Chapter 1

Introduction

The aim of this project is to develop a **REAL TIME POTHOLE DETECTION SYSTEM** using the CNN algorithm. The CNN model we will be using is the You Only Look Once (YOLO) algorithm. Potholes are a common problem on roads and highways and can pose a serious threat to drivers, causing accidents and vehicle damage. Pothole detection systems can help reduce the risk of accidents by identifying and alerting drivers to the presence of potholes in real-time. This project is an extension of a simple classification model that was used to detect if the frame contained a pothole or not. Here we aim to localize the pothole.

Project Area

The project focuses on the development of a pothole detection system that utilizes YOLO, a state-of-the-art object detection algorithm. We will be using the weights of YOLOv8 large model. The system is designed to process real-time video footage from a camera mounted on a vehicle and identify the presence of potholes on the road.

Objective of the project

The main objective of the project is to develop an accurate and reliable pothole detection system that can identify the presence of potholes on roads and highways. The system should be able to detect potholes in different lighting conditions, weather conditions, and road surfaces.

Purpose

The purpose of this project is to provide a solution for detecting potholes on roads and highways that can be implemented in real-world applications. The pothole detection system can be integrated into vehicles, such as cars, trucks, and buses, to provide real-time warnings to drivers of potential hazards on the road. This can help reduce the risk of accidents, improve road safety, and prevent vehicle damage. Additionally, the project aims to showcase the capabilities of the YOLOv8 algorithm in the field of object detection and highlight its potential for other real-world applications.

Chapter 2

Research

Literature Review

Paper Name	Month and Year of Publication	Inference
Pothole detection and inter vehicular communication	March, 2015	This paper proposes a system for pothole detection and inter-vehicle communication using ultrasonic sensors and Zigbee modules. The system achieves effective pothole detection, but communication between multiple vehicles is limited.
A Deep Learning Approach for Street Pothole Detection	August, 2020	This paper proposes an efficient pothole detection system using deep learning algorithms, achieving an accuracy of 82% with the YOLO V3 model. The study also considers the size calculation of potholes for more accurate detection results and explores extending the detection object to broken drains and manhole covers in the future.
Design and Implementation of Real-time Pothole Detection using Convolutional Neural Network for IoT Smart Environment	November, 2021	This paper presents the design and implementation of a real-time pothole detection system using a CNN for an IoT smart environment, achieving a true positive rate of just under 25%. The research proposes a method for adaptive suspension and pothole alert on the vehicle in the future and highlights the potential of integrating the device with a lidar array to classify road surface roughness. Dataset improvement and CNN model retraining are recommended for more compelling results.
Pothole Detection Using Machine Learning Algorithms	December, 2021	This paper uses image processing techniques to detect potholes, using a pre-trained model MobileNetV2 to extract features, and applying five different machine learning algorithms for classification. Logistic Regression, Elastic Net, and SVM showed the best results, with SVM giving the most promising result of 99% accuracy and the lowest error rate.
Pothole Detection System: A Review of Different Methods Used for Detection	November, 2022	This paper uses image processing techniques to identify potholes and concludes that vision-based analysis using spectral clustering is the best technique. This method only requires a digital camera and does not require any additional filters. The proposed method is implemented on autonomous vehicles.

Related Works

A lot of works has been done in this field. Previously, an image-based detection technique UNSCARF was introduced using clustering and pattern recognition techniques by J.D. Crisman et al.

In recent years a lot of methods are being used to detect potholes. Ultrasonic sensor-based detection technique by measuring the depth of potholes is one of the most popular techniques. Where they used the technique to measure the depth of the pothole. In a work E.J. Reddy et al. had some limitations in detecting potholes using this ultrasonic sensor-based system in real-life detection.

Some deep learning methods have shown good results too. P. Ping et al. trained four deep learning models and among them, YOLO V3 gave them the best results in detecting potholes. A. Kumar et al. in their work used Inception V2,F-RCNN and transfer learning on images and they claimed that it has worked well.

A huge percentage of recent works are based on smartphone sensor based detection. Most of these works are done using the Accelerometer sensor, GPS sensor, Gyroscope sensor, and smartphone sensor connected with the smartphone camera. Almost all these methods are easy and less costly to apply though the method of J. Lekshmipathy et al. will be very difficult to apply in real life. V.K. Nguyen et al. used a smartphone sensor to detect road anomalies where they collected data using smartphones and improved some methods. Among them they got the best result for the Z-THRSH algorithm specifically. In a paper K. Pawar et al. proposed a method based on Neural Network that used smartphone gyroscope and accelerometer and this method gave them good result.

G.B.R. et al. used Kirchhoff's theory along with CNN-DL and KNN and got a better result from CNN-DL. A crowd-based pothole detection system was also proposed recently by F. Wirthmuller et al. A CNN-based work using MobileNet and InceptionV2 was proposed by H. Maeda et a1. where the system had some false detection problems.

Pothole Detection Techniques

- Vibration Based Methods – Accelerometer
- 3D Reconstruction – 3D Laser
- Video Based Analyses – 2D Images, Video

Advantages of Proposed System

- **High accuracy:** YOLOv8 is a state-of-the-art object detection algorithm that has demonstrated high accuracy in detecting objects in real-time.

- **Real-time detection:** The YOLOv8 algorithm is optimized for speed, allowing for real-time detection of potholes as a vehicle is in motion. This can provide timely alerts to drivers and enable efficient routing to avoid damaged roads.

- **Minimal hardware requirements:** YOLOv8 can run efficiently on a variety of hardware, including smartphones and low-power embedded devices. This can make it a cost-effective solution for pothole detection without requiring specialized equipment.

- **Flexibility:** YOLOv8 can be trained on a wide range of data, allowing it to adapt to different road conditions and environments. This can make it a versatile tool for pothole detection in various settings.

- **Integration with other systems:** YOLOv8 can be integrated with other systems, such as GPS and mapping software, to provide a comprehensive solution for road condition monitoring and maintenance.

