detection, light detection and ranging, YOLOX (You Only Look Once) model,different type and sizes of pothole. These researches concludes the crucial review of the related work and the model architecture that need to be used.

Chapter 3 describes how the project was carried out. This involves putting the pothole detecting model into practice.

Chapter 4 provides an analysis of the tests, studies, and analytical findings related to the pothole detecting model.

Chapter 5 concludes with a discussion of the difficulties encountered during the implementation phase, the project's limitations, and future work and reflection.

# Chapter 2 Background

Pothole detection techniques have evolved over time, moving from manual inspection methods to sophisticated machine learning approaches. Road maintenance was initially primarily done through labor-intensive manual inspections, which introduced subjectivity and lengthy procedures. A wider view was offered by the integration of remote sensing technologies, although resolution issues were a problem. Early attempts at automation using simple image processing algorithms appeared as computing power increased. The real innovation, though, was in the use of machine learning, as supervised learning models started to identify patterns linked to potholes. With the introduction of deep learning and convolutional neural networks, which provided previously unheard-of accuracy, the evolution proceeded. Despite these developments, problems like changing environmental conditions and the requirement for real-time detection still exist. As a result, research and development are continuously conducted in an effort to create pothole detection systems that are more reliable and effective.

Road potholes are never a good thing for any kind of car. Most drivers tend to avoid the ones that are visible to them, but occasionally they go unnoticed, resulting in unintentional damage to tires and cars. Not only can potholes cause damage to cars, but they can also cause damage to trucks, motorcycles, and even bicycles.

## 2.1. Pothole Risks

The risk of pothole may sustaining serious structural damage from an abrupt and violent bump to a car wheels. Consequently, this damage may result in a series of catastrophic human injuries and mechanical issues, such as the following: [18]

- Wheel damage and tire blowouts. The deep pothole's sharp edges have the potential to split or puncture tires when they go through it. A blowout and the sudden release of air are possible outcomes of any structural damage to the tire. A blowout can result in more damage than just a flat tire; it can also cause your car to swerve into oncoming traffic, a ditch, or the median.

Reports of studies claimed that potholes result in accidents by causing a loss of control. Potholes caused 0.8% of traffic incidents in 2021 that resulted in 1.4% fatalities and 0.6% in injuries [20]. It has been demonstrated that imperfections in road surfaces cause emissions to rise by 2.49% and vehicle speeds to drop by 55% [27]. According to a report by the Canadian Automobile Association (CAA) [29], potholes result in an annual increase in operating expenses of $3 billion. Furthermore, bad road conditions cause traffic jams and driver annoyance, which can lead to risky driving practices [9]. Accurate traffic flow models are required to mitigate these issues.

## 2.2. Literature Review

Numerous fatalities have been caused by road damage, so it is more important to detect and warn of damaged roads. Due to high-distance transmission, the majority of road damage detection systems store and process data in the cloud, which increases latency. The cloud's mass storage and processing capabilities, along with its ability to respond quickly, are leveraged by the proposed ECRD detection and warning system. This study's first contribution is a road segmentation method for quick and accurate road area identification. Next, for fast identification of unsafe road damage, a lightweight road damage detector is constructed using level concurrence matrix features at the edge. [33]

Byeong-ho Kang et al.'s method [12] makes use of a 2D LiDAR (Light Detection and Ranging) sensor and camera to locate and measure potholes. Techniques for edge detection and noise filtering are applied to the images that the camera takes. The world is rapidly moving toward autonomy, which is why there is a growing need for it. Pothole detection is the main application of the Least Square Support Vector Machine, a machine learning technique proposed by Nhat-Duc Hoang [13]. The model can only be used to predict a limited variety of potholes because the training dataset only contains 200 photos. Furthermore, training support vector machine-based systems takes longer.

In a country like India where roads span millions of kilometers, K.C. et al.'s method [5] for illustrating a system for identifying potholes on the road is difficult to implement. As a result, pothole detection must be automated quickly and accurately in real time. YOLOX (You Only Look Once) is an object detection algorithm whose main goal is to train and analyze the YOLOX model for pothole detection. The image processing methodology uses a range of statistical techniques, such as Gray-Level Co-Occurrence (GLCM), Radial Basis Function

(RBF), etc., to extract and distinguish various textures and features of an image. Owing to the massive processing power requirements, these methods are often used in conjunction with other machine learning approaches. The data set used to train the model contains numerous potholes as well as photos of potholes in various shapes. According to the analysis, the YOLOX nano model is the best option for pothole detection because it is small, requires little storage space, and uses little energy. It can be quickly deployed.

A proposed system that makes use of the deep learning-based YOLO (You Only Look Once) pothole detection method is described by P. A. Chitale et al. [21]. The proposed technology yields reassuringly accurate pothole detection findings. The recommended method helps to reduce the amount of time required for road upkeep. The technique uses a specially constructed dataset containing images of different types and sizes of potholes, both wet and dry. The proposed technology would reduce the need for human laborers to maintain roads, especially during an epidemic. Potholes are identified, and their diameters are measured with remarkable precision and a notably low mistake rate. The estimated pothole measurements may be used to calculate the quantity of raw materials required to patch the holes as well as to assess the extent of the road's damage. This means that most of the planning and inspection could be completed online.

To identify road distresses like potholes, patches, and fractures, Lokeshwor Huidrom et al. [19] have proposed a system that makes use of predefined thresholds for standard deviation and object circularity. This is accomplished by using image processing algorithms. This method's disadvantage is that because road distresses vary in size and shape, not all of them can be classified under the same set of established criteria.

## 2.3. Object Detection

The field of computer vision called object detection is dedicated to the task of identifying things in pictures or videos. Road signs, vehicles, people, and animals are examples of objects. The two basic phases in object detection are object localization, which locates an object in a given image or video, and image classification, which determines what the object is. Because CNNs can learn and then classify, the second step of identifying the item is comparatively easier, especially with their assistance. A CNN's layers can be utilized to distinguish between different parts of the image, such as edges, texture, and brightness, before determining what the identified item is in

the end. Finding an object in a picture or video is the former, and it is the most difficult task.

[8] have suggested use the gradient distribution to identify trends. Once more, a sliding window with cell divisions is used in this technique [22]. The pixel edge orientations and gradients are used to compute a histogram. A feature is created by combining the histograms of all the cells. Prior to obtaining the final feature vector, local contrast normalization is carried out in order to improve accuracy. While HOG is slower but generally more precise than Haar-like, it is faster.

### 2.3.1. RCNN

[23] employs "recognition using regions" from [7] to leverage CNN for object detection. With a particular input image, their model produces 2000 separate regions, which are then sent to a CNN. To extract features from each of those regions, they have employed CNN with Caffe [32]. Their CNN consists of two fully linked layers and five convolutional layers. The vectors containing the collected characteristics are input into a "set of category-specific linear SVM's" to produce a score for a particular area. Their approaches were quick when they were first introduced, but they have two drawbacks: first, they include three steps, and second, the first two stages of detection process process undesired regions.

### 2.3.2. Fast-RCNN

In contrast to [23], a single stage method is proposed in [**?**]. Their algorithm takes an input image and a set of item proposals, and first creates a CNN feature map with multiple convolutional layers and a pooling layer. A region of interest pooling layer, or RoI, extracts a feature vector for every object proposition. This layer creates a feature map from the features in a ROI by using max pooling. This feeds into two output levels using fully connected layers (FC). The softmax layer predicts the class in the Roi, and another layer outputs the locations of the bounding boxes for the classes that are identified. Fast RCNN is quicker than the original RCNN, but it has limitations because it requires the input image and object proposals. Another drawback is that because it is a sequential process and employs selective search, it is slow for big datasets.

### 2.3.3. Faster-RCNN

[25] replaced selective search with Region Proposal Networks (RPNs) which uses convolutional feature maps to generate region proposals. So Faster RCNN, works with two modules, one is the RPN that gives object proposals which are passed onto the Fast RCNN detector and follow the same step mentioned the previous section. In short RPM tells the Fast RCNN detector what region to look for the object. Since RPN is a neural network it can be trained for specific objectives like for example pothole, man-made pothole etc and will thus perform better with more training. To prevent the unwanted processing of regions without objects as done in 2.3.1, a feature vector is generated that is passed onto two layer, cls and reg. which generates a score of whether the region contains the object or the background and bounding box. Selective search trained on PASCAL VOC 2007 and 2012 gave mAP (mean Average Percesion) of 70% while RPN with VGG trained on the same dataset gave 73.2% mAP. Therefore, Faster RCNN is great improvement from RCNN that started it all

### 2.3.4. ResNet

Gradients disappear or explode as the layer depth increases. The paper proposes a 'deep residual learning framework' to address the latter problem, while normalization layers can solve the former. Degradation occurs when connections between layers are not optimal. ResNet uses shortcut connections to learn optimal connections and skips unnecessary ones using the identity function.

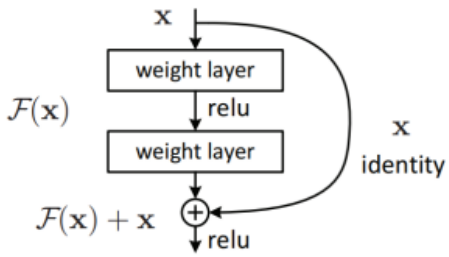



**Figure 2.1:** From [16] Presents the skip connection

Using this concept, a 34-layer plain network achieved a top-1 error of 28.54, while the same network with ResNet achieved a top-1 error of 25.03 during ImageNet validation. They conducted additional experiments (see table 2.2) with a larger number of layers and discovered that 110 was optimal.

| method | | | error (%) |
|---|---|---|---|
| Maxout [10] | | | 9.38 |
| NIN [25] | | | 8.81 |
| DSN [24] | | | 8.22 |
| | # layers | # params | |
| FitNet [35] | 19 | 2.5M | 8.39 |
| Highway [42, 43] | 19 | 2.3M | 7.54 (7.72±0.16) |
| Highway [42, 43] | 32 | 1.25M | 8.80 |
| ResNet | 20 | 0.27M | 8.75 |
| ResNet | 32 | 0.46M | 7.51 |
| ResNet | 44 | 0.66M | 7.17 |
| ResNet | 56 | 0.85M | 6.97 |
| ResNet | 110 | 1.7M | **6.43** (6.61±0.16) |
| ResNet | 1202 | 19.4M | 7.93 |

**Figure 2.2:** From [16] Result of experiment on CIFAR-10

### 2.3.5. DenseNet

[11] describes an architecture where a layer processes data and concatenates the outputs of previous layers with the current one. To optimize, two new parameters were introduced: Bottleneck Layers, which reduce the number of input features for the next layer, and Compression factor, which determines the amount of compression applied to layer outputs. The network's dense structure reduces the number of layers and parameters, reducing the risk of overfitting. ResNet (2.3.8) had 30M parameters, while DenseNet had 15.3M during testing. This model uses "implicit deep supervision," which ensures that all layers are aware of the important inputs and decisions made by previous layers. This approach improves feature classification, especially in scenarios with limited training data.

