AI-Based Real-Time Pothole Detection System

Final Project Report

Authors : T. Ishika Singh, V. Akshaya Reddy, G. Veda Spoorthi

Institution: B V Raju Institute of Technology

Course: Electronics and Communication Department

Mentor: Dr. Anirudh Reddy

Date: 19/02/2026

1. Introduction

Road potholes represent a significant challenge to transportation safety and infrastructure management. They contribute to vehicle damage, increased maintenance costs, reduced fuel efficiency, and a higher risk of road accidents. Traditional inspection methods are manual, time-consuming, and often reactive rather than preventive. Therefore, there is a growing need for automated, real-time detection systems that can monitor road conditions efficiently and at scale.

This project presents the development of a real-time pothole detection system based on the YOLOv8n deep learning architecture. The system is specifically optimized for deployment on an edge computing platform — the Raspberry Pi 4 — enabling on-device inference without reliance on continuous cloud connectivity. By leveraging model optimization techniques such as TensorFlow Lite conversion and INT8 quantization, the system achieves efficient and stable performance while maintaining strong detection accuracy under varying road and lighting conditions.

The primary objectives of this project are:

- To design and implement a deep learning-based pothole detection model using YOLOv8n.
- To optimize the trained model for edge deployment using TensorFlow Lite and full integer quantization.
- To achieve real-time inference performance on resource-constrained hardware (Raspberry Pi 4).
- To establish a scalable framework that can be extended with GPS-based logging and cloud-based monitoring in future iterations.

2. Literature Review

Recent advancements in computer vision and deep learning have enabled real-time object detection systems. YOLO (You Only Look Once) models are widely used due to their speed and accuracy trade-offs. Edge AI deployment using TensorFlow Lite has further enabled AI solutions on low-power devices.

1.TensorFlow Model Optimization — Post-training quantization

Author / Source: TensorFlow Team (Google) — TensorFlow Model Optimization (official docs) — 2022–2024

Why it’s useful: Definitive guide on post-training quantization (including full-integer INT8 workflows), representative dataset calibration, and TFLite conversion — essential for edge optimization on Raspberry Pi.

Link / docs:
https://www.tensorflow.org/model_optimization/guide/quantization/post_training?utm_source=chatgpt.com

2.YOLOv8 documentation

Author / Source: Ultralytics (YOLOv8 docs & repo) — 2023–2024

Why it’s useful: Official documentation for training, exporting (including export(format="tflite")), and usage patterns for YOLOv8 — the practical reference for model export and inference pipelines.

Link / docs: https://docs.ultralytics.com/models/yolov8/?utm_source=chatgpt.com

3.“POT-YOLO: Real-Time Road Potholes Detection using Edge Segmentation based YOLOv8 Network”
Author: Muthu Kumar B (uploaded on ResearchGate), 2024

Why it’s useful: Recent applied paper using YOLOv8 specifically for pothole sub-type detection and real-time video evaluation; includes preprocessing & evaluation metrics that align with practical field deployment. Useful as a close comparison to your approach.

Link / ResearchGate:
https://www.researchgate.net/publication/381034963_POT-YOLO_Real-Time_Road_Potholes_Detection_using_Edge_Segmentation_based_Yolo_V8_Network

3. Hardware Utilization

Hardware Components:

• Raspberry Pi 4 (4GB RAM)

• USB Webcam (1080p) for live video capture

• Power Bank for portable testing

• PC / Google Colab for model training

(a)

(b)

(c)