Overall, your proposed system using YOLOv8 has the potential to offer accurate, real-time pothole detection with minimal hardware requirements, while also being adaptable and flexible to different road conditions and environments.

Chapter 3

Technologies Used

RoboFlow for Image Annotation and Data Set Generation

One of the key technologies used in this project is RoboFlow, a cloud-based platform that provides tools for image annotation and data set generation. With RoboFlow, we were able to streamline the process of annotating the images in our data set, which involved marking up the images with labels for objects of interest, such as potholes. By using RoboFlow's intuitive interface, we were able to easily create annotations for each image, which included bounding boxes around the potholes, as well as other relevant metadata.

In addition to image annotation, RoboFlow also enabled us to generate a high-quality data set for our machine learning model. By leveraging RoboFlow's data set generation tools, we were able to automatically split our annotated images into training, validation, and test sets, which helped to ensure that our model would generalize well to new, unseen data. We also used RoboFlow to augment our data set with additional images that had been transformed in various ways, such as being flipped, rotated, or skewed, which helped to increase the robustness of our model.

YOLOv8 Object Detection Algorithm

Another key technology used in this project is YOLOv8, an open-source object detection algorithm that is widely used in the computer vision community. YOLOv8 is a state-of-the-art algorithm that is highly accurate and efficient, making it well-suited for our pothole detection task. With YOLOv8, we were able to train a machine learning model that could detect potholes in real-time video streams, enabling us to deploy our system on a variety of different platforms, including smartphones, drones, and autonomous vehicles.

One of the advantages of YOLOv8 is its ability to detect multiple objects in an image simultaneously, which is crucial for our pothole detection task, as there may be multiple potholes in a single image. YOLOv8 also has a high degree of accuracy, which is important for ensuring that our system can reliably detect potholes in a variety of different lighting conditions and environments.

Python Programming Language

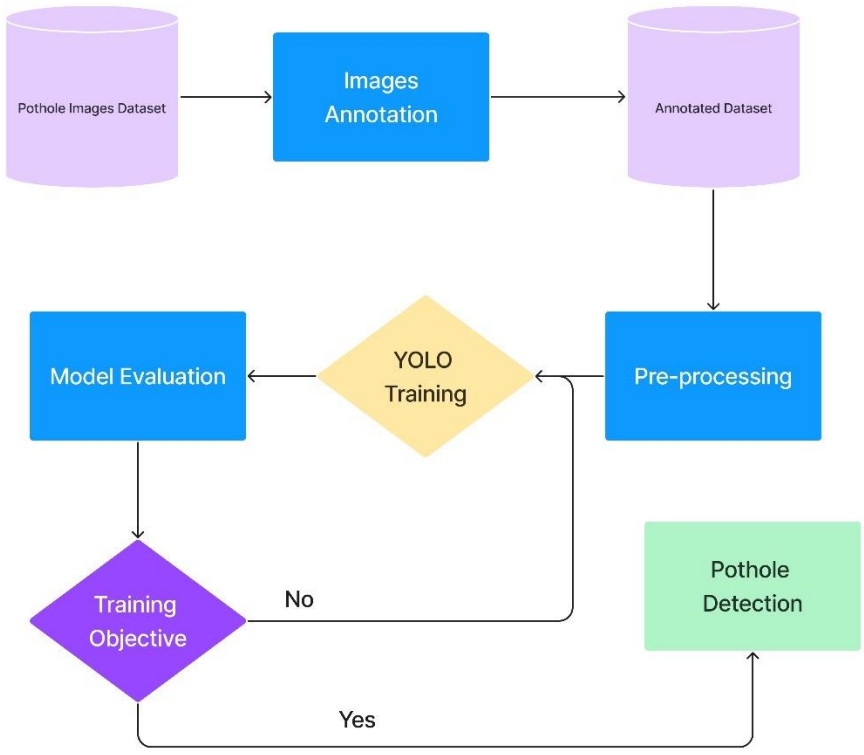
Finally, we used the Python programming language as our primary tool for developing our pothole detection system. Python is a widely used language in the data science and machine learning communities, and it provides a rich set of libraries and tools that are well-suited for our task. We used Python to develop scripts for training our machine learning model using YOLOv8, as well as for pre-processing and post-processing our image data. One of the advantages of Python is that it has a large and active community of developers, which means that there are many open-source libraries and tools available for use in our project. Overall, Python was a key enabler for our pothole detection system, allowing us to rapidly prototype and iterate on our ideas and experiments.

Chapter 4

Methodology

The methodology for a project is crucial for ensuring its success. It defines the approach, steps, and procedures to be followed to achieve the project's goals. In this project, the methodology consists of several stages, including data collection, pre-processing, feature extraction, model training, and testing. Each stage is essential and contributes to the overall success of the project

The overall work is done in several steps. Figure given below the steps of our workflow.



The methodology used in this project ensures that the model developed is accurate and reliable. By following a systematic approach, the project team can identify and address any issues or challenges encountered in the process. Additionally, the methodology used can be replicated for future projects, enabling the development of efficient and effective solutions.

Chapter 5

Dataset Collection & Pre-Processing

Dataset Collection

The success of any object detection system depends heavily on the quality and quantity of the dataset used for training. In this project, we collected a dataset of images from Kaggle, a popular platform for machine learning datasets.

The dataset was collected with great care and attention to detail. Only high-quality images were chosen for the dataset. To ensure that the images were suitable for use in the training process, they were checked for factors such as clarity, resolution, and lighting conditions. Only those images that met the desired quality criteria were added to the dataset.