### 2.3.6. SqueezNet

[10] aims to reduce network size while maintaining the same accuracy as AlexNet [1]. The approach consists of three strategies: reducing hyperparameters, replacing 3x3 layers with 1x1, and reducing input layers. To compensate for reduced network depth, a third strategy involves down sampling late in the network. This exposes more layers to larger activation maps, resulting in higher accuracy.

They introduced the fire module, which has three parameters, to accomplish this. In order to implement strategy three, the number of filters per fire module will rise as the network becomes deeper. As can be seen in Figure 2.3, SqueezeNet outperformed AlexNet in terms of model size prior to compression and achieved an identical accuracy of 80.3% on the ImageNet Dataset (ILSVRC 2012).

| CNN architecture | Compression Approach | Data Type | Original → Compressed Model Size | Reduction in Model Size vs. AlexNet | Top-1 ImageNet Accuracy | Top-5 ImageNet Accuracy |
|---|---|---|---|---|---|---|
| AlexNet | None (baseline) | 32 bit | 240MB | 1x | 57.2% | 80.3% |
| AlexNet | SVD (Denton et al., 2014) | 32 bit | 240MB → 48MB | 5x | 56.0% | 79.4% |
| AlexNet | Network Pruning (Han et al., 2015b) | 32 bit | 240MB → 27MB | 9x | 57.2% | 80.3% |
| AlexNet | Deep Compression (Han et al., 2015a) | 5-8 bit | 240MB → 6.9MB | 35x | 57.2% | 80.3% |
| SqueezeNet (ours) | None | 32 bit | 4.8MB | **50x** | 57.5% | 80.3% |
| SqueezeNet (ours) | Deep Compression | 8 bit | 4.8MB → 0.66MB | **363x** | 57.5% | 80.3% |
| SqueezeNet (ours) | Deep Compression | 6 bit | 4.8MB → 0.47MB | **510x** | 57.5% | 80.3% |

**Figure 2.3:** Comparison on SqueezeNet and model compression

### 2.3.7. You Only Look Once (YOLO)

[15] is a high accuracy, single-stage detection system with low processing burden. Unlike other Two Stage approaches where the first stage involves determining the object's position, the model uses the complete image during training. In this case, the input image is split into grids, and each grid projects a bounding box and a confidence value, which is the predicted object's intersection with ground truth.

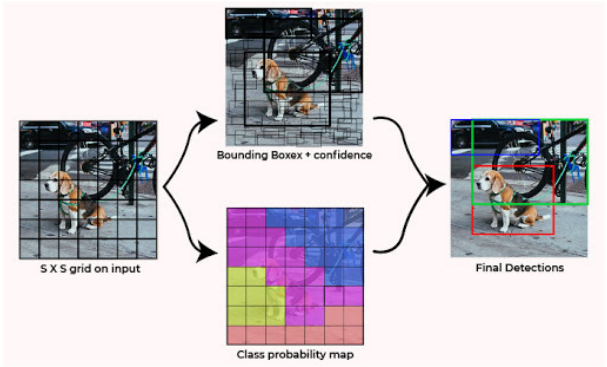



**Figure 2.4:** Process of object detection using YOLOv1

Every bounding box outputs its width, height, confidence score, and center of the box coordinates. For every grid cell, a set of class probabilities is computed (not for bounding box). The figure 3.3 illustrates these. Prediction head uses data from both processes to make the final choice. On Pascal VOC 2007, they evaluated a model with 24 convolutional layers and 2 fully linked layers. YOLO has a mean average precision score of 63.4. In comparison to real-time detectors such as DPM, Yolo achieved higher FPS (155 versus DPM's 100) and mean average precision (52.7 versus 16.0) with Fast RCNN and Faster RCNN.

Unlike other models, the full image is used for training instead of only a portion where objects might be present, which allows for the acquisition of more contextual information and explains the high accuracy. YOLOv5, which runs at 140 frames per second, is faster than

YOLOv4, which runs at 40 frames per second and is about 90% smaller. This is because YOLOv5 employs batching. The YOLOv5 is enhanced by TPH-YOLOv5 [31]. Transformer Prediction Heads (TPH) take the place of the prediction layers. Subsequently, they added an additional layer of prediction for smaller things, bringing the total number of objects to four (tiny, small, medium, and giant). They made use of CBAM (Convolutional Block Attention Module) from [26], which creates an attention map for photos that span a wide area when given a feature map.

#### 2.3.8. SSD

Another one-stage moded system, [30], is constructed on top of VGG-16 [28], but lacks fully connected layers. The feature map that is produced by the VGG-16 base network is then fed into a mutilbox predictor. Bounding boxes with varying aspect ratios are constructed for every place, based on which the class score is calculated. When YOLOv3 and SSD were tested in [4], it was discovered that while both performed similarly, YOLO had greater accuracy and SSD had fewer false positives. Considering that YOLOv5 has improved over YOLOv3, we may assume that it will perform significantly better than SSD.

### 2.4. Training Method

#### 2.4.1. Online Hard Example Mining (OHEM)

To train CNN, they suggest an algorithm in [3] that is an adaptation of SGD. Their method seeks to minimize the numerous steps needed for SGD training. They suggest sampling non-uniform samples from training examples according to the example's current loss. The algorithm computes the loss for each image after forcing the network to perform a feedforward on all of the input images.

Following the computation of all training examples, the images with the highest loss are categorized as negative examples by sorting them using NMS. The model is then trained using this. They were able to improve Fast RCNN's accuracy by 3-4% on PASCAL VOC 2007 and 2012 by employing this technique. By applying it to MS COCO, OHEM was able to achieve a 2.9% increase in mean average precision.

As most approaches fail when tested on datasets they haven't been trained on, [14] suggests

that training a pothole detection model should be generalized. It has been demonstrated that when tested on a large and diverse dataset, general object detection models outperform pothole detection models. Their argument—supported by the findings of experiments—is that object detectors are more capable of generalizing, making them more appropriate for spotting pothole in novel situations.

They have suggested a training pipeline that calls for first training the model on a broad and varied dataset and then fine-tuning it on a dataset that is specifically targeted at potholes.

### 2.4.2. Transfer Learning

This is a common machine learning technique in which a previously trained model serves as the foundation for a newly created model [6]. Therefore, the weights for a similar model that is trained for a similar task are obtained from a model that was trained for that task.

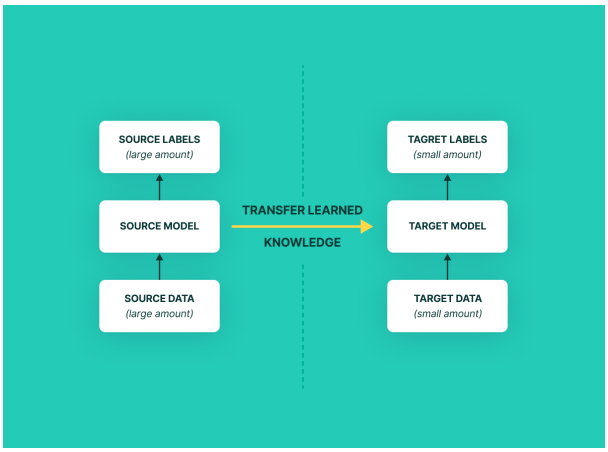



**Figure 2.5:** Transfer Learning Diagram from [6]

### 2.4.3. Bag of freebies and tricks

Outlines a number of methods in [16] for increasing a CNN model's accuracy without compromising its performance (model complexity). With the introduction of the "bag of freebies," [34] enhances [16] and is primarily focused on object detection. The model trained (YOLOv3) on Pascal VOC 2007 was found to perform better in the "Elephant in the room" [2] test (which involves artificially inserting an object into a scene and measuring the number of frames the new object is detected for) with a recall score of 94.12 compared to 42.95. This was achieved through the use of interpolated training images, which involves combining images.

This increases the model's resilience to identify objects in previously unseen scenarios. Ad-

ditional methods include data augmentation, as described in [1], warm-up learning rate, synchronized batch normalization (which improves training time by standardizing inputs to layers from layers of varying sizes), and randomly alerting the network's shape. By using this method, they were able to increase Pascal VOC 2007 by 3.43 and YOLOv3 on MS COCO by 4.

## 2.5. Datasets

This link directs to the dataset which contains 720 images for the pothole detection system to train.

## 2.6. Conclusion

To sum up, the literature review emphasizes various methods of identifying potholes and road damage, highlighting the need to act quickly to reduce the number of fatalities and infrastructure problems brought on by deteriorating roads. By utilizing cloud-based processing capabilities, the proposed ECRD detection and warning system addresses the latency problem in current systems and provides a comprehensive solution for rapid and precise road area identification. Numerous approaches, including LiDAR-based strategies and deep learning models like YOLOX and YOLO, demonstrate technological breakthroughs in the accurate and effective detection of potholes. A foundation for efficient road maintenance is laid by the proposed project's integration of a model within a website, which enhances the practicality of real-time pothole detection and satisfies the need for accessibility across multiple devices. This strategy, which is based on the knowledge gained from the literature, has the potential to provide efficient, accessible, and user-friendly road damage detection, which will ultimately lead to better-maintained and safer road infrastructure.

# Chapter 3 Project Implementation

## 3.1. Hardware and Software Prerequisites

### 3.1.1. Hardware

In order to process and analyze sensor data efficiently, certain hardware requirements must be met. Using Radeon Vega Mobile Graphics in conjunction with an AMD Ryzen 7 3750H processor provides a powerful computing platform for real-time data processing and analysis. Machine learning algorithms require a processor that can balance multi-core processing power and energy efficiency, and the AMD Ryzen 7 3750H delivers just that. By speeding up parallel processing tasks, its Radeon Vega Mobile Graphics further improves the system's capabilities. This is especially helpful for image and video analysis needed for pothole detection.