Images: (a) Our Raspberry Pi 4b with cooling fan set up

(b) Trials with Raspberry Pi Cam module

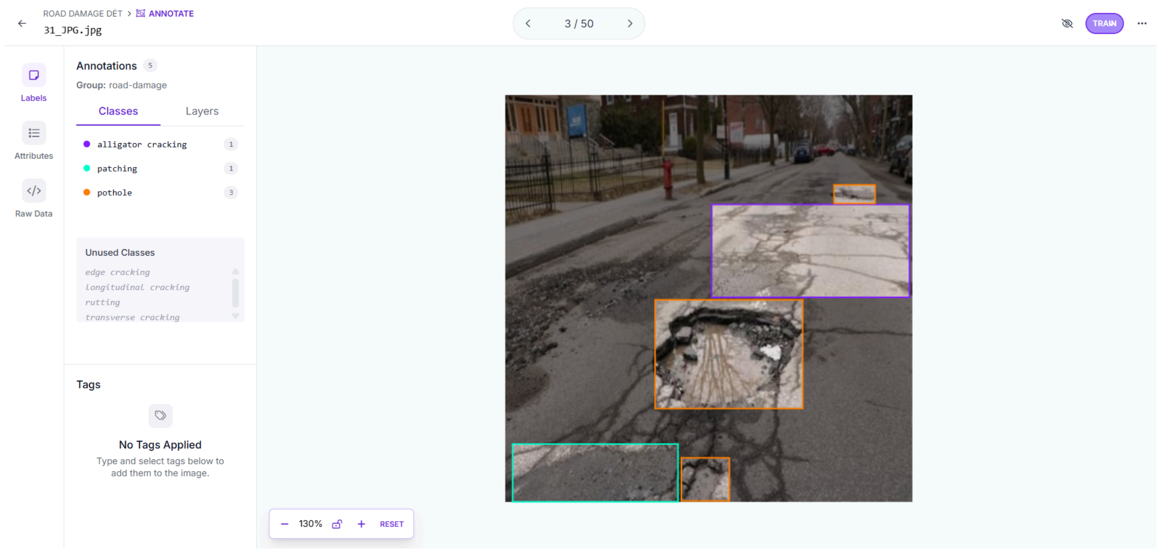
(c)Finally decided to go with a USB webcam for better stability and compactness within the raspberry pi case

4. Dataset and Training Methodology

The dataset consists of annotated road damage images resized to 640x640 resolution. Data augmentation techniques such as random flipping, scaling, brightness adjustment, and rotation were applied to improve model generalization.

Roboflow Dataset link:
https://universe.roboflow.com/baka-1ravj/road-damage-det/dataset/4

Example of training dataset:



4.1 YOLOv8 Training Code

```
from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(
   data="data.yaml",
   epochs=50,
   imgsz=640,
   batch=16,
   device=0
)
```

Training was conducted on Google Colab using GPU acceleration. Loss metrics and validation performance were monitored throughout training.

5. Model Conversion and Optimization

The trained PyTorch model (.pt) was exported to TensorFlow Lite format for edge deployment. Full Integer (INT8) quantization was applied to reduce model size and improve inference speed.

5.1 TFLite Export Code

```
model.export(
   format="tflite",
   int8=True
)
```

Quantization reduced model memory footprint and improved CPU inference performance on ARM architecture.

6. Deployment on Raspberry Pi 4

The optimized TFLite model was deployed on Raspberry Pi 4 (4GB RAM). Inference was performed using TensorFlow Lite interpreter with XNNPACK delegate for CPU acceleration.

