POTHOLE DETECTION USING YOLO V8

1. Introduction

1.1 Background

Road potholes are a persistent global infrastructure problem affecting vehicle safety, travel efficiency, and maintenance costs. Traditional detection relies on manual surveys which are time-consuming, expensive and labor-intensive. Modern computer vision techniques can automate and scale the detection process.

Object detection using deep learning has emerged as a powerful solution for real-time pothole detection in images and videos by identifying and localizing potholes as class objects. Among these, the YOLO (You Only Look Once) family of models is widely adopted due to its **single-shot detection paradigm** that enables fast inference.

1.2 Purpose of the Project

This project implements a custom **YOLO-V8 based pothole detection system** using a labeled dataset of road images with potholes annotated in YOLO format. The system is capable of training, validation, and inference on both images and videos.

1.3 Scope

The project focuses on:

- Preparing and training a YOLO-V8 model on a custom pothole dataset.
- Evaluating the model performance using standard metrics.
- Performing real-world object detection via images and videos.
- Documenting the architecture, workflow, challenges, and outcomes.

2. Objectives

2.1 Primary Objectives

- To design and train an efficient deep learning model for pothole detection on road surfaces.
- To implement a complete pipeline from data preprocessing to post-training inference.

2.2 Technical Objectives

- Collect and label a dataset suitable for training object detection models.
- Configure and train YOLO-V8 architecture.
- Evaluate model accuracy, precision, and generalization.
- Provide inference scripts for practical detection tasks.

2.3 Non-technical Objectives

- Document the entire workflow for reproducibility and future improvements.
- Present results suitable for deployment or further optimization.

3.Architecture and Working

3.1 Overview of YOLO Architecture

YOLO (You Only Look Once) is a **single-stage object detector** which directly predicts bounding boxes and class probabilities from full images in one evaluation, enabling fast real-time performance.

Key architectural components:

- **Backbone Network:** Feature extractor (e.g., CSPDarknet or other CNN variants).
- **Neck:** Fuses features across scales (FPN/PAN structure).
- **Head:** Final layers that predict bounding boxes, class probabilities, and object confidence.

YOLO-V8 introduces improvements in training stability, inference speed, and modularity within Ultralytics' framework. viso.ai

3.2 Project System Architecture

Include a block diagram showing:

- **Input:** Road images/video
- **Preprocessing:** Resizing, normalization, augmentation
- **Model Training:** YOLO-V8 train loop
- **Validation & Testing:** Performance metrics
- **Deployment / Inference:** Image/Video detection output

3.3 Dataset Preparation

- **Dataset Structure:**

  Dataset/

  ├── data.yaml

  ├── train/

  │ ├── images/

  │ └── labels/

  └── valid/

  ├── images/

  └── labels/

- Images: 2105 pothole-labeled road images.

- Labels: YOLO format (class + bounding box normalized coordinates).

3.4 YOLO-V8 Training Pipeline

Explain:

- Installation of dependencies (requirements.txt)
- Data loading and augmentation
- Command used:

python pothole_detection.py --train

Training uses default Nano model (yolov8n) offering real-time performance with moderate accuracy — explain model choice rationale.

3.5 Key Algorithms and Operations

Break down code sections (e.g., preprocessing, training loop) and explain logic.

- **Annotation parsing**
- **Batch training**
- **Loss functions**
- **Backpropagation**

3.6 Inference and Detection Module

Show how images and videos are processed:

python pothole_detection.py --detect --source path/to/video.mp4

Discuss how bounding boxes are drawn, confidence thresholds, and result logging.

4. Results and Evaluation

4.1 Metrics Used

Discuss commonly used object detection metrics:

- **Precision**
- **Recall**
- **mAP (mean Average Precision)**
- **IoU (Intersection over Union)**

Provide formulas and explain their significance.

Precision: 98.4% — extremely accurate when detecting potholes
Detection Accuracy (mAP@0.5): 94%

Supports both image and video input for real-time detection

4.2 Training Results

Present:

- Training loss curves with epochs
- Validation accuracy or confusion matrices (tables/charts)
- Visual examples of detection outputs (with bounding boxes)

Discuss performance here.

4.3 Inference Performance

Show several test images and videos demonstrating pothole detection accuracy and highlighting false positives/negatives. Include tables summarizing detection confidence values.

5. Discussion

5.1 Challenges in Pothole Detection

- Variability in pothole shapes and sizes
- Diverse lighting and weather conditions
- Limited dataset size

Refer to general pothole detection challenges from literature.

5.2 Possible Improvements

- Collect larger datasets or use augmentation
- Try segmentation models (YOLOv8-seg) for detailed pothole borders
- Fine-tune hyperparameters for better accuracy

Refer to recent works improving YOLO-V8 performance.

6. Conclusion

Reiterate that the system provides an automated way to detect potholes with reasonable accuracy and can be expanded for production usage (e.g., road inspection vehicles, smart cities monitoring systems).

7. References

Include all sources used in documentation:

Journal, web, and library references:

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. YoLO: Unified, Real-Time Object Detection. Wikipedia.
2. YOLO-V8 custom pothole detection repository (similar examples).
3. Pothole Detection research with YOLOv8.
4. YOLO-V8 training on custom dataset tutorial.
5. Enhanced YOLOv8 model for pothole detection.

Pothole Detection System

Output Video :

https://www.linkedin.com/posts/deepak-kumar-124909249_computervision-yolov8-potholedetection-activity-7338099005895122945-Y7Wk?utm_source=share&utm_medium=member_desktop&rcm=ACoAAD2JeWwB0sWFJwrECik8x8cOStt8PqqH_0Y