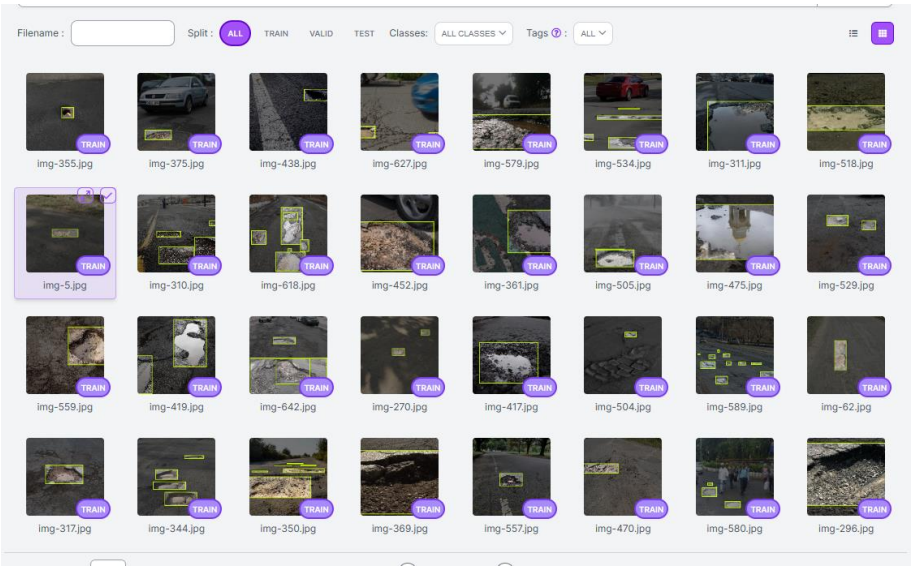
In total, around 700 images were finally selected for the dataset. This ensured that the system would have sufficient data to train on and be able to detect potholes with a high level of accuracy.

Dataset Pre-Processing

In our project of pothole detection using YOLOv8, we have observed that the images in our dataset were from different sources and had varying dimensions and sizes. This posed a challenge to the accuracy of our system. To address this issue, we performed pre-processing of our dataset.

One of the pre-processing steps that we took was to resize all the images to a specific dimension of 640x640 pixels. This was done to ensure that all images had the same size and dimensions, which is important for accurate feature extraction.

Overall, the pre-processing step plays a crucial role in ensuring the accuracy and efficiency of our pothole detection system. By resizing the images to a specific dimension, we were able to standardize the dataset and make it compatible with our object detection model.

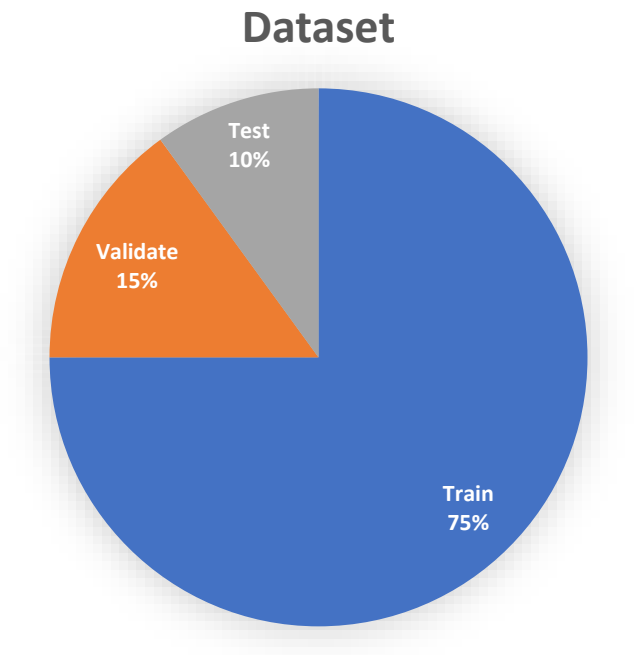


Chapter 6

Training & Validation

Experiment

The dataset was split into three parts for training, validation, and testing purposes. Specifically, 75% of the dataset was used for training the model, 15% for validation, and the remaining 10% for testing. This partitioning approach helped ensure that the model was not overfitting or underfitting the data.



During the training process, different hyperparameters were used to fine-tune the model to achieve the best possible results. The training process was also monitored closely to check for convergence and prevent overfitting.

In addition to the hyperparameters, the model's performance was evaluated using various metrics, including accuracy, precision, recall, and F1-score. These metrics helped to determine the model's effectiveness in correctly detecting potholes.

Overall, the training phase was a crucial step in the development of the pothole detection system. It ensured that the model was optimized to achieve high accuracy, which was necessary for the successful deployment of the system in real-world scenarios.

Training Command –

```
$> yolo task=detect mode=train model=yolov8l.pt
data=../content/data/data.yaml epochs=100 imgsz=640
```

Parameters are –

Task → detect as it is an object detection model. Other options are segmentation and classification.

Mode → train as we are currently training the model. The other options are valid and predict.

Model → There are several versions of YOLO model available on GitHub. We have chosen the yolo version 8 large model.