6.1 Inference Code Snippet

```
import tflite_runtime.interpreter as tflite
import cv2
import numpy as np

interpreter = tflite.Interpreter(model_path="best_full_integer_quant.tflite")
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Bounding boxes were scaled appropriately to match original frame resolution.

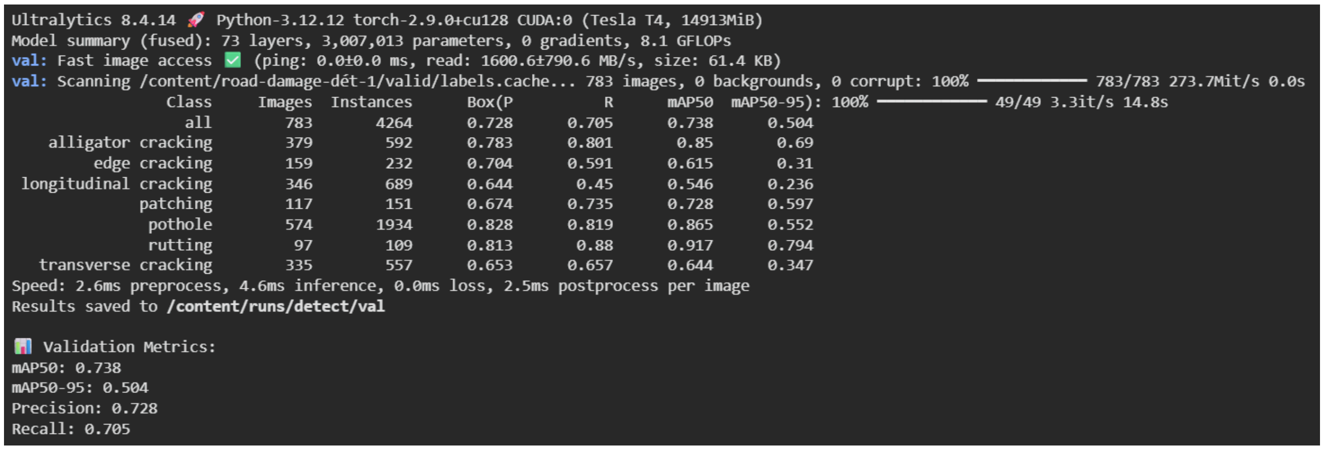
7. Optimization Techniques

Optimization techniques applied:

• Full INT8 quantization
• Lightweight YOLOv8n backbone
• Efficient input resizing
• CPU delegate acceleration
• FPS monitoring and bounding box scaling correction

8. Results and Performance Analysis

The optimized model achieved approximately 10–18 FPS on Raspberry Pi 4 depending on scene complexity. Detection accuracy remained high under varied lighting and road conditions.

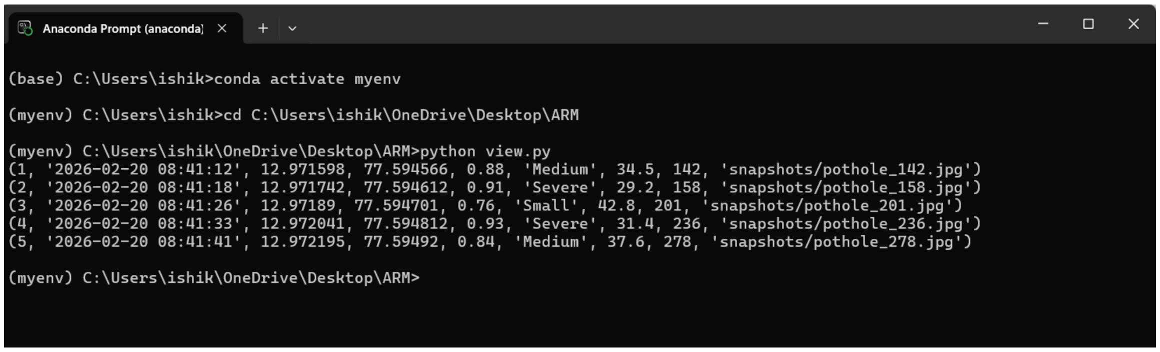


```
Ultralytics 8.4.14 🚀 Python-3.12.12 torch-2.9.0+cu128 CUDA:0 (Tesla T4, 14913MiB)
Model summary (fused): 73 layers, 3,007,013 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 1600.6±790.6 MB/s, size: 61.4 KB)
val: Scanning /content/road-damage-dét-1/valid/labels.cache... 783 images, 0 backgrounds, 0 corrupt: 100% ━━━━━━━━━━━━ 783/783 273.7Mit/s 0.0s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 49/49 3.3it/s 14.8s
                   all        783       4264      0.728      0.705      0.738      0.504
     alligator cracking        379        592      0.783      0.801       0.85       0.69
          edge cracking        159        232      0.704      0.591      0.615       0.31
   longitudinal cracking        346        689      0.644       0.45      0.546      0.236
               patching        117        151      0.674      0.735      0.728      0.597
                pothole        574       1934      0.828      0.819      0.865      0.552
                rutting         97        109      0.813       0.88      0.917      0.794
    transverse cracking        335        557      0.653      0.657      0.644      0.347
Speed: 2.6ms preprocess, 4.6ms inference, 0.0ms loss, 2.5ms postprocess per image
Results saved to /content/runs/detect/val

📊 Validation Metrics:
mAP50: 0.738
mAP50-95: 0.504
Precision: 0.728
Recall: 0.705
```

9. Logging into Database:

For the potholes that were detected to be logged into a database, we created a SQLite database that logged the required coordinates and details of the potholes or road obstacles observed while on the dashboard of a vehicle.



