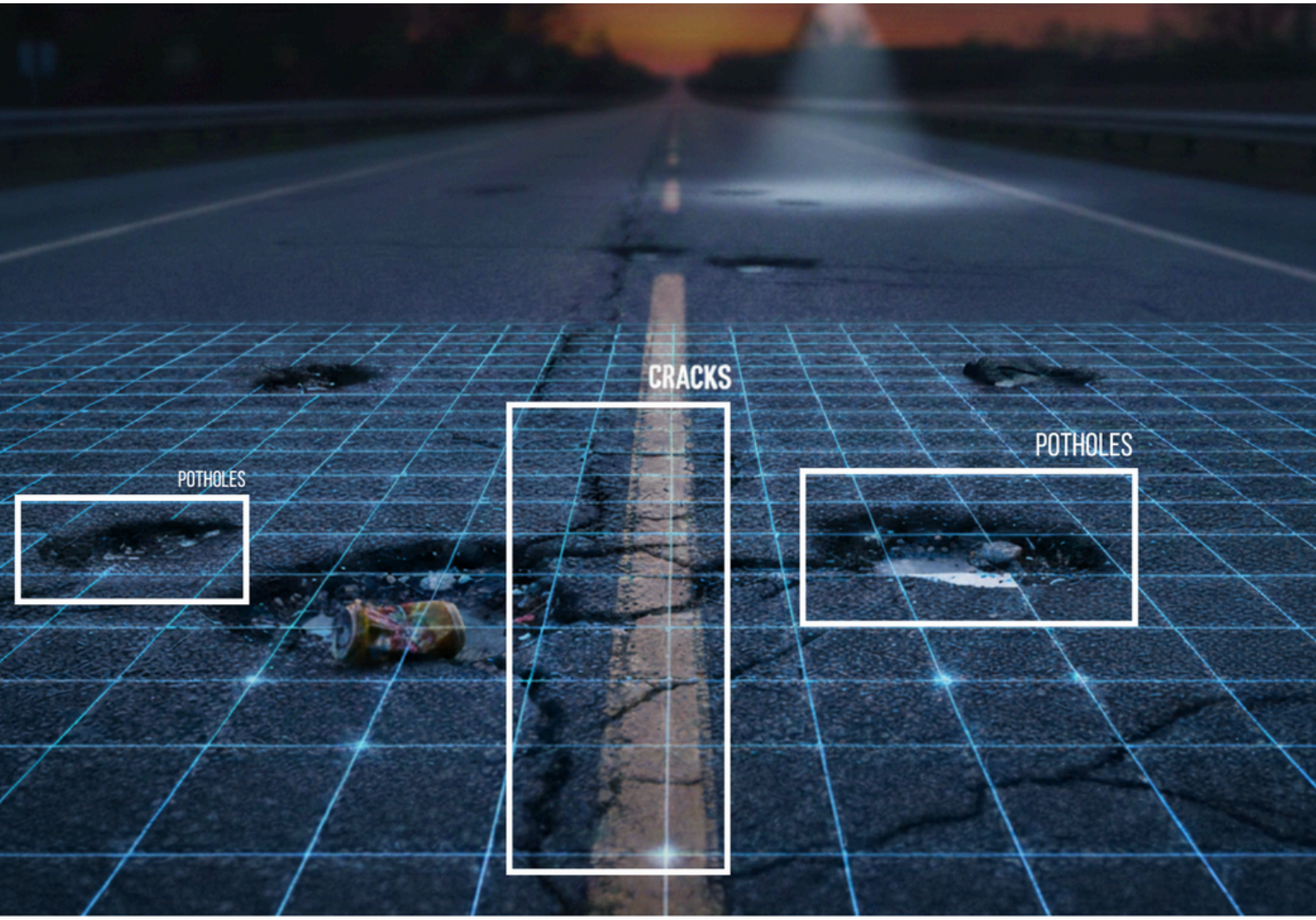
REAL-TIME ROAD ANOMALY DETECTION FROM DASHCAM FOOTAGE ON RASPBERRY PI



INTRODUCTION

India's vast 6.6 million km road network powers 85% of passenger traffic and 71% of freight, yet potholes, cracks, and open manholes demand urgent action. These defects claim 9,438 lives (up 53% from 2020-2024), spark 23,056 accidents, and injure 19,956 people .The scale of the problem is compounded by the inadequacy of existing inspection methods. Road condition monitoring in India continues to rely heavily on manual field surveys , a process that is slow, labor-intensive, inconsistent across states.