### 3.1.2. Software

The main project is utilize in Python, specifically version 3.11.8, as the primary programming language for project development. Leveraging PyTorch's YOLO framework, I trained a model specifically tailored for pothole detection. Additionally, I incorporated tkinter, a Python GUI toolkit, to facilitate the creation of a user-friendly interface for the model. This GUI allows users to interact with the pothole detection model seamlessly, and by exporting it as a .exe file, it ensures the capability to run the application on various platforms without requiring Python or additional dependencies.

## 3.2. Source Code

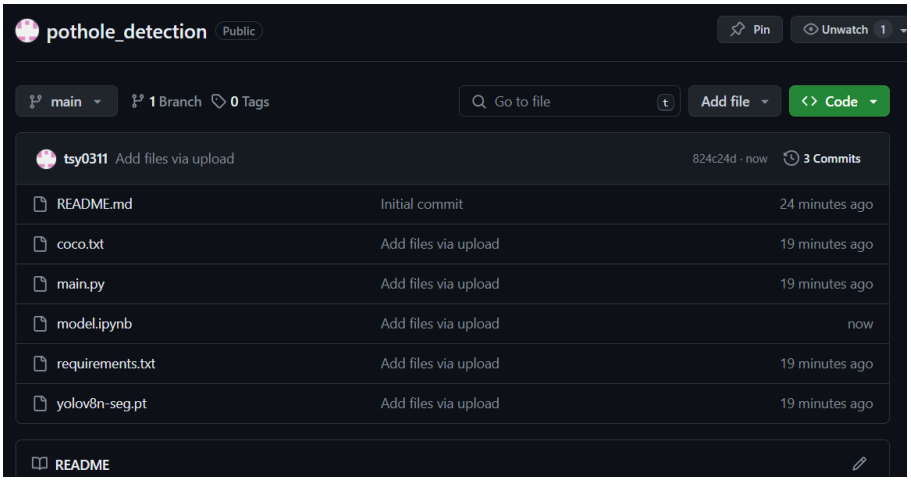



**Figure 3.1:** GitHub Repository

The source code is made available in a public GitHub repository (Shown in Figure 3.1). The dataset was not uploaded as it was from public dataset and this is the link to download. There is a requirement.txt file (Shown in Figure 3.2). This is require to do so that the code can run successfully. Use command line type the command "pip install -r requirement.txt" to download the required libraries and dependencies. For example, (path/to/the/file/requirement.txt).

```
1  # requirements
2  # Usage: pip install -r requirements.txt
3
4  # Base ----------------------------------------
5  ultralytics
6  tk
7  opencv-python
8  pillow
9  requests
```

**Figure 3.2:** Requirement File

## 3.3. Pothole Detection

### 3.3.1. Data Collection and Pre-Processing

The dataset used to train the pre-trained model is from here. The train folder is the main folder to do the training for the model. The dataset contain 720 images and 60 valid images. With this number of images to train, the model is able to train well with confidence rate of above 0.7.

Auto-orientation of pixel data, which involved reorienting the image in accordance with the

EXIF (Exchangeable image file format) metadata, was one of the pre-processing steps applied to each image. This ensures that the image is displayed in the correct orientation regardless of how it was originally captured. This stage is essential for preserving the images' consistency and visual integrity across various platforms and devices. The pictures were also resized to a fixed size of 640 by 640 pixels. Resizing the photos to a common size makes handling and processing them easier and guarantees consistency in how they are presented and analyzed.

## 3.4. Methodology

### 3.4.1. System Architecture Design

- The architecture of the pothole detection system is designed to facilitate seamless integration of data processing, model inference, and user interaction components.
- The system consists of three main modules: the data processing module, the detection module, and the user interface module.
- The data processing module is responsible for pre-processing input data, including live video streams or static images, to prepare them for inference by the detection model.
- The detection module encompasses the trained YOLOv8n segmentation model and associated algorithms for real-time object detection and classification.
- The user interface module, built using tkinter, provides a graphical interface for users to interact with the system, visualize detection results, and initiate actions such as saving or exporting results.

### 3.4.2. Data Processing Pipeline

- Within the data processing module, a robust pipeline is established to handle input data streams efficiently.
- This pipeline includes stages for data acquisition, pre-processing, and transformation, ensuring compatibility with the input requirements of the detection model.
- Techniques such as frame extraction, resizing, and normalization are applied to optimize data quality and facilitate seamless integration with the detection module.

### 3.4.3. Detection Model Integration

- The heart of the pothole detection system lies in the integration of the YOLOv8n segmentation model for real-time pothole detection.
- The model is seamlessly integrated into the architecture, leveraging its capabilities for accurate and efficient detection of potholes within input data streams.
- Techniques such as transfer learning and model optimization are employed to adapt the pre-trained model to the specific task of pothole detection, enhancing its performance and robustness.

### 3.4.4. User Interface Design

- The user interface module plays a crucial role in facilitating user interaction and feedback within the system.
- Through the use of tkinter, a user-friendly graphical interface is developed, featuring intuitive controls, visualizations, and feedback mechanisms.
- The interface allows users to initiate the pothole detection process, monitor live video streams, visualize detection results with bounding boxes and confidence scores, and perform additional actions such as result saving or exporting.

## 3.5. Architecture Of Models

The architecture of YOLOv8n-seg represents a significant advancement in object detection frameworks, combining the strengths of YOLO (You Only Look Once) with semantic segmentation capabilities. Built upon a feature extraction backbone network like Darknet or ResNet, YOLOv8n-seg extracts hierarchical features from input images, laying the groundwork for subsequent layers responsible for object detection and semantic segmentation. With an augmented detection head, YOLOv8n-seg predicts both bounding box coordinates and class probabilities for object detection, as well as pixel-wise semantic segmentation masks to precisely delineate object boundaries within the image. Through feature fusion mechanisms and multi-scale processing, the model achieves robust segmentation performance across a wide range of object sizes and complexities, enabling accurate and detailed object detection and segmentation.

Customizable and adaptable, YOLOv8n-seg can be tailored for specific tasks such as pothole detection. By integrating domain-specific data augmentation techniques, optimizing loss functions, and fine-tuning model parameters, YOLOv8n-seg can effectively detect and segment potholes within road surfaces. Its versatility makes it a powerful tool for building advanced pothole detection systems, offering automated detection and monitoring capabilities to enhance road safety and infrastructure maintenance efforts. With YOLOv8n-seg's comprehensive approach to object detection and semantic segmentation, it represents a significant step forward in the field of computer vision, facilitating more accurate and efficient detection of objects in real-world environments.

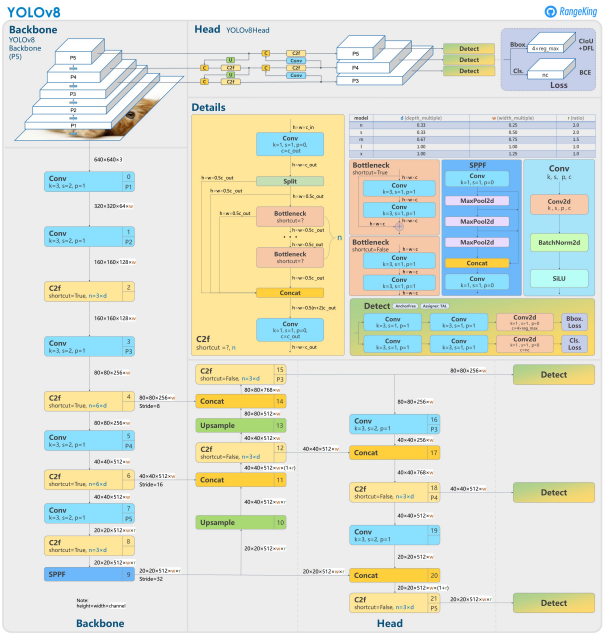



**Figure 3.3:** Architecture of YOLOv8n-seg [24]

## 3.6. Design Decision to Work with Insufficient Hardware And Data

**Decrease Image Size** : Using the actual resolution would extend training time required for processing each image. In order to counter this, the images were resized to 320x320, which greatly decreased the amount of memory used and shortened the training period. By cutting the pixel count in half, this resizing technique maximized memory usage and sped up the training process.

## 3.7. Training Pothole Detection Model

### Training Data Configuration

```
results = model.train(
    data=yaml_file_path,     # Path to the dataset configuration file
    epochs=80,               # Number of epochs to train for
    imgsz=320,               # Size of input images as integer
    batch=8,                 # Number of images per batch
    optimizer='auto',        # Optimizer to use, choices=[SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto]
)
```

**Figure 3.4:** Resize Images

From Figure 3.4, resize was used to reduce the image's scale by half to 256 x 320 (due to hardware limitation).

### Data Augmentation

The techniques employed are:

- Horizontal Flip: Images have a 50% chance of being horizontally flipped, effectively doubling the dataset size and introducing orientation variations.
- Random Cropping: Images undergo random cropping operations, with a range of 0 to 20 percent of the image's dimensions. This allows the model to learn from different areas of the image, improving its generalization capabilities.
- Random Rotation: Images are randomly rotated between -15 and +15 degrees, enabling the model to learn from various perspectives.
- Random Shear: Random shear transformations between -5° and +5° are applied to the horizontal and vertical axes, simulating perspective shifts and distortions commonly observed in real-world scenarios.
- Random Brightness Adjustment: Random brightness adjustments ranging from -25 to +25 percent are applied to account for variations in illumination levels, ensuring the model's robustness to different lighting conditions.
- Random Exposure Adjustment: Random exposure adjustments ranging from -25 to +25 percent are made to simulate exposure variations frequently seen in photos taken under diverse lighting conditions.

**Epochs**

To improve pothole detection performance, we have decided to start an 80 epochs training process. In our training regimen, every epoch is carefully designed to guarantee that the dataset is utilized to its fullest. We create robustness and adaptability in our model by exposing it to a variety of scenarios through the use of augmentation and repetition techniques. Even though 80 epochs may not seem like a long time, the augmentation techniques used greatly increase the effective training period, which improves the model's learning process. In addition, we have strategically modified the scheduler to maximize performance during the whole training process. This methodical approach shows our commitment to gleaning insightful information and optimizing the pothole detection model's efficacy.