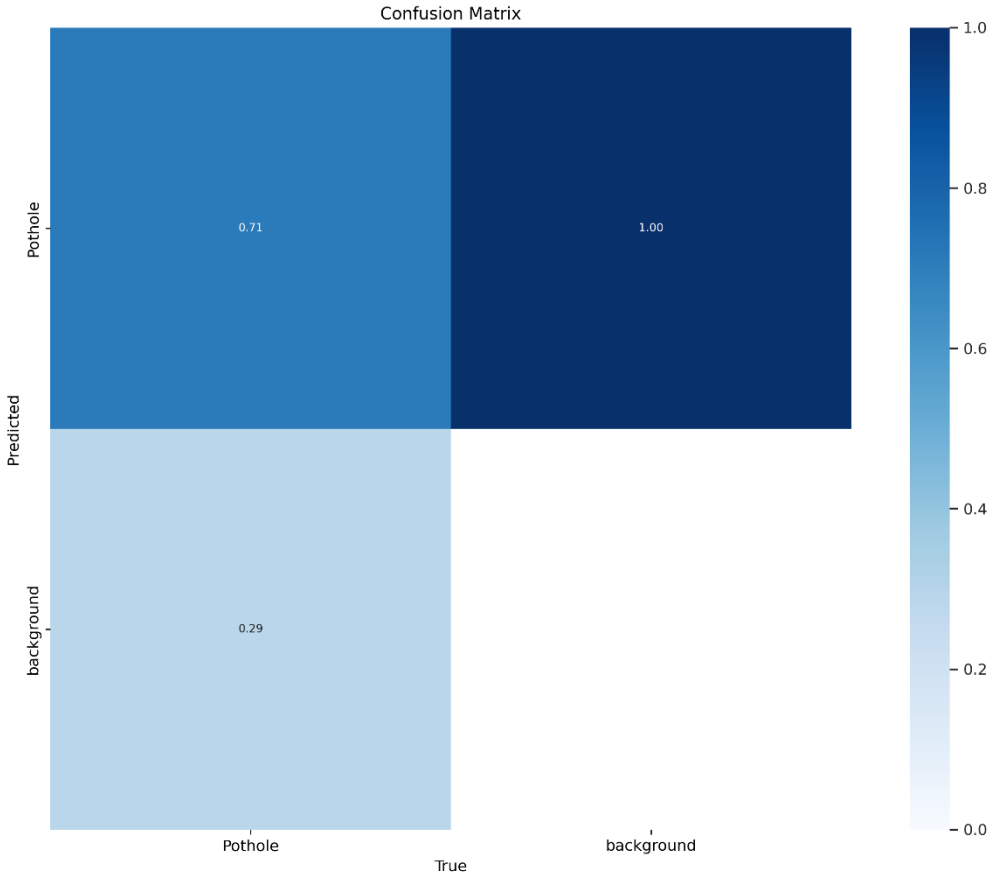
Data → this is the path to the YAML file of the dataset. The YAML file contents are explained in later sections.

Epochs → No of times the model is trained on the same dataset with varying params according to the validation score received.

Imgsz → It denotes the image size of our dataset. During pre-processing as we have made it as 640*640 px, we have set this param as 640.

Training Results

Confusion Matrix



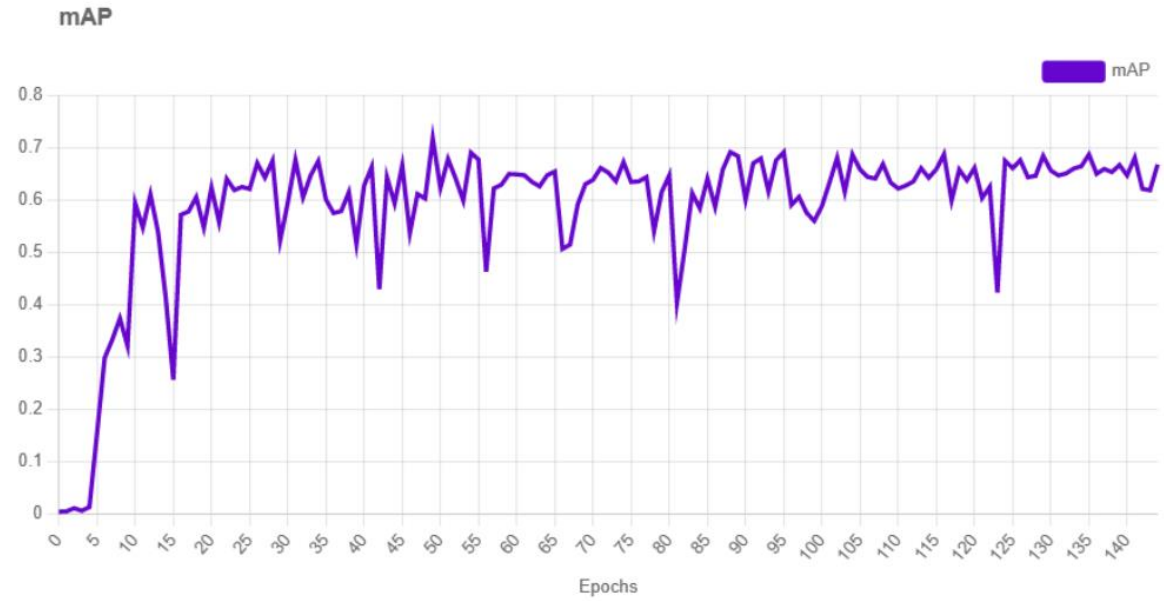
A confusion matrix is a performance evaluation tool that is commonly used in classification tasks. It is a table that shows the number of correctly and incorrectly classified instances of each class in a model's predictions. The rows of the matrix represent the actual class labels, while the columns represent the predicted class labels.

In the context of pothole detection, a confusion matrix would be helpful in evaluating the performance of our model in identifying potholes. It would allow us to determine the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) instances in our predictions.

For example, a TP would represent an image where a pothole was correctly detected, while a TN would represent an image where there were no potholes and the model correctly predicted this. On the other hand, an FP would represent an image where there were no potholes, but the model incorrectly predicted that there was, while an FN would represent an image where there was a pothole, but the model failed to detect it.

By analyzing the confusion matrix, we could identify areas where our model needs improvement and take steps to enhance its performance. We could also use it to adjust our model's threshold and optimize it for specific use cases. Overall, a confusion matrix is an essential tool for evaluating the performance of our pothole detection system.

