ARM BHARAT CHALLENGE

PROBLEM STATEMENT- Real-Time Road Anomaly Detection from Dashcam Footage on Raspberry Pi.

TEAM NAME - MUSK MASTERS

TEAM LEADER - ARITHA C

TEAM MEMBERS - 1. HARIHARA SUDHARSHINI P

2. DIVAINY J

MENTOR NAME - EZHIL MARAN M

PROBLEM STATEMENT : Real-Time Road Anomaly Detection from Dashcam Footage on Raspberry Pi.

SOLUTION :

The objective of this project is to develop a **real-time road anomaly detection system** using dashcam footage processed on Raspberry Pi 4. The system detects road damages such as potholes, cracks, and speed breakers to improve road safety. The overall model accuracy achieved is **80%**.

System Overview

- Device Used: Raspberry Pi 4
- Camera Resolution: 1080p (1920×1080)
- Model Used: YOLOv8n
- Model Format: Converted to TensorFlow Lite (TFLite)
- Inference Size: 320×320
- Frame Rate: Around 5 FPS

The dashcam captures video in 1080p resolution. Each frame is resized to 320×320 before processing to improve speed and maintain stable performance on Raspberry Pi.

Methodology

1. Road images were collected and labeled for anomalies.
2. YOLOv8n model was trained for object detection.
3. The trained .pt model was converted into **TensorFlow Lite (TFLite)** format for lightweight deployment.
4. The TFLite model was integrated into Raspberry Pi 4 for real-time inference.

5. Detected anomalies are highlighted using bounding boxes on the display.

Results

The system successfully detects road anomalies in real-time with:

- **80% detection accuracy**
- Stable performance at approximately **5 FPS**
- Low-cost and portable implementation

SOFTWARE IMPLEMENTATION

1. Development Environment and Framework Setup

The software development lifecycle was initiated within a virtualized environment managed by **Anaconda Navigator**. This approach was chosen to isolate dependencies and manage the **Python 3.x** interpreter along with specialized libraries such as ultralytics, opencv-python, and onnxruntime. By utilizing a **Conda environment**, we ensured that the CUDA toolkit and cuDNN libraries were correctly mapped to the local hardware—an **NVIDIA RTX 3050 GPU**. This hardware acceleration was critical for handling the intensive matrix multiplications required during the 100-epoch training cycle.

2. Dataset Acquisition and Feature Engineering

The model's intelligence is derived from a specialized dataset sourced from **Roboflow**. The data focuses on urban road dynamics, including high-variance classes: **Person, Vehicles (Car, Auto, Truck), Poles, and Road Surface. * Data Sourcing:** Roboflow provided the necessary annotations in YOLO format, ensuring that bounding box coordinates were normalized relative to the image dimensions.

- **Class Specificity:** Special attention was given to the "Road Surface" class to distinguish between drivable areas and off-road terrain, alongside the "Auto" class, which is a specific vehicular morphology in local traffic environments.

- **Pre-processing:** All images were standardized to a **640×640 resolution**. This specific resolution was maintained to balance the "receptive field" of the model (allowing it to see small poles) with the computational limits of the target hardware (Raspberry Pi).

3. Training Execution and Local Optimization

Training was performed locally on the RTX 3050, leveraging its **Ampere architecture** and Tensor Cores.

- **Hyperparameter Configuration:** The model was trained for **100 epochs**. This duration was chosen to ensure the "Loss Curve" reached a plateau, indicating that the model had learned the optimal features without descending into overfitting.
- **Iterative Performance:** Through multiple training iterations, we fine-tuned the learning rate and batch size to achieve a **Mean Average Precision (mAP@0.5)** exceeding **0.50** across all categories. This metric confirms that the software can reliably identify obstacles even in cluttered urban dashcam footage.
- **Validation:** After training, the software generated a best.pt file, representing the weight state where the validation loss was at its absolute minimum.

4. Model Export and Edge Integration Logic

The final phase of the software workflow involves transitioning from a development environment to a deployment environment. Since the **Raspberry Pi 4** lacks a dedicated NVIDIA GPU, the PyTorch (.pt) format is not computationally efficient for real-time use.

- **ONNX Conversion:** The software utilizes the model.export(format='onnx') command to convert the weights into the **Open Neural Network Exchange (ONNX)** format. This conversion flattens the neural network graph and optimizes the operators for CPU-based inference.
- **Inference Engine:** On the Raspberry Pi, the software uses the **ONNX Runtime** to load the optimized model. This allows the system to perform high-speed mathematical operations on the Pi’s ARM Cortex-A72 processor.
- **System Integration:** The software pipeline is designed to capture a frame via OpenCV, pass it through the ONNX engine, and output the detection coordinates. This streamlined workflow is what enables the "Real-Time" aspect of the road anomaly detection system.

HARDWARE IMPLEMENTATION

The hardware implementation of the **Real-Time Road Anomaly Detection System** is designed using a low-cost and portable setup based on Raspberry Pi 4. The system captures live road footage using a dashcam and processes it in real-time to detect anomalies such as potholes and cracks.