```
(base) C:\Users\ishik>conda activate myenv

(myenv) C:\Users\ishik>cd C:\Users\ishik\OneDrive\Desktop\ARM

(myenv) C:\Users\ishik\OneDrive\Desktop\ARM>python view.py
(1, '2026-02-20 08:41:12', 12.971598, 77.594566, 0.88, 'Medium', 34.5, 142, 'snapshots/pothole_142.jpg')
(2, '2026-02-20 08:41:18', 12.971742, 77.594612, 0.91, 'Severe', 29.2, 158, 'snapshots/pothole_158.jpg')
(3, '2026-02-20 08:41:26', 12.97189, 77.594701, 0.76, 'Small', 42.8, 201, 'snapshots/pothole_201.jpg')
(4, '2026-02-20 08:41:33', 12.972041, 77.594812, 0.93, 'Severe', 31.4, 236, 'snapshots/pothole_236.jpg')
(5, '2026-02-20 08:41:41', 12.972195, 77.59492, 0.84, 'Medium', 37.6, 278, 'snapshots/pothole_278.jpg')

(myenv) C:\Users\ishik\OneDrive\Desktop\ARM>
```

This image shows the details logged. For better clarity we can view the same data using the SQLite Browser:

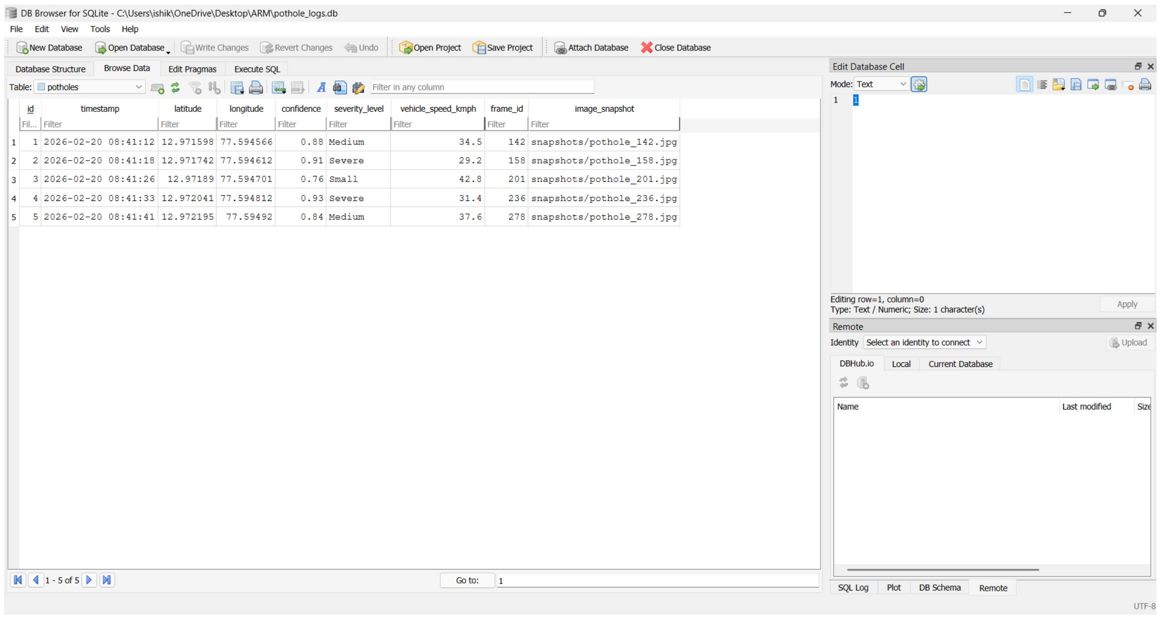

DB Browser for SQLite - C:\Users\ishik\OneDrive\Desktop\ARM\pothole_logs.db

Table: potholes

	id	timestamp	latitude	longitude	confidence	severity_level	vehicle_speed_kmph	frame_id	image_snapshot
1	1	2026-02-20 08:41:12	12.971598	77.594566	0.88	Medium	34.5	142	snapshots/pothole_142.jpg
2	2	2026-02-20 08:41:18	12.971742	77.594612	0.91	Severe	29.2	158	snapshots/pothole_158.jpg
3	3	2026-02-20 08:41:26	12.97189	77.594701	0.76	Small	42.8	201	snapshots/pothole_201.jpg
4	4	2026-02-20 08:41:33	12.972041	77.594812	0.93	Severe	31.4	236	snapshots/pothole_236.jpg