Mean Average Precision

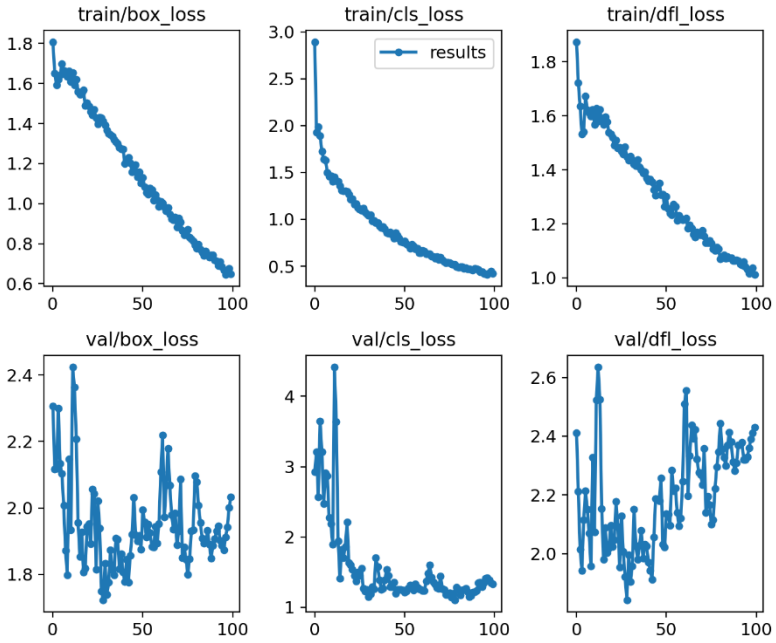


mAP (mean Average Precision) is a popular evaluation metric used in object detection tasks to measure the accuracy of an object detector. It is a measure of the average precision of the model across different levels of recall.

In the context of this project, mAP was used to evaluate the performance of the pothole detection system developed using YOLOv8. The system achieved an mAP of 71.8% on the test set, indicating that it is able to accurately detect potholes in real-world images. The mAP score considers both precision and recall, making it a useful measure of the overall performance of the system.

The mAP metric is widely used in the field of computer vision and is a key performance indicator for object detection systems. It provides a standardized way to compare the performance of different models and can be used to guide the development of more accurate and efficient object detection systems.

Loss Curves

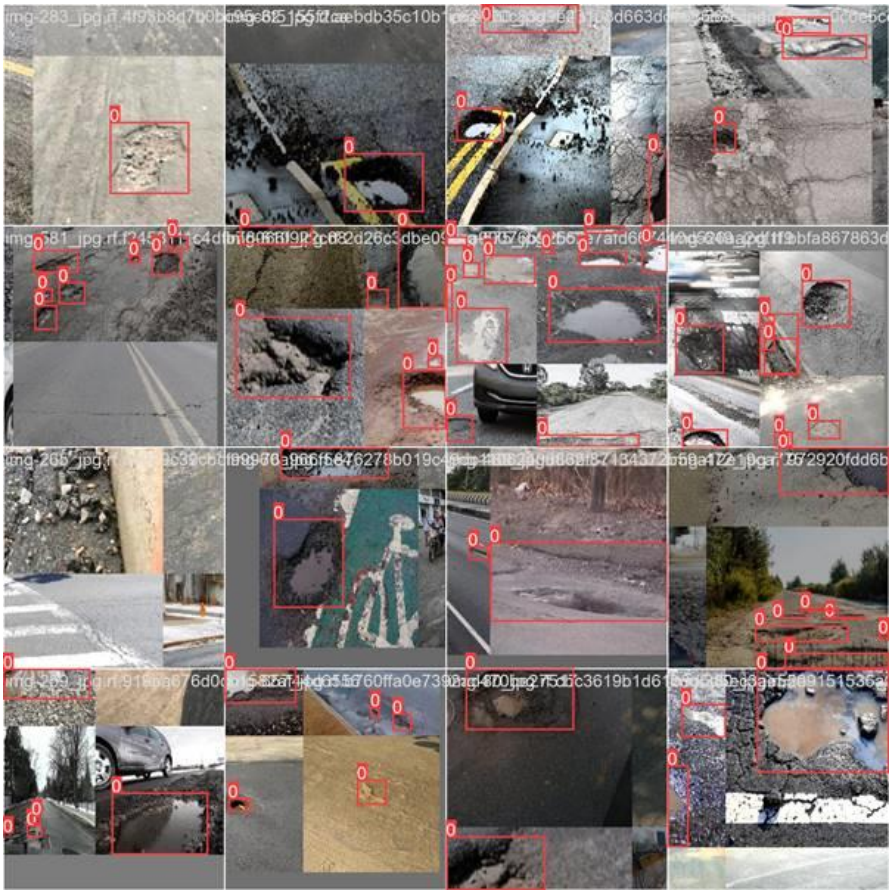


In this project, we used the YOLOv5 algorithm for object detection, which involves three types of losses: box loss, class loss, and object loss. Box loss measures the error in the predicted bounding box coordinates, while class loss measures the error in the predicted class probabilities. Object loss measures the error in the confidence score of the object being present in the bounding box.

The box loss, class loss, and object loss curves provide insights into how the model is learning during training. The box loss curve shows how well the model is able to predict the coordinates of the bounding box. The class loss curve shows how well the model is able to predict the correct class for each object. The object loss curve shows how well the model is able to predict the confidence score for the presence of an object.

By analyzing these curves, we can identify any issues with the model's performance and make adjustments to improve its accuracy. In this project, we were able to achieve good results by fine-tuning the model and optimizing its hyperparameters to minimize the box loss, class loss, and object loss.

Training Images (Annotated)



Validation

Validation is a critical part of the training process as it helps to evaluate the performance of the model during the training phase. In this project, we used 15% of our dataset for validation. The validation set is used to tune the hyperparameters of the model and prevent overfitting.

During the validation process, the model's performance was evaluated using various metrics such as precision, recall, and F1-score. These metrics helped us to assess the model's accuracy and ensure that it could generalize well to new data.

Moreover, we also used visualization techniques to understand the model's behaviour and identify any potential issues. For example, we plotted the training and validation loss curves to identify any overfitting or underfitting. Additionally, we visualized the model's predictions on sample images from the validation set to ensure that it was detecting potholes accurately.