### 3.7.1. Hyper Parameters

The model configurations are listed below: 3.7.1

| **Training Configuration** | **YOLOv8n-seg** |
|---|---|
| Images for Training | 720 |
| Images for Validation | 60 |
| Number of Epochs Trained | 80 |
| Image size | 320 |
| Batch size | 8 |
| Optimizer | auto |

## 3.8. Software Interface

### 3.8.1. Create Gui Interface

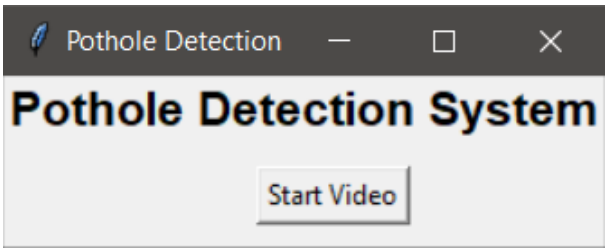



**Figure 3.5:** GUI Interface

The GUI interface's initial setup gives users the flexibility to control the system's continuous processing demands by allowing them to start or stop the video stream. With the help of this feature, users can regulate the streaming video feed according to the available computational power, which makes operations run more smoothly and guarantees that pre- and post-processing tasks are handled effectively.

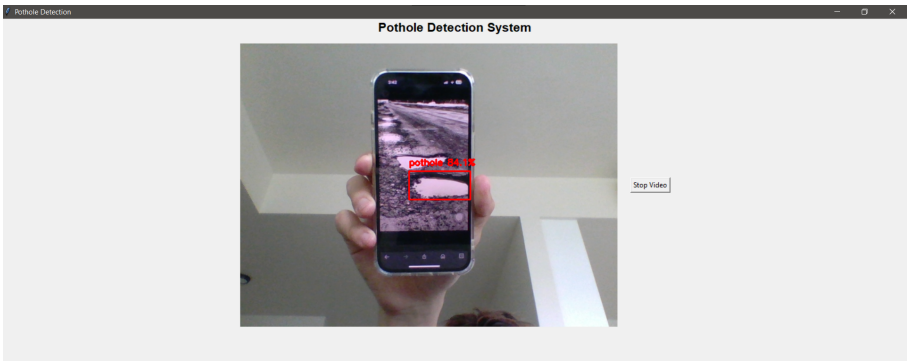



**Figure 3.6:** Results of the detection system with GUI interface

Our GUI interface offers a smooth and user-friendly platform for seamless interaction, all while keeping the user experience in mind. Users can easily understand the functionality and goal of the interface thanks to its simple layout and intuitive navigation. The incorporation of a live stream display affords users an instantaneous perspective of the road conditions, thereby enabling proactive surveillance for potential hazards such as potholes. Furthermore, users have control over the video stream thanks to the stop button, which lets them pause the feed for more in-depth examination or analysis whenever necessary. Our GUI interface puts an emphasis on usability, simplicity, and user control to make sure users have a positive and productive experience.

### Tkinter

The foundation of Python GUI development is Tkinter, which provides programmers with a powerful toolkit that makes creating complex graphical user interfaces simple. Its versatility covers a broad range of application needs, from simple widgets like buttons and labels to more complex components like canvas and listboxes. Because of Tkinter's event-driven architecture, developers can easily create applications that are responsive to user interactions and improve the user experience in general. Furthermore, Tkinter's cross-platform compatibility guarantees that programs created with it can operate without difficulty on a variety of operating systems, doing away with the need for platform-specific adjustments and optimizations. Apart from its technical prowess, Tkinter's extensive usage in the Python community highlights its dependability and usability. Whether they are creating enterprise-grade applications or starting their first GUI project, developers of all skill levels will find it to be a great option due to its clear design and thorough documentation. Tkinter's position as the preferred GUI toolkit for Python developers is further cemented by its incorporation into the Python standard library, which guarantees uniform maintenance and support across various Python versions. All things considered, Tkinter's blend of adaptability, dependability, and accessibility renders it an essential instrument for creating robust and intuitive graphical user interfaces in Python.

### Integration with Detection System

The graphical representation clearly illustrates the effectiveness of our pothole detection system, as shown in Figure 3.6. A red bounding box clearly identifies each pothole that has been found, giving users a visual cue as to where it is in the scene that has been captured. In addition to helping to spot potholes, this clear visual aid speeds up decision-making and action. Additionally, each detection's confidence level is clearly shown next to the bounding box, providing users with important context regarding the detection's dependability. This degree of openness enables users to evaluate the predictive accuracy of the system and form well-informed opinions based on the identified confidence intervals.

### Design Layout

The complete integration of our pothole detection system with the GUI interface is shown in Figures 3.5 and 3.6.The GUI interface acts as the main point of contact between users

and the system, offering an easy-to-use platform for smooth control and navigation. The GUI interface's user-friendly design and operation are demonstrated in Figure 3.5, which also includes interactive controls like the start and stop buttons and a live video stream display. These buttons allow users to easily start and stop the video stream, providing flexibility and ease of use when monitoring road conditions.

The system's efficacy is further illustrated by the results of the pothole detection process displayed in the GUI interface of Figure 3.6. A red bounding box is used to visually highlight each pothole that has been found, making it easy to identify them in the photographed scene. Furthermore, each detection's confidence level is clearly shown, allowing users to assess how reliable the system's predictions are. This thorough incorporation of real-time detection results into the GUI interface, combined with the addition of buttons for start and stop, increases user interaction and promotes well-informed decision-making. In summary, Figures 3.5 and 3.6 show how the pothole detection system and the GUI interface work together seamlessly to provide a reliable and user-friendly solution for road maintenance and safety.

## 3.9. Automated Image Saving and Location Tagging using Google Geolocation API

We've made it possible for the system to recognize potholes in live video feeds. When this is detected, the system saves the matching images on its own in a directory called "ResultPrediction." By doing away with the necessity for human intervention, this automated image saving mechanism guarantees that each and every road hazard is quickly recorded for additional study and action.

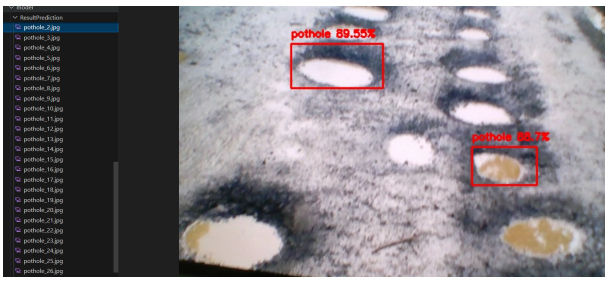



**Figure 3.7:** ResultPrediction File

Our system successfully locates Wi-Fi signals by integrating Google Geolocation services, guaranteeing precise device positioning. This method uses the distinctive IDs of neighboring

Wi-Fi access points to precisely triangulate the device's location. By utilizing the features of the Google Maps API, we are able to gain access to a reliable mapping platform that gives us the exact latitude and longitude coordinates associated with the location of the pothole that has been detected.

```
def get_wifi_location(self):
    # Function to get the MAC addresses of nearby WiFi networks
    def get_wifi_mac_addresses():
        try:
            if 'darwin' in platform.system().lower():
                results = subprocess.check_output(["/System/Library/PrivateFrameworks/Apple80211.framework/Resources/airport", "-s"])
            elif 'win' in platform.system().lower():
                results = subprocess.check_output(["netsh", "wlan", "show", "network"])
            elif 'linux' in platform.system().lower():
                results = subprocess.check_output(["iwlist", "scanning"])
            else:
                raise Exception("Unsupported operating system")

            mac_addresses = []
            lines = results.decode("utf-8").strip().splitlines()[1:]  # Decode bytes to string and skip the header line
            for line in lines:
                fields = line.split()
                mac_addresses.append(fields[0])  # Only store the MAC address
            return mac_addresses
        except Exception as e:
            return []

    wifi_mac_addresses = get_wifi_mac_addresses()
    wifi_data = [{"macAddress": mac} for mac in wifi_mac_addresses]

    google_maps_api_key = "google-api"
    url = f"https://www.googleapis.com/geolocation/v1/geolocate?key={google_maps_api_key}"

    response = requests.post(url, json={"wifiAccessPoints": wifi_data})
    response_data = response.json()

    if 'location' in response_data:
        latitude = response_data['location']['lat']
        longitude = response_data['location']['lng']
        return latitude, longitude
    else:
        return None, None
```

**Figure 3.8:** Code Snippet of retrieving location from WiFi

This technology combination improves our system's accuracy and dependability. We guarantee reliable and thorough location metadata contained in the "ResultLocation" filenames by employing Google Maps and Geolocation data. This metadata provides insights into the distribution of potholes and facilitates targeted repair efforts, making it an invaluable tool for infrastructure management and road maintenance. Overall, our pothole detection system is more useful and effective now that Google Geolocation is integrated for Wi-Fi signal location retrieval, allowing for wise decision-making and effective maintenance plans.

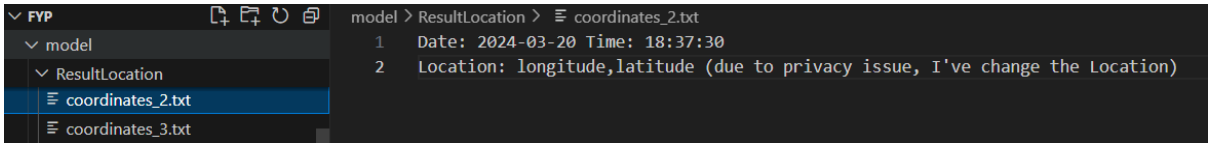



**Figure 3.9:** ResultLocation File

## 3.10. Workflow

### 3.10.1. Pothole Detection System Workflow

The thorough workflow used during the pothole detection model's implementation is depicted in Figure 3.10. The thorough process of creating and assessing the pothole detection model is included in the model workflow, which guarantees accurate and reliable results. The first step is data collection, during which pertinent datasets with pictures of road surfaces are acquired.

These datasets are carefully preprocessed to improve their quality and training suitability. To improve model generalization, data augmentation techniques are used to enrich the dataset with a variety of samples. To enable efficient model training and validation, the augmented dataset is then divided into training and evaluation sets. YOLOv8n-seg is used to train the model on the augmented dataset during the training phase. To reduce prediction errors and maximize performance, the model's parameters are iteratively adjusted during the training process. Concurrently, the assessment set is employed to authenticate the model's functionality, offering discernments into its precision and capacity for generalization. To evaluate a model's efficacy in precisely and consistently identifying potholes, extensive testing and analysis are required. Metrics like precision, recall, and F1-score are among the metrics that are considered in Chapter 4.