This project proposes a real-time road anomaly detection system built around a YOLOv8 deep learning model deployed on a Raspberry Pi 5 The system mounts on any moving vehicle, captures continuous road footage via a webcam, performs on-device inference to detect defects such as potholes, surface cracks, and open manholes, and automatically logs detection events for downstream reporting and repair prioritization.

The *novelty of the system* lies in three properties that are rarely achieved together in existing literature: real-time detection capability, fully edge-based inference without reliance on cloud connectivity, and hardware portability at a fraction of the cost of specialized road survey vehicles.By enabling proactive, scalable, and low-cost defect mapping, this system offers a practical pathway toward data-driven road maintenance.

Pothole

Rutting

Crack

Open Manhole

Detected Road Anomalies in Tar roads

Class	Description
Pothole	Depression in road surface caused by wear and weather
Crack	Linear or alligator patterns indicating structural damage
Open Manhole	Exposed utility access points posing safety hazard
Rutting	Longitudinal depressions in wheel paths from repeated traffic

1. System Overview

The proposed system is a portable, real-time road anomaly detection system built on a Raspberry Pi 5 single-board computer. The system operates autonomously without requiring a monitor, keyboard, or mouse, and is powered by a portable power source, making it suitable for field deployment on public roads. The pipeline integrates computer vision, deep learning inference, in-built storage & cloud storage, and instant messaging notification into a unified embedded system.

2. Hardware Components

The system comprises the following hardware:

1. Raspberry Pi 5 — serves as the central processing unit responsible for running the detection model, managing peripherals, and handling network communication
2. USB Webcam (Logitech UVC Camera) — captures real-time video footage of the road surface at a resolution of 1280×720 pixels
3. Portable Power source — provides mobile power supply enabling field deployment without mains electricity
4. MicroSD Card — stores the operating system, trained model, and recorded video segments
5. Wi-Fi / Internet— provides internet connectivity for uploading to Google Drive and sending Telegram notifications

3. Software Architecture

The software stack is built entirely on open-source tools and runs on Raspberry Pi OS (Ubuntu-based Linux). The core components are:

- Python 3 — primary programming language
- OpenCV — video capture, frame preprocessing, bounding box rendering, and video writing
- ONNX Runtime — hardware-agnostic deep learning inference engine for running the exported YOLOv8 model
- Google Drive API — cloud storage for periodic video upload
- Telegram Bot API — instant notification delivery for detected anomalies
- psutil — real-time system monitoring of CPU, RAM, FPS, and inference time
- systemd — Linux service manager used to enable autoboot on power supply

4.Methodology

Processing Pipeline

1. Frame Acquisition: USB webcam captures 640×480 frames at target 15 FPS
2. Inference: YOLO model processes every 2nd frame (inference skip optimization)
3. Detection Threshold: Confidence threshold set to 0.55 to minimize false positives
4. Event-based Recording: System buffers 2 seconds pre-detection, records until anomaly absent for 3 seconds
5. Annotation: Bounding boxes and class labels drawn on frames in real-time.
6. Location Capture: IP-based geolocation via free API (ip-api.com)
7. Storage: Video (.mp4), snapshot (.jpg), and location data (.txt) saved in timestamped folders
8. Notification: Telegram alert with snapshot and location sent instantly
9. Cloud Upload: Folder uploaded to Google Drive via rclone in background thread

5. Model Training and Export

The anomaly detection model was trained using the YOLOv8 (You Only Look Once version 8) object detection architecture. The model was trained on a custom dataset containing four classes of road anomalies:

- Rutting — longitudinal depressions caused by repeated traffic loading
- Open Manhole — exposed or displaced manhole covers posing safety hazards
- Potholes — bowl-shaped surface depressions caused by pavement deterioration
- Crack — linear or alligator surface fractures indicating structural weakness

After training, the model was exported to ONNX (Open Neural Network Exchange) format to enable platform-independent inference on the Raspberry Pi without requiring PyTorch or GPU dependencies. The exported model accepts input of shape [1, 3, 640, 640] and produces output of shape [1, 8, 8400], where 8 represents 4 bounding box coordinates and 4 class confidence scores across 8400 anchor points.

6.AI Model Results:

Dataset Source: Collected from Kaggle and Roboflow, containing annotated images of road anomalies.