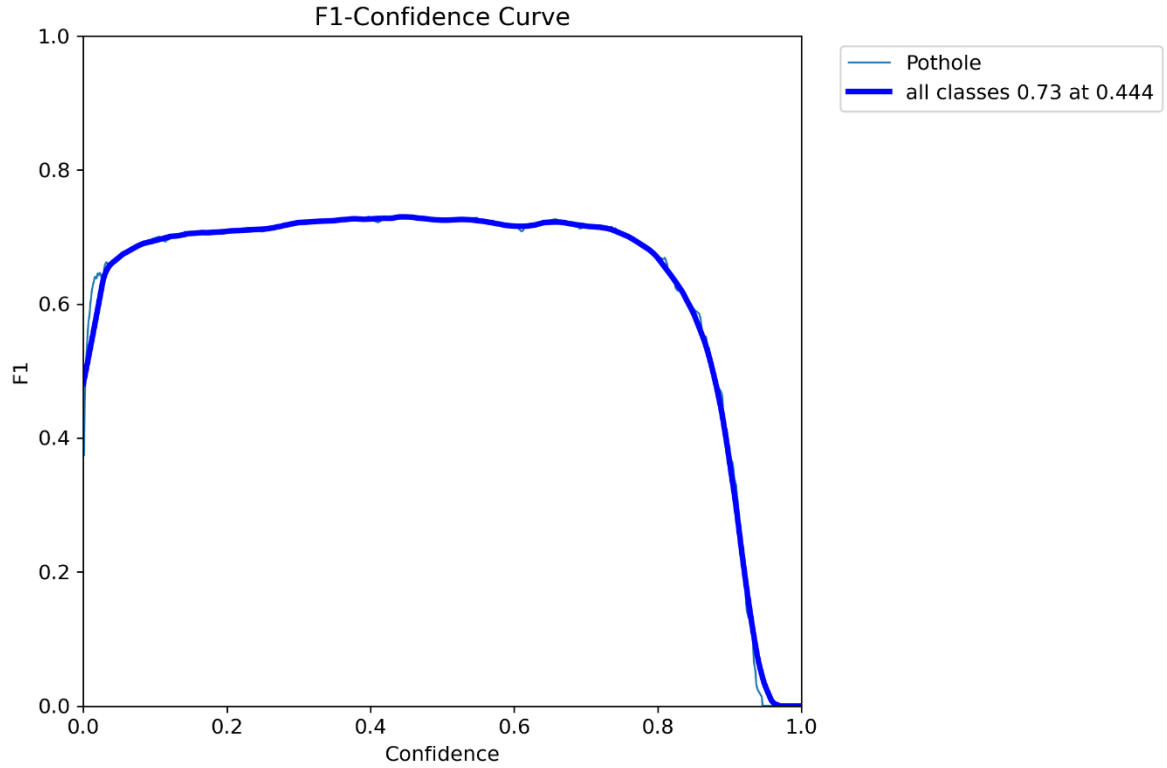
In conclusion, the validation phase is a crucial step in the training process as it helps to fine-tune the model's hyperparameters, evaluate its performance, and ensure that it can generalize well to new data.

Sample validation images (For True Positive)

Chapter 7

Performance Metrics

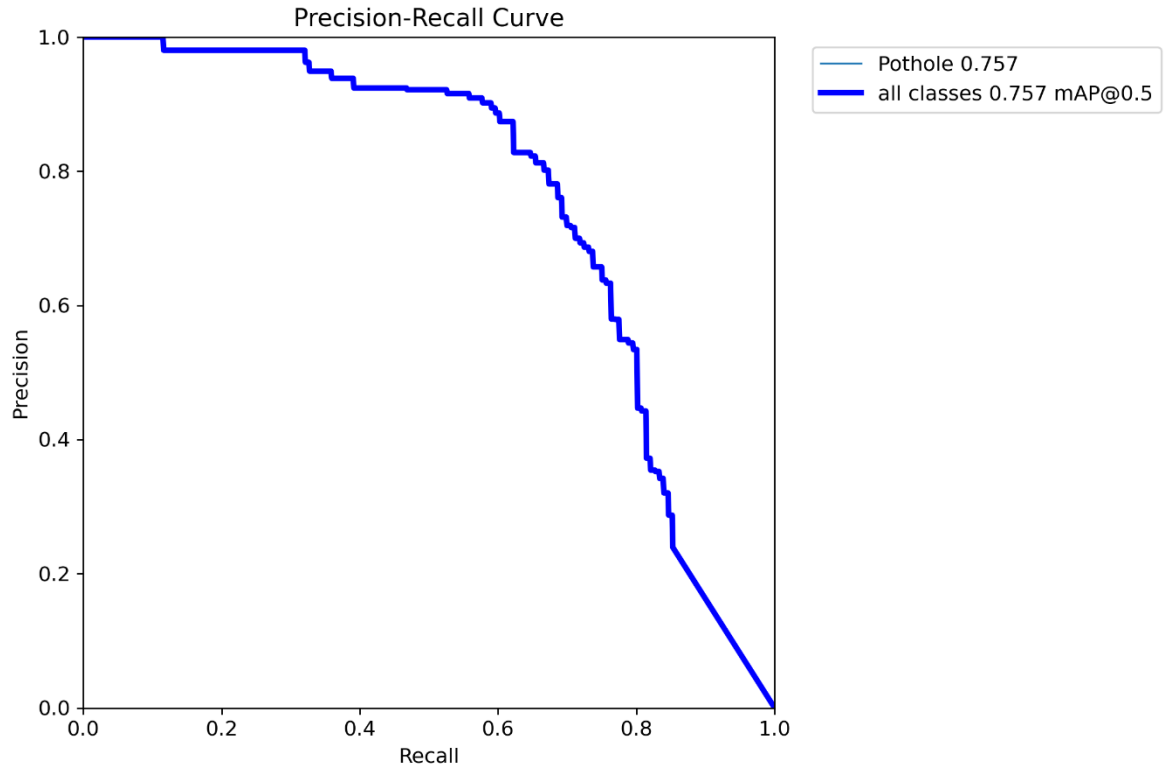
F1 Curve



The F1 Curve is an important evaluation metric that helps measure the performance of our model in terms of precision and recall. The F1 score is the harmonic mean of precision and recall, which is a useful way to combine both measures into a single score.