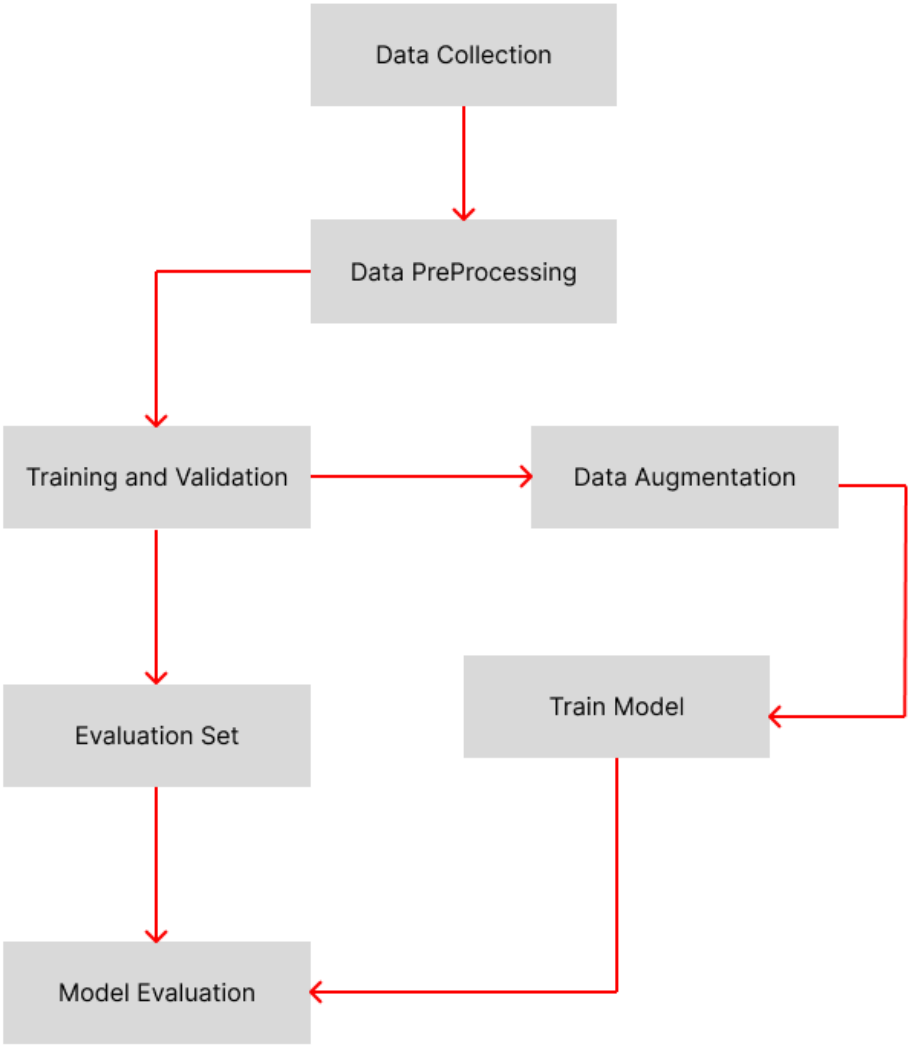



**Figure 3.10:** Workflow of Pothole Detection models

### 3.10.2. GUI Interface Workflow

By streamlining user interactions with the pothole detection system, the GUI workflow increases user control and engagement. The process starts with designing an intuitive graphical user interface and continues until all system functionalities are easily accessible to users. Users can engage directly with the detection system as a result of the pothole detection model's integration with the GUI interface, which makes it a crucial component of the graphical platform. Pothole detection tasks can be monitored and managed more effectively thanks to this integration, which also simplifies the user experience. The GUI interface functions as a dynamic dashboard for real-time pothole detection, giving users access to live video feeds and feedback on potholes that have been detected. With the ability to monitor in real-time, users can watch the detection process as it happens, which helps them make decisions and react quickly to changing road conditions. Users can manage the detection process in accordance with their needs as the workflow advances by utilizing features like start/stop video controls, which ensure flexibility and adaptability in system operation.

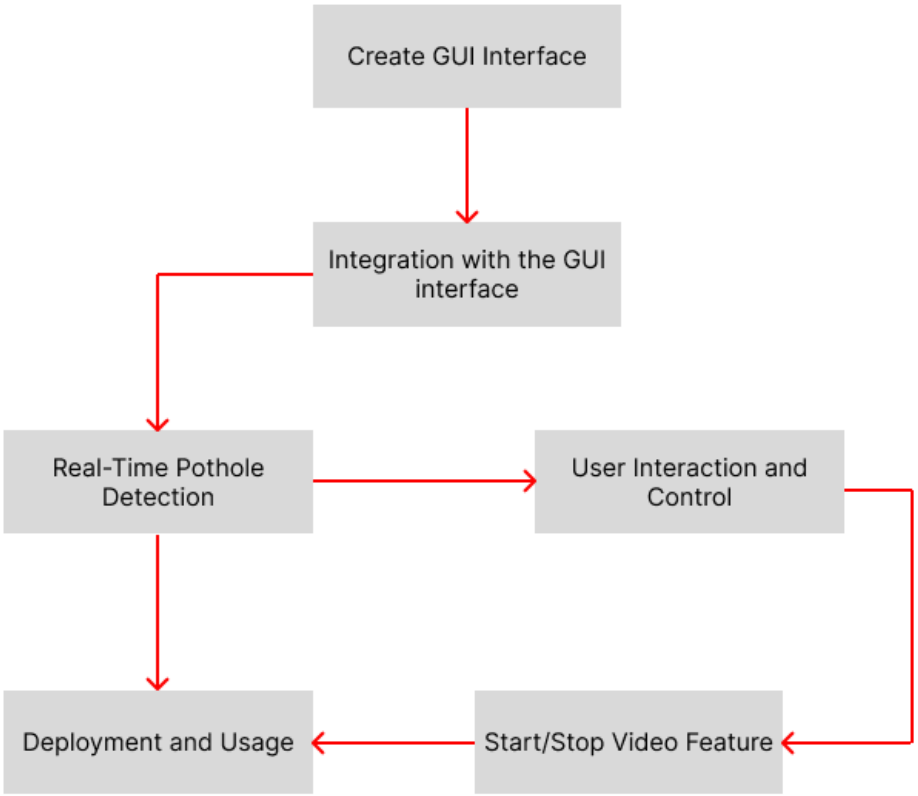



**Figure 3.11:** Workflow of the GUI Interface

# Chapter 4 Evaluation And Result Analysis

This chapter examines our pothole detection system's evaluation and result analysis, which is essential for determining the system's effectiveness and performance. Then we will talk about the experiment and evaluation of the results with pothole detection system. Analysis of the functional and non-funtional requirement analysis concludes the chapter.

## 4.1. Software Evaluation

**Functionality and Features**


**Figure 4.1:** Start Button of the GUI Interface

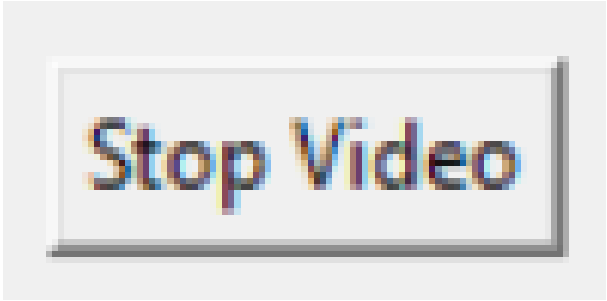



**Figure 4.2:** Stop Button of the GUI Interface

Full screenshot of the start and stop video is shown in Figure 3.5 and Figure 3.6

- Start/Stop Video Button: This button lets users start or stop the live video stream coming from the system-connected camera. It functions as the main control for turning on and off the video stream, giving users the ability to start or stop the pothole detection procedure.
- Result Display: The interface overlays bounding boxes on the video feed to provide visual feedback when it detects potholes in the video stream. Users receive instant visual feed-

back on the system's performance with these bounding boxes, which highlight the location and extent of detected potholes.

- Real-time Feedback: Detection confidence levels and the quantity of potholes found are provided in real-time via the GUI interface..

**Integration with System**

The underlying pothole detection system and the GUI interface integrate seamlessly, allowing for effective control and management of system functionalities.

- Control Interface: The pothole detection system's main control interface is the graphical user interface (GUI), which enables users to efficiently communicate with and oversee system functions. User input within the interface directly affects how the system behaves, for example, by starting or stopping the detection process.
- Feedback Mechanism: By offering feedback on detection outcomes and system performance, the interface enables users to keep an eye on the efficacy of the detection process and make well-informed decisions about how to operate the system. The user is presented with detection outcomes in a timely and accurate manner thanks to integration with the system's feedback mechanism.

### 4.1.1. Back-end Operations and Data Flow

```
0: 256x320 1 Pothole, 98.0ms
Speed: 1.0ms preprocess, 98.0ms inference, 2.0ms postprocess per image at shape (1, 3, 256, 320)

0: 256x320 1 Pothole, 77.5ms
Speed: 1.0ms preprocess, 77.5ms inference, 2.0ms postprocess per image at shape (1, 3, 256, 320)

0: 256x320 1 Pothole, 91.6ms
Speed: 1.0ms preprocess, 91.6ms inference, 3.0ms postprocess per image at shape (1, 3, 256, 320)

0: 256x320 (no detections), 79.0ms
Speed: 1.0ms preprocess, 79.0ms inference, 0.0ms postprocess per image at shape (1, 3, 256, 320)

0: 256x320 (no detections), 79.0ms
Speed: 1.0ms preprocess, 79.0ms inference, 0.0ms postprocess per image at shape (1, 3, 256, 320)

0: 256x320 (no detections), 81.0ms
Speed: 1.0ms preprocess, 81.0ms inference, 0.0ms postprocess per image at shape (1, 3, 256, 320)
```

**Figure 4.3:** The performance in the back-end

#### Performance Metrics

The metrics provided provide a thorough understanding of the pothole detection system's processing speed and effectiveness during inference. Every line denotes a distinct inference operation performed on an input image. It includes information about the image's dimensions, the number of detected potholes , and the duration of each stage of the inference process. The "Speed" section provides a detailed understanding of the computational cost associated with each phase by further breaking down the processing time into preprocessing, inference, and postprocessing stages. Understanding the system's responsiveness, throughput, and resource usage through analysis of these metrics is essential for evaluating the system's performance in various operating environments and real-world applicability.

## 4.2. Model Evaluation

### 4.2.1. Pothole Detection Model Testing, Experiment And Result

Because we have lowered the image resolution to better fit our hardware, the memory usage shown in this section will be less than that shown in other repositories. The results of local testing with lower image resolution are used to calculate average precision (AP) scores.