- Dataset Size: Train: 5028, Test: 368, Validation: 362
- Training Framework: YOLOv8 via Ultralytics
- No of Epochs: 80
- Hardware Used for Training: Google Colab Pro (NVIDIA A100 GPU)
- Model Export: Converted to ONNX format for deployment

```
# Verify GPU
print("GPU available:", torch.cuda.is_available())
print("GPU name:", torch.cuda.get_device_name(0))

# Choose model size (A100 can handle bigger models)
model = YOLO('yolov8m.pt')   # try 'yolov8m.pt' if you want even better accuracy

model.train(
    data='/content/drive/MyDrive/armhack/armhack.v3i.yolov8/data.yaml',  # change if your datas
    epochs=80,                # more epochs since A100 is fast
    imgsz=640,
    batch=32,                 # larger batch size for A100
    device=0,                 # GPU 0 (A100)
```

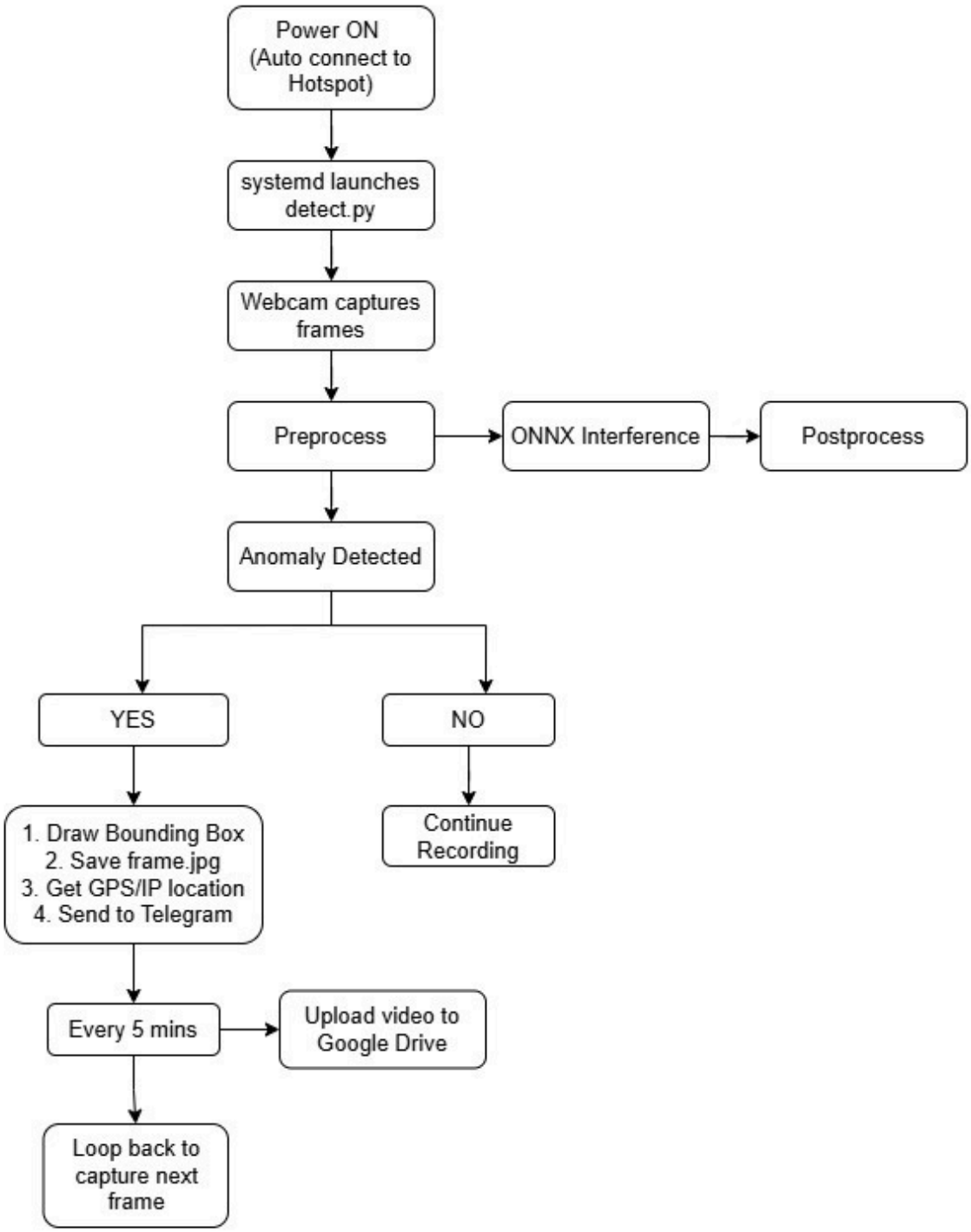
```
               all        362        408      0.724      0.757      0.766      0.446

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      77/80      13.5G     0.4445     0.2916     0.9085         11        640: 100% ──────────── 158/158 4.6i
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 6
                   all        362        408      0.867       0.67      0.781       0.46

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      78/80      13.7G     0.4398     0.2952     0.9061          7        640: 100% ──────────── 158/158 4.6i
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 6
                   all        362        408      0.767      0.734      0.772      0.441

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      79/80      13.8G     0.4226     0.2774     0.9001         24        640: 100% ──────────── 158/158 4.6i
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 6
                   all        362        408      0.791      0.711      0.781      0.449

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      80/80        14G      0.424     0.2785     0.9043          3        640: 100% ──────────── 158/158 4.7i
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 6
```