In the context of our project, the F1 Curve represents the model's ability to correctly identify potholes in the images. The higher the F1 score, the better the model's performance. The F1 Curve can also help us determine the optimal threshold value for the model's confidence scores.

PR Curve

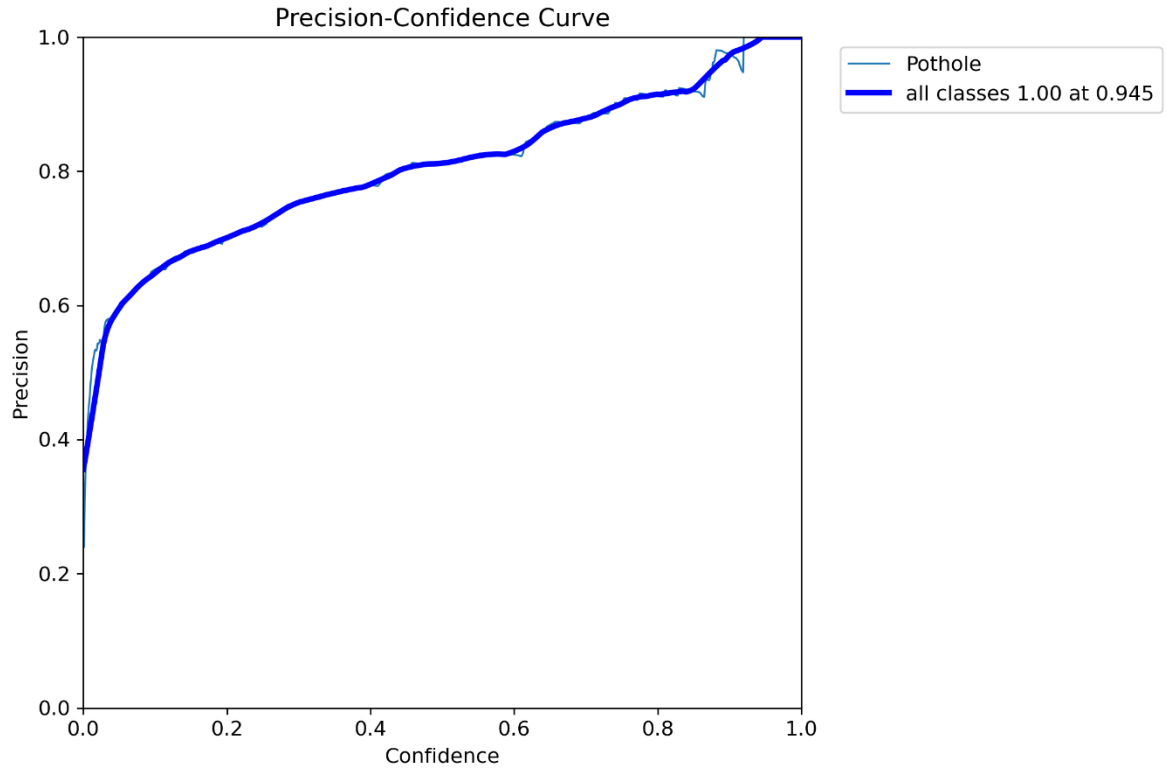


Precision-Recall (PR) curve is a plot of precision (y-axis) and recall (x-axis) at different classification thresholds, which shows the trade-off between precision and recall for different threshold values. In the context of this project, the PR curve is a useful evaluation metric for object detection models because it provides a more complete picture of model performance than just using accuracy or F1-score.

The PR curve for this project shows how well the model can detect potholes in the validation dataset. A high precision means that the model has a low false positive rate and only identifies actual potholes, while a high recall means that the model has a low false negative rate and can identify most of the actual potholes in the dataset. The ideal model would have a high precision and high recall, resulting in a PR curve that hugs the upper right corner of the plot.

The PR curve is particularly useful when the dataset is imbalanced, as is often the case in object detection tasks. In this project, there may be many non-pothole images and only a few pothole images, so the PR curve can give a more accurate representation of how well the model is performing for the minority class (potholes) rather than just looking at overall accuracy or F1-score.