#### Intersection over Union(IoU)

When evaluating the precision and dependability of object detection models, the Intersection over Union (IoU) metric is an essential tool. IoU measures the degree to which predicted bounding boxes and ground truth annotations overlap, offering valuable information about how well the model locates objects in an image. A successful detection is indicated by classifying the predicted bounding box as a True Positive (TP) when the IoU surpasses a predetermined threshold, usually 0.5 or higher. On the other hand, the predicted bounding box is marked as a False Positive (FP), indicating that the model misidentified it, if the IoU is below the threshold. Furthermore, False Negatives (FN) indicate situations in which the model is unable to identify real objects in the picture, indicating possible areas in which the sensitivity and recall of the model needs to be increased.

**Evaluation Metrics**

Equation is used to calculate for a given set of images :

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Accuracy = \frac{TrueNegative + TruePositives}{TrueNegative + FalsePositives + FalseNegative + TruePositves}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegative}$$

$$F1Score = 2X\frac{RecallXPrecision}{Recall + Precision}$$

- **Precision** measures the proportion of correctly predicted positive instances among all instances predicted as positive, indicating the model's ability to avoid false positives.
- **Accuracy** assesses the overall correctness of the model's predictions by considering both true positive and true negative predictions relative to all instances.
- **Recall** evaluates the model's ability to capture all positive instances by measuring the proportion of correctly predicted positives among all actual positive instances.
- **F1-score** harmonizes precision and recall, providing a balanced assessment of the model's performance by considering both false positives and false negatives.

### 4.2.2. Pothole Detection Model Training Summary

**Fine-Tuning YOLOv8n-seg Training Summary**

| **Metrics** | **Value (YOLOv8n-seg)** |
|---|---|
| Training Time for 80 Epochs | 3.664 Hours |
| AP at the end of 80th Epochs | 60 |
| Model Architecture | 195 layers, 3,258,259 parameters |
| Detection Performance | 60 Images and 201 Instances |
| Box Precision (P) | 75.3% |
| Box Recall (R) | 60.5% |
| Average Precision For Box | 71.8% |
| Mask Precision (P) | 74.2% |
| Mask Recall (R) | 61.7% |
| Average Precision For Mask | 70.8% |
| Speed | 73.1ms inference per image |

**Validation Performance Metric Assessment**

| **Metrics** | **Value (YOLOv8n-seg)** |
|---|---|
| Precision (Box) | 64.4% |
| Recall (Box) | 70.1% |
| Average Precision (Box) | 72.6% |
| Precision (Mask) | 67.4% |
| Recall (Mask) | 65.9% |
| Average Precision (Mask) | 72.2% |
| Fitness | 91.6% |

In validation, the precision metrics pertaining to masks (M) and bounding boxes (B) exhibit high values of 64.4% and 67.4%, respectively, suggesting that the predictions are accurate. Bounding boxes and masks have slightly lower recall scores (70.1% and 65.9%), indicating that some real potholes might go unnoticed. With a 91.6% overall fitness score, the model is well-trained and has room for improvement.

## 4.3. Evaluation Strategy

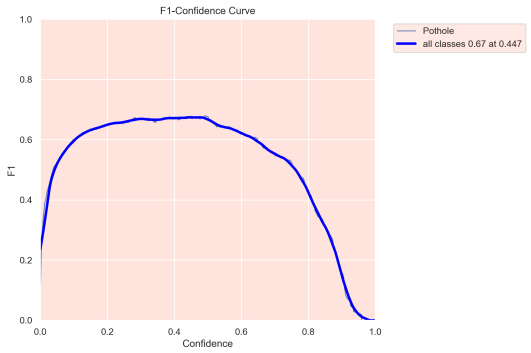
To make sure the pothole detection system is effective in addressing maintenance issues with roads and maintaining public safety, evaluation is essential. This section describes the evaluation method used to gauge the system's performance.

Several metrics are utilized to evaluate the system's performance:

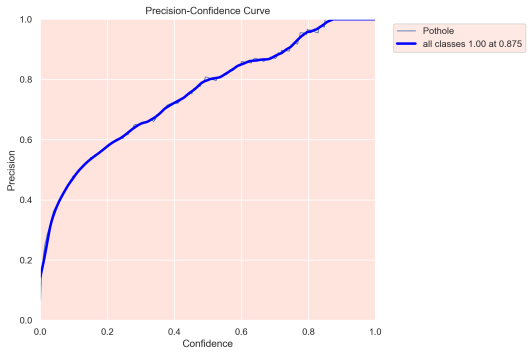
- **Box F1 Score**: Measures the harmonic mean of precision and recall for object detection.
- **Precision**: Indicates the ratio of correctly predicted potholes to the total predicted potholes.
- **Recall**: Represents the ratio of correctly predicted potholes to the total ground truth potholes.
- **Confusion Matrix**: Provides a tabular representation of true positive, true negative, false positive, and false negative predictions

## 4.4. Interpretation of Results

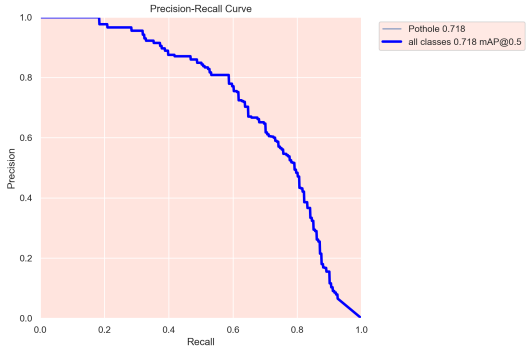
### 4.4.1. Box



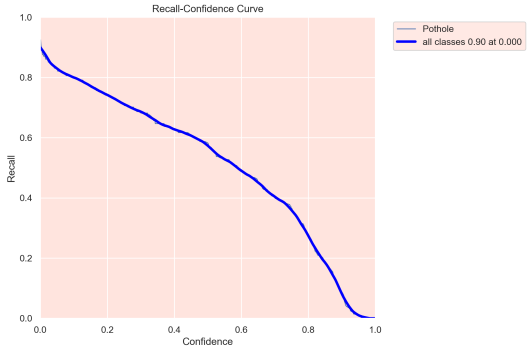

**(a)** Box F1 Score Curve




**(b)** Box Precision Curve

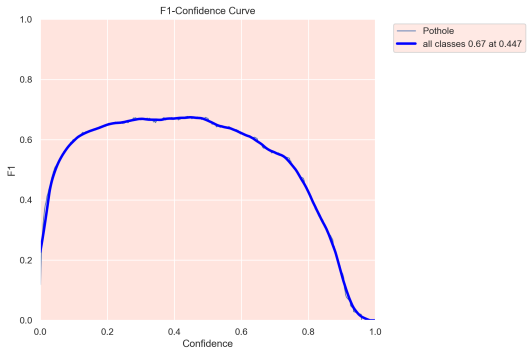



**(a)** Box Precision-Recall Curve

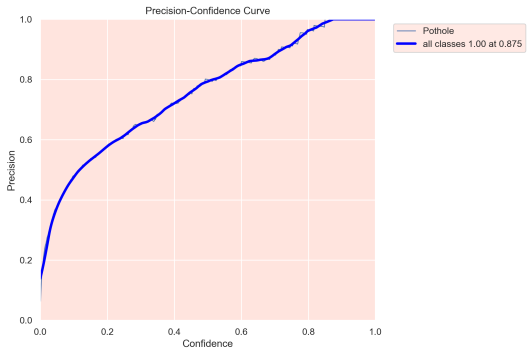



**(b)** Box Recall Curve

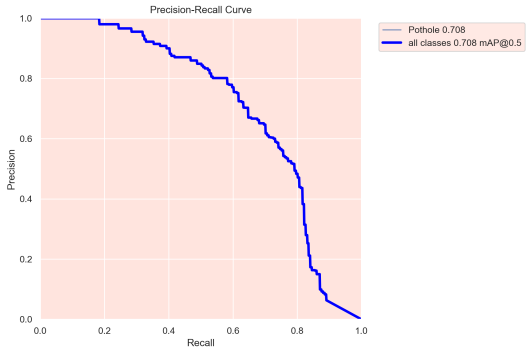
## Mask



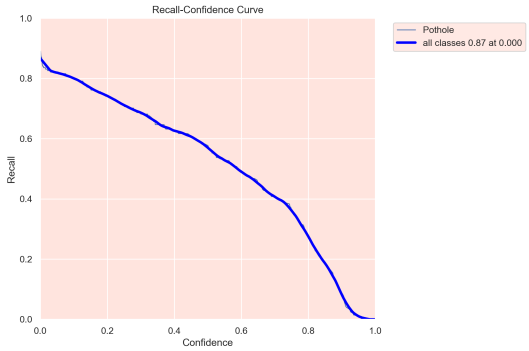

**(a)** Mask F1 Score Curve




**(b)** Mask Precision Curve

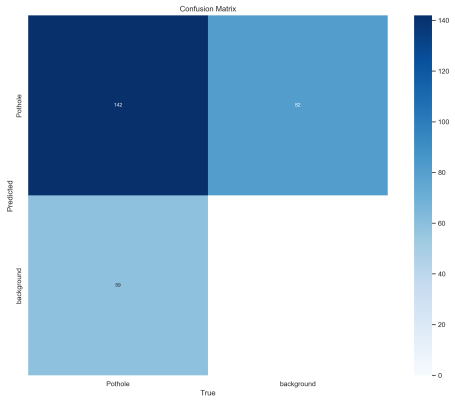



**(c)** Mask Precision-Recall Curve

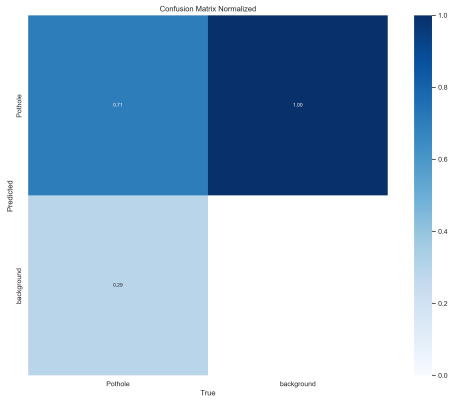



**(d)** Mask Recall Curve

### Confusion Matrix




**(a)** Confusion Matrix




**(b)** Confusion Matrix Normalized

## 4.5. Experiment

Random is provided to the models to analyze in order to find potholes. An image is used to manually test the inference (see Figure 4.8). The code is given in Figure 4.9


**Figure 4.8:** Result of the testing

```python
import random

# Define the path to the validation images
valid_images_path = os.path.join(dataset_path, 'valid', 'images')

# List all jpg images in the directory
image_files = [file for file in os.listdir(valid_images_path) if file.endswith('.jpg')]

# Select 16 random images
selected_images = random.sample(image_files, 16)

# Initialize the subplot
fig, axes = plt.subplots(4, 4, figsize=(20, 21))
fig.suptitle('Validation Set Inferences', fontsize=24)

# Perform inference on each selected image and display it
for i, ax in enumerate(axes.flatten()):
    if i < len(selected_images):
        image_path = os.path.join(valid_images_path, selected_images[i])
        results = best_model.predict(source=image_path, imgsz=640)
        annotated_image = results[0].plot()
        annotated_image_rgb = cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB)
        ax.imshow(annotated_image_rgb)
        ax.axis('off')
    else:
        ax.axis('off')

plt.tight_layout()
plt.show()
```

**Figure 4.9:** Random choose 9 images For Experiment

### 4.5.1. Experiment on live camera

Now to test it on live camera , due to the hardware limitation to perform a pothole experiment, external device is required to test. We use iPad to to display out an image of the pothole and test it. Below is one of the 3 result images 4.10. Although the live camera could not fully detect all the pothole in the external device, it is enough for live camera to detect one of the pothole and together, the location of the pothole will be stored.