System Workflow

7.Novel Subsystems:

A.Location Tagging

Since no dedicated GPS hardware module is used, the system obtains geographic coordinates through IP-based geolocation using the ipinfo.io API. Upon anomaly detection, the system retrieves the current latitude and longitude and generates a plain text file containing the coordinates and a direct Google Maps hyperlink for field verification.

B.Notification System

When an anomaly is detected, the system triggers a Telegram Bot notification limited to once every 30 seconds to prevent message flooding. The notification package sent to the designated Telegram chat includes:

The annotated frame image (JPEG) showing the detected anomaly with bounding box

A location text file containing timestamp, latitude, longitude, and Google Maps link

All Telegram transmissions are handled in a separate background thread to avoid blocking the main detection loop.

C.Cloud Storage (Additional Layer)

The system automatically segments recorded video into 5-minute clips. At the end of each segment, the completed video file is uploaded to a designated Google Drive folder using a Service Account credential, eliminating the need for manual authentication. Uploads are handled asynchronously in background threads to maintain continuous recording without interruption.

D.System Monitoring

The terminal continuously displays real-time performance metrics including frames per second (FPS), model inference time in milliseconds, CPU utilization percentage, and RAM usage percentage. This allows operators to monitor system health during field deployment.

E. Autonomous Operation

The system is configured for fully autonomous operation through the following mechanisms:

- Wi-Fi Auto-connect: NetworkManager is configured with the mobile hotspot credentials so the Pi connects automatically on boot without user intervention
- Autoboot Service: A systemd service unit is registered to launch the detection script automatically on every power-on with a 20-second delay to allow network initialization and Fault Recovery: The systemd service is configured with Restart always ensuring the detection process automatically restarts if it crashes unexpectedly

8.Technology Stack

AI and Machine Learning Layer

The intelligence core responsible for anomaly detection.

1. YOLOv8 Ultralytics YOLOv8 architecture Object detection model architecture
2. Custom Dataset 4 classes of road anomalies Training data
3. ONNX Format Open Neural Network Exchange Cross-platform model export format
4. ONNX Runtime CPU Execution Provider Runs model on Pi without GPU
5. Input Tensor Shape: [1, 3, 640, 640] Preprocessed frame fed to model
6. Output Tensor Shape: [1, 8, 8400] Raw detections from model
7. NMS Algorithm IoU threshold: 0.45 Removes duplicate detections
8. Confidence Filter Threshold: 0.25 Filters weak detections

Connectivity and Network Layer

Handles all internet communication for the system.

1. NetworkManager (nmcli) WPA2-PSK Wi-Fi Auto-connect to mobile hotspot
2. ipinfo.io API HTTPS/REST IP-based geolocation fallback
3. Telegram Bot API HTTPS/REST Send anomaly alerts and images
4. Google Drive API v3 HTTPS/OAuth2 Cloud video storage upload
5. requests library HTTP client All API communications

Notification Layer

Delivers real-time anomaly alerts to the operator.