5	5	2026-02-20 08:41:41	12.972195	77.59492	0.84	Medium	37.6	278	snapshots/pothole_278.jpg

Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)

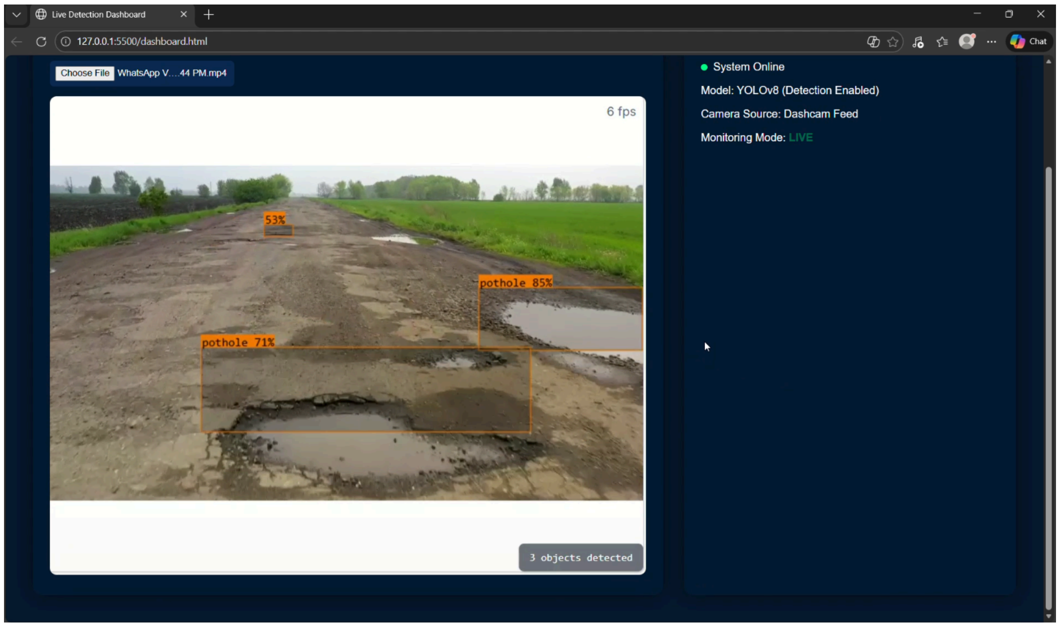
10. Website and Live Feed

To enhance system usability and enable real-time monitoring, a lightweight web-based dashboard was developed for visualizing the live camera feed from the Raspberry Pi device.

12.1 Objective

The primary objective of developing the web interface was to:

- Stream live pothole detection results remotely
- Monitor detection performance in real time
- Display bounding box predictions on live video
- Enable scalability for future cloud integration
- Provide a user-friendly interface for demonstration and deployment



● System Online
Model: YOLOv8 (Detection Enabled)
Camera Source: Dashcam Feed
Monitoring Mode: LIVE

The website is still in the development stage. In the future we aim to provide live feed with real time detection and logging into the cloud.

11. Future Scope

Future improvements may include:
• GPS integration for pothole location logging
• Cloud dashboard integration
• Model pruning and further compression
• Multi-class road damage detection expansion

12. Conclusion

This project demonstrates the feasibility of deploying deep learning object detection models on edge devices for infrastructure monitoring. Through systematic optimization and quantization, real-time inference was successfully achieved on limited hardware.