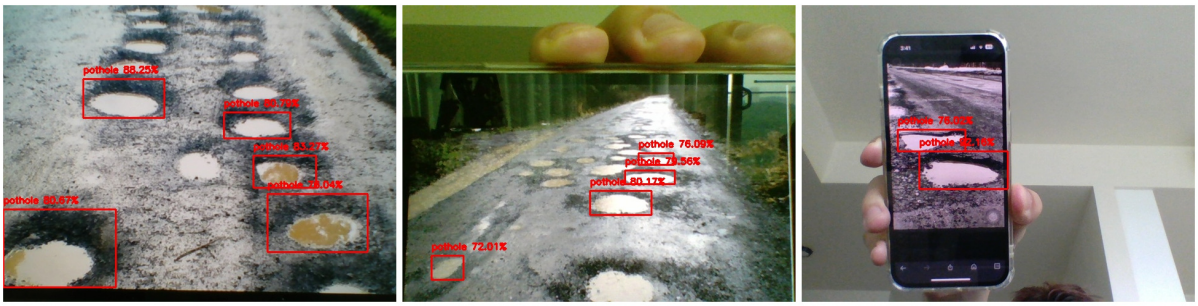



**Figure 4.10:** Live Detection using external device to show pothole

### 4.5.2. Experiment on the output of results

As mention in the previous Section 3.8.1, the output of the resuts will be automatically saved to the file "ResultPrediction"(See Figure 3.7) with the red bounding box. Additionally, date and time and the location of the results will also be stored in the file "ResultLocation" (See Figure 3.9).

## 4.6. Requirements Analysis and Validation

The functional requirements (indicated in table 4.7) and non-functional requirements (indicated in table 4.9) that were included in the project's original plan will be verified in this section. Table 4.6's legend indicates whether or not the requirements have been fulfilled.

**MoSCoW** is the prioritization technique used:

1. M: Must Have
2. S: Should Have
3. C: Could Have
4. W: Will Have

| **Status** | **Color** |
|---|---|
| Completed | |
| Not Completed | |

**Table 4.1:** Legend for functional and non-functional requirements

**Functional Requirements**

| Description | Priority |
|---|---|
| Real-Time Detection: The system must detect potholes in real-time using the YOLO object detection model | Must |
| Save Images: Detected pothole images must be saved to the designated directory for further analysis and record-keeping | Must |
| Record Metadata: The system could record metadata such as the date, time, and location of each detected pothole image. | Could |
| User Interface: A user-friendly interface could be implemented to allow users to interact with the system easily. | Could |
| Generate Alerts: The system will generate alerts or notifications when a pothole is detected to inform relevant stakeholders. | Could |
| Provide Reporting: Reporting functionality will be implemented to generate comprehensive reports on pothole detection activities, including statistics and trends. | Will |
| Continuous Monitoring: The system must continuously monitor the road environment for potential potholes, ensuring timely detection and intervention. | Could |
| Integration with Geographic Information System (GIS): The system must integrate with GIS platforms to overlay detected potholes on digital maps for visual analysis and mapping. | Could |

**Table 4.2:** Functional Requirements

**Non-Functional Requirements**

| Description | Priority |
|---|---|
| Reliability: The system must be reliable, ensuring accurate detection and minimal false positives/negatives. | Must |
| Performance: The system must perform efficiently, with minimal processing time for each detection operation. | Must |
| Scalability: The system could be designed to scale easily to handle increased data volume and user load in the future. | Could |
| Security: Security measures could be implemented to protect sensitive data, such as encryption for stored images and metadata | Could |
| Usability: The system will be designed with usability in mind, providing a straightforward and intuitive user experience. | Will |
| Compatibility: The system will be compatible with various operating systems and devices to ensure widespread accessibility and usability. | Will |
| The system must achieve high accuracy (Average 0.9% of confidence rate) in pothole detection to minimize false alarms and ensure reliable performance. | Must |
| Data Privacy: The system must adhere to strict data privacy regulations and standards to protect sensitive information collected during pothole detection. | Must |

**Table 4.3:** Non-Functional Requirements

# Chapter 5 Conclusion and Future Work

## 5.1. Challenges Faced During Project Implementation

The implementation phase of the project took a long journey to finish due to the fact that we had faced a lot of challenges mainly to due to incompatible and dedicated device. The project need a better GPU and processor to run the project without crashing the application due to unable to respond to the window.

## 5.2. Limitations

As we mentioned the limitation of the project above 5. Limited processor resources does affect the progress of achieving a complete work.

- **Insufficient hardware capabilities:** Even though the AMD Ryzen 7 3750H (which is the device that is use to train model) is a capable processor, it might have trouble with real-time object detection tasks, particularly if the GPU is not being used to its full potential or if the algorithm is computationally demanding. Deep learning tasks may not be well suited for the Radeon Vega Mobile Graphics, despite its suitability for graphics processing.
- **Limited Training Data:** The quantity and quality of training data that is available has a significant impact on how well machine learning-based detection models perform. Models that are less successful at accurately identifying potholes may be the consequence of insufficient or biased training data, particularly in diverse or underrepresented environments.

## 5.3. Future Work

**Multi-Modal Data Fusion**

- Enhanced Detection Accuracy: By cross-referencing visual detections with vehicle motion data, integrating data from multiple sensors—including accelerometers and cameras—improves the accuracy of pothole detection.
- Robustness to Environmental Factors: The system's dependability in a variety of operating environments is ensured by the use of multiple modalities, which help reduce the impact of environmental factors like dim lighting or occlusions.

**Collaboration with Municipalities and Transportation Agencies**

- Data Integration and Exchange: Working with local government agencies entails exchanging pothole data that the system has gathered. This allows for improved resource allocation and coordination for road maintenance operations.
- Policy and Planning Support: Having close ties to transportation agencies and municipalities gives planners and policymakers important information that helps them create efficient plans for infrastructure investment and road maintenance.

## 5.4. Contribution

We are starting a ground-breaking project to transform pothole detection with the help of state-of-the-art machine learning methods and a user-friendly Tkinter GUI. Potholes continue to be a common problem on roads, endangering drivers as well as the structural integrity of the system. By utilizing machine learning algorithms, especially YOLO (You Only Look Once)2.3.7, we hope to effectively address this challenge head-on with our project by quickly and precisely identifying potholes in real-time from live video streams. In addition to increasing pothole detection efficiency, automating the process allows for quicker responses to reduce potential hazards. Furthermore, a key component of our project's accessibility and usability is the creation of a user-friendly GUI interface using Tkinter. Our user-friendly interface provides a smooth interface for users to engage with the system, view identified potholes, and quickly retrieve relevant data. By streamlining the monitoring and maintenance procedure, this interface gives

transportation agencies and municipalities vital information that helps them efficiently allocate resources and prioritize repairs. Through the integration of cutting-edge machine learning technology and intuitive design, we are laying the foundation for an approach to pothole detection and infrastructure management that is both more proactive and efficient. Our project is a major step forward in resolving the persistent problems related to potholes. Our novel strategy has potential for wider developments in road safety and infrastructure upkeep in addition to its immediate uses. We're improving pothole detection efficiency and accuracy by utilizing machine learning and user-friendly interfaces. We're also giving stakeholders the knowledge and resources they need to build more resilient and safe road networks. Together, we're bringing about positive change and influencing how infrastructure management and pothole detection will develop in the future

## 5.5. Reflection And Conclusion

During the process of developing our pothole detection system, we faced many obstacles and development opportunities. Improving the system's performance to guarantee real-time detection while preserving accuracy was one of the most significant challenges. It turned out to be a delicate but rewarding task to balance computational resources and algorithm efficiency, requiring us to continuously streamline procedures and fine-tune parameters. A further degree of complexity was introduced by integrating the Google GeoLocation API for location retrieval, which called for extensive testing and validation to guarantee accurate geolocation data. To sum up, this dissertation marks a critical turning point in the effort to resolve the enduring issue of potholes on roads. The project creates a pothole detection model that is easy to use and accessible, which opens up new possibilities for managing road infrastructure and increasing road safety. It is hoped that as time goes on, this model will be extensively used and adopted, resulting in noticeable improvements to the state of the roads and eventually making travel safer and more enjoyable for all users. This report lays the foundation for a more proactive and successful pothole detection strategy that can be easily scaled up to have a significant impact on communities across the globe.

# References


[1] I. Sutskever A. Krizhevsky and G. E. Hinton. "imagenet classification with deep convolutional neural networks.

[2] R. Zemel A. Rosenfeld and J. K. Tsotsos. The elephant in the room. *Nursing, vol. 50, pp. 42–46*, 8 2018.

[3] A. Gupta A. Shrivastava and R. Girshick. Training region-based object detectors with online hard example mining. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-December, pp. 761–769*, 4 2016.

[4] A. B. Moreno ´ Angel D. Sappa Angel Morera, ´ Angel S´anchez and J. F. ´ V´elez. "ssd vs. yolo for detection of outdoor urban advertising panels under multiple variabilities. *Sensors (Basel, Switzerland), vol. 20, pp. 1–23*, 2020.

[5] KC S. B, M. P. Enhanced pothole detection system using yolox algorithm. 2022.

[6] P. Baheti. A newbie-friendly guide to transfer learning. 4 2023.