1. Telegram Bot API HTTPS REST Notification delivery platform
2. sendPhoto endpoint Multipart POST Sends annotated anomaly frame
3. sendDocument endpoint Multipart POST Sends location text file
4. Bot Token Unique API key Bot authentication
5. Chat ID Target chat identifier Recipient identification
6. 30-second cooldown Time-based throttle Prevents message flooding
7. Background threading Python threading Non-blocking notification sending

System Monitoring Layer
Provides real-time visibility into system performance during operation.

1. FPS time module Frames per second
2. Inference Time time module Milliseconds per frame
3. CPU Usage psutil Percentage utilization
4. RAM Usage psutil Percentage utilization
5. Detection Count len(detections) Anomalies per frame

Open Source Tools Used
Tool License Version

1. Python PSF License 3.11+
2. OpenCV Apache 2.0 4.8+
3. ONNX Runtime MIT License 1.17+
4. NumPy BSD License 1.24+
5. psutil BSD License 5.9+
6. YOLOv8 (Ultralytics) AGPL-3.0 8.x
7. Raspberry Pi OS Mixed open source Bookworm

9.Performance Metrics Monitored

1. FPS Frames processed per second
2. Inference Time Time taken by ONNX model per frame (ms)
3. CPU Usage Processor load on Raspberry Pi 5 (%)
4. RAM Usage Memory consumption during runtime (%)
5. Detection Count Number of anomalies found per frame

RESULTS

1.Performance Metrics

Metric	Value
Frame Rate (FPS)	12-15 FPS (capture) / 7-8 FPS (inference)
Detection Latency	130-150 ms per inference
Confidence Threshold	0.55 (optimized for precision)
Input Resolution	640×480 pixels
Telegram Notification Delay	<2 seconds (with network)
Location Accuracy	5-50 km (IP-based geolocation)

2. System Capabilities

The system successfully demonstrates autonomous operation with the following capabilities:

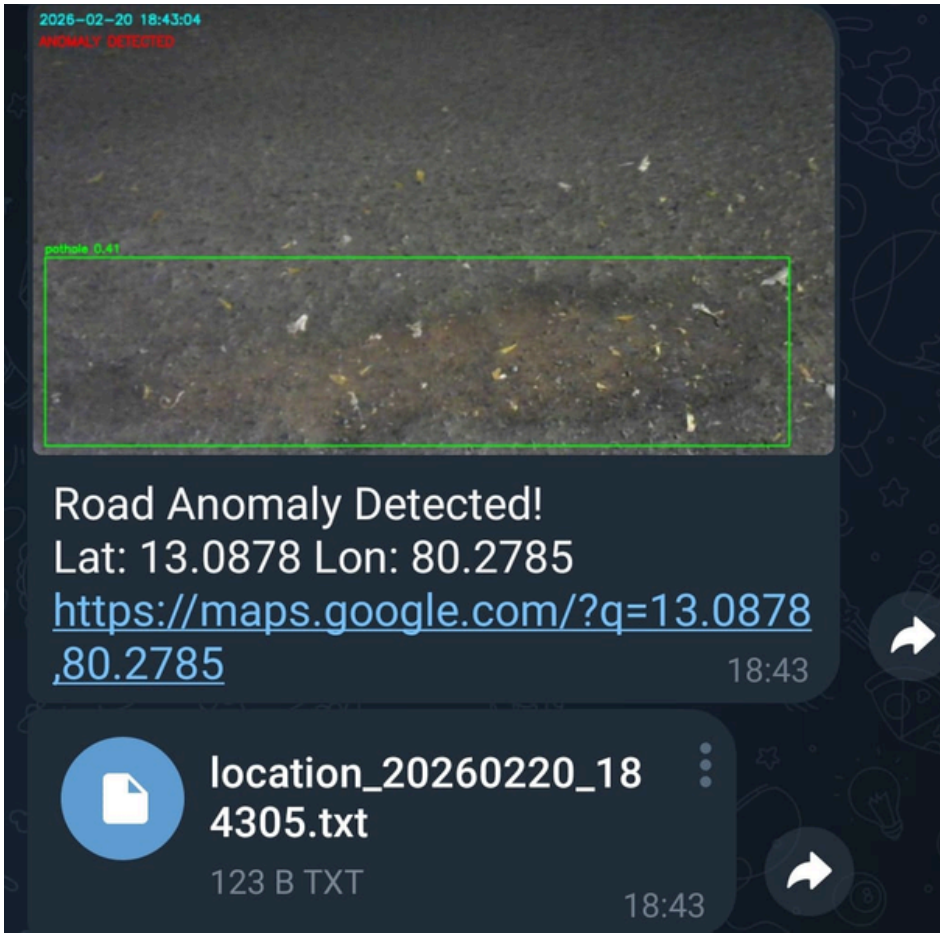
• Real-time detection at 7-8 inferences per second, sufficient for road survey speeds up to 40 km/h