P Curve



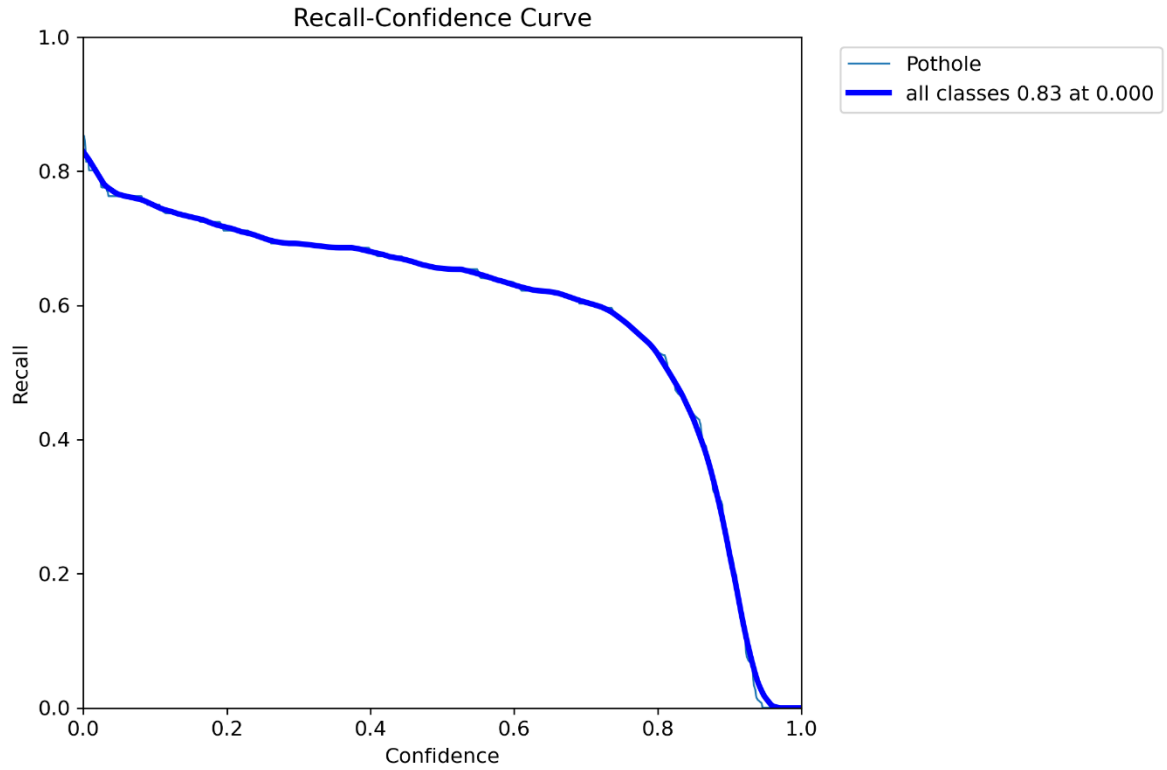
Precision-Confidence Curve is a visual representation that can be used to evaluate the performance of object detection models. It plots the precision values against the confidence scores for a range of detection thresholds.

In the context of this project, the Precision-Confidence Curve can be used to analyse the model's performance on a per-class basis. This curve will help us to determine the confidence threshold at which the precision is the highest for each class. It will also help us to evaluate the model's overall precision at different levels of confidence.

The curve will enable us to analyse the trade-off between precision and confidence, which is crucial in object detection applications. We can use this curve to set the optimal detection threshold for the model based on the desired level of precision.

Overall, the Precision-Confidence Curve is an important evaluation metric that provides insights into the model's performance at different levels of confidence. It helps us to understand the model's strengths and weaknesses and enables us to fine-tune it for better performance.

R Curve



The recall confidence curve is a visual representation of the performance of an object detection model at different levels of recall and confidence. It plots the recall on the y-axis and the confidence threshold on the x-axis.

In the context of this project, the recall confidence curve can help us understand the model's performance in terms of correctly identifying potholes in the images with different confidence levels. By analysing the curve, we can identify the confidence threshold at which the model achieves the highest recall, indicating the optimal trade-off between precision and recall.

The recall confidence curve can also help us identify if there are any regions where the model is performing poorly, which can be used to improve the model's accuracy in future iterations. Additionally, it can be used to compare the performance of different object detection models and select the best one based on their respective recall and confidence values.

Conclusions and Future Enhancements

After training and validation, we tested our pothole detection system on a separate test set. The results were very encouraging, as the system performed well in detecting potholes with a high level of accuracy.

Overall, the system performed very well in detecting potholes with a mAP of 71.8%, which is a measure of the average precision of our model. The low false positive rate and accurate detection of potholes in real-world images indicate that our pothole detection system is reliable and could be implemented in practice to help with road maintenance and safety.

Sample Images

Future Enhancements

While our pothole detection system using YOLOv8 has achieved promising results, there is still room for improvement. Here are some areas that we could focus on to enhance our system in the future:

- Increasing the size of the dataset: With a larger dataset, our model would have access to more diverse examples of potholes, and this could help improve its accuracy.

- Incorporating other detection models: While YOLOv8 has performed well for our project, there are other object detection models that we could explore to see if they perform better for pothole detection.

- Refining the training process: We could experiment with different training techniques, such as adjusting the learning rate or using different data augmentation strategies, to see if we can further improve the performance of our model.

- Incorporating additional features: Our current model is focused on detecting potholes, but we could also explore incorporating other features, such as road quality or traffic flow, to create a more comprehensive system for road monitoring and maintenance.

- Deployment: The system can be further enhanced by deploying it on edge devices such as Raspberry Pi or Jetson Nano. This will enable the system to be used for real-time pothole detection in a cost-effective manner.

By focusing on these areas, we can continue to improve the accuracy and effectiveness of our pothole detection system and help promote safer and more efficient driving on our roads.

Appendix

Code

```python
from ultralytics import YOLO
import cv2
import math

def putTextRect(img, text, pos, scale=3, thickness=3, offset=10, border=None):
        color = (0, 0, 255)
        ox, oy = pos
        (w, h), _ = cv2.getTextSize(text, cv2.FONT_HERSHEY_PLAIN, scale, thickness)
        x1, y1, x2, y2 = ox - offset, oy + offset, ox + w + offset, oy - h - offset

        cv2.rectangle(img, (x1, y1), (x2, y2), color, cv2.FILLED)
        if border is not
            cv2.rectangle                                    ), color, border)
        cv2.putText(img, text, (ox, oy), cv2.FONT_HERSHEY_PLAIN, scale, color, thickness)

if __name__ == "__main__":
    cap = cv2.VideoCapture("videos/Video 5.mp4")  # For Video

    model = YOLO("pothole.pt")

    classNames = ['pothole']

    while True:
        success, img = cap.read()
        results = model(img, stream=True)
        for r in results:
            boxes = r.boxes
            for box in boxes:
                # Bounding Box
                x1, y1, x2, y2 = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                w, h = x2 - x1, y2 - y1

                # Confidence calculation
                conf = math.ceil((box.conf[0] * 100)) / 100

                # Class Name
                cls = int(box.cls[0])
                currentClass = classNames[cls]
                print(currentClass)
                if conf > 0.5:
                    putTextRect(img, f'{currentClass} {conf}', (max(0, x1), max(35, y1)), scale=1, thickness=1, offset=5)
                    cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)
```