[7] P. Arbel´aez C. Gu, J. J. Lim and J. Malik. Recognition using regions. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 1030–1037*, 2009.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. I, pp. 886–893*, 2005.

[9] Eco-Grit. Dangers of potholes. *How Potholes Effect Cars  Cyclists*, 2022.

[10] M. W. Moskewicz K. Ashraf W. J. Dally F. N. Iandola, S. Han and K. Keutzer. "squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. 2 2016.

[11] L. V. D. Maaten G. Huang, Z. Liu and K. Q. Weinberger. Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017-January, pp. 2261–2269*, 8 2016.

[12] Byeong ho Kang and Su il Choi. Pothole detection systemusing 2d lidar and camera. *International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017.

[13] Nhat-Duc Hoang. An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction. *Advances in Civil Engineering, vol. 2018, 2018, pp. 1- 12*, 2018.

[14] J. Li S. U. Akram I. Hasan, S. Liao and L. Shao. Generalizable pedestrian detection: The elephant in the room. 3 2020.

[15] R. Girshick J. Redmon, S. Divvala and A. Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-December, pp. 779–788*, 2015.

[16] S. Ren K. He, X. Zhang and J. Sun. "deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-December, pp. 770–778,*, 12 2015.

[17] T. Korkut. Excelling in software development with scrum methodology part 2. medium. *https://blog.stackademic.com/excelling-in-software-development-with-scrum-methodology-part-2-e2d0b29437ce*, 9 2023.

[18] Steven M. Lee. A pothole can damage your vehicle, your passengers, and you. n.d.

[19] Lalit Kumar Das Lokeshwor Huidrom and S.K. Sud. Method for automated assessment of potholes, cracks and patchesfrom road surface video clips. *Conference of Transportation Research Group of India (CTRG), 2013, pp. 312-321*, 2013.

[20] Mathrubhumi. 1481 deaths, 3103 injured: Pothole-related accidents on the rise, says report. 2023.

[21] H. R. Shenai R. Karani P. A. Chitale, K. Y. Kekre and J. P. Gala. Pothole detection and dimension estimation system using deep learning (yolo) and image processing. *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ), Wellington, New Zealand, 2020, pp. 1-6*, 2020.

[22] M. J. Jones P. Viola and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision, vol. 63, pp. 153–161*, 2005.

[23] T. Darrell R. Girshick, J. Donahue and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 580–587*, 2013.

[24] RangeKing. Phttps://github.com/ultralytics/ultralytics/issues/189.

[25] R. Girshick S. Ren, K. He and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, pp. 1137–1149*, 2015.

[26] J. Y. Lee S. Woo, J. Park and I. S. Kweon. Cbam: Convolutional block attention module. *" Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11211 LNCS, pp. 3–19*, 2018.

[27] I.; Syafi'i. Setyawan, A.; Kusdiantoro. *The effect of pavement condition on vehicle speeds and motor vehicles emissions.* Procedia Eng. 125, 424–430, 2015.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2014.

[29] Toronto. Caa study finds poor road conditions cost canadian drivers 3*billionannually*.2021.

[30] D. Erhan C. Szegedy S. Reed C.-Y. Fu W. Liu, D. Anguelov and A. C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9905 LNCS, pp. 21–37*, 2015.

[31] X. Wang X. Zhu, S. Lyu and Q. Zhao. Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios. 2021.

[32] J. Donahue S. Karayev J. Long R. Girshick S. Guadarrama Y. Jia, E. Shelhamer and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. 2014.

[33] Y. Yuan S. Wang T. Baker Y. Yuan, M. S. Islam and L. M. Kolbe. Ecrd: Edge-cloud computing framework for smart road damage detection and warning. *IEEE Intemet of Things Journal, vol. 8, no. 16, pp. 12734-12747*, 2021.

[34] H. Zhang Z. Zhang J. Xie Z. Zhang, T. He and M. Li. Bag of freebies for training object detection neural networks. 2 2019.

# Appendix A ProjectManagement

The project development methodology, risk assessment, and social, legal, ethical, and professional issues are the main topics of this section of the report.

## A.1. Project Development Methodology

This project uses an Agile-based Scrum methodology to plan and execute the necessary tasks in a step-by-step manner. This is done in order to complete the project quickly and effectively and to prevent missing deadlines.

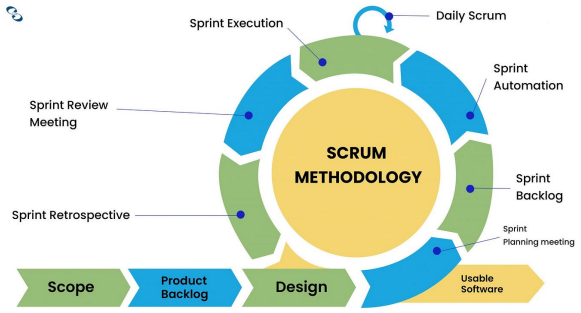



**Figure A.1:** Scrum Methodology [17]

Unlike traditional waterfall approaches, SCRUM is an incremental and iterative development methodology that recognizes the possibility of changing project requirements. We review progress and implement ongoing improvements at regular weekly meetings to maximize flexibility and reduce errors. The SCRUM Master, who acts as the project supervisor, and the individual team members—often referred to as students in educational settings—are two essential roles within the SCRUM framework.

The project has been broken up into smaller tasks known as sprints. These time blocks, known as sprints, are brief intervals of time dedicated to finishing specific tasks. It also provides a chance to evaluate and think back on the work accomplished, identify areas for development, and explore fresh concepts.

## A.2. Project Plan

A Gantt chart outlining the project plan and key deadlines is shown in the following figure.

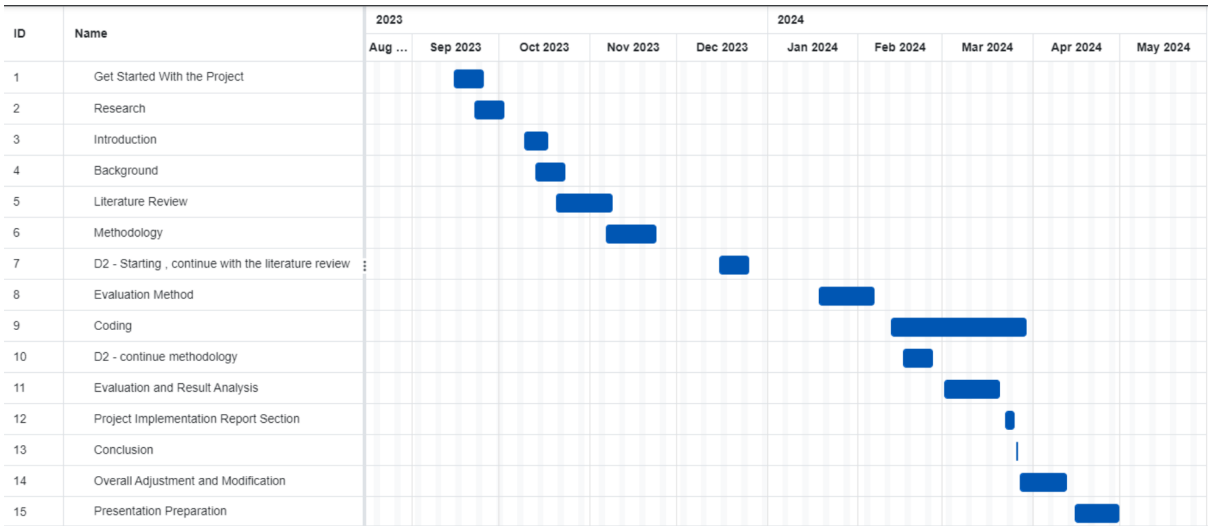



**Figure A.2:** Gnatt Chart

## A.3. Risk Assessment

This section focuses on identifying possible risks, analyzing them, and then determining the best countermeasures and risk monitoring techniques. It will help to have a plan of action in case there are risks during development.

### A.3.1. Risk Type

1. Technology: Any software, hardware, or even technical documents used in this project could pose a risk.
2. Estimation: These take place when planning for project delivery and completion. It focuses on project elements that are related to time.
3. People: Hazards directly associated with the individuals working on the project's completion.
4. Requirements: Hazards resulting from requirements that are altered or modified following the preliminary requirement analysis.
5. General: These are risks that are either general or external and that are uncontrollable

when a project is being implemented.

### A.3.2. Probabilities of Risk

Three different risk probability levels are possible. The color scheme for each of the three levels is shown in Table A.1:

| **Status** | **Color** |
|---|---|
| Low | |
| Moderate | |
| High | |

**Table A.1:** Legend for Risk Probability

### A.3.3. Risk Assessment

| Risk | Risk Type | Description | Impact | Mitigation Strategy |
|---|---|---|---|---|
| Technical issues | Technology | Software may have difficulty detecting potholes | High | Continuous testing and refinement of algorithms |
| Hardware failure | Technology | Camera or related hardware malfunctions | Moderate | Regular maintenance and backup systems |
| Technical issues | Technology | Software may have difficulty detecting potholes | High | Continuous testing and refinement of algorithms |
| Inaccurate estimation | Estimation | Incorrectly identifying non-pothole objects as potholes | High | Implementing machine learning models for improved accuracy |
| Insufficient data quality | Estimation | Poor image quality or environmental conditions | Moderate | Use high-resolution cameras, filtering algorithms |
| Lack of user adoption | People | Users may not utilize the software effectively | Moderate | User training, intuitive interface design |
| Resistance to change | People | Resistance from supervisor | Moderate | Effective communication, demonstrating benefits |
| Misunderstood requirements | Requirement | Misinterpretation of what constitutes a pothole | High | Detailed requirements gathering, supervisor involvement |
| Legal and privacy concerns | General | Privacy violations or legal issues related to recording video | High | Compliance with privacy laws, informed consent |

**Table A.2:** Risk Assessment

## A.4. Professional, Legal, Ethical and Social Issues

### A.4.1. Professional and Legal Issues

This project meticulously follows standard citation formats to cite all research papers and articles consulted in accordance with academic integrity and ethical standards. Additionally, all code that comes from outside authors or repositories is credited in the project documentation as well as in the code comments. The implemented system's source code is made available under the GNU General Public License (GPL), which encourages cooperation and adherence to open-source principles. This license allows users to alter and redistribute the software as long as they follow licensing guidelines and give due credit.

### A.4.2. Ethical and Social Issues

There will be no human testing involved in this project, and no private or sensitive information about them will be utilized. The aforementioned dataset does not contain any sensitive information about any of the participants.