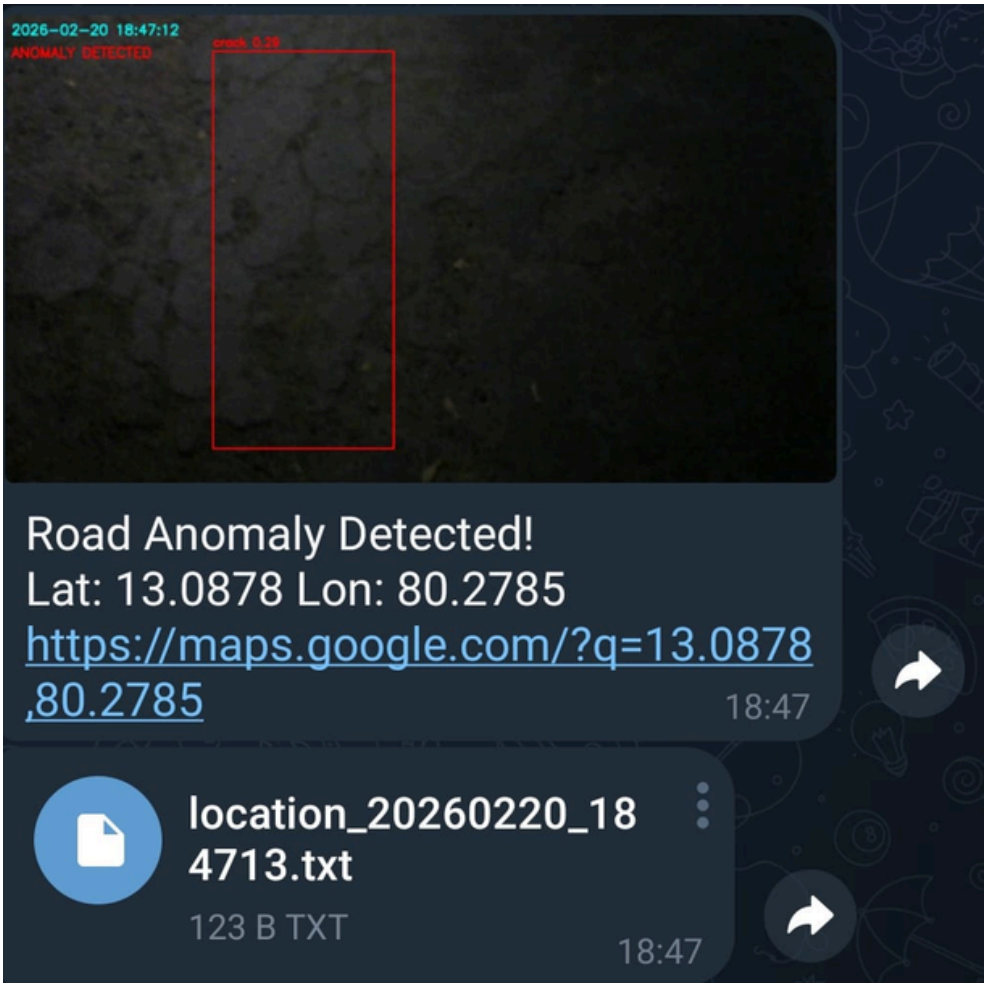
• Event-based recording reduces storage consumption by 90% compared to continuous recording

• Instant notifications enable rapid response for critical infrastructure issues
• Automated cloud backup ensures data integrity and remote accessibility
• Portable operation via mobile hotspot connectivity for field deployment
• Auto-start functionality enables deployment without technical supervision

3. Output Data Structure

Each detection event generates a timestamped folder containing three files:
1. Snapshot image (JPG): Frame with annotated bounding boxes at moment of detection
2. Video clip (MP4): Event-based recording with 2-second pre-buffer and 3-second post-detection
3. Location metadata (TXT): Coordinates, timestamp, anomaly classes, and Google Maps link





CONCLUSION

1.Works best even at low light environment.

2.Higher Accuracy

3.No need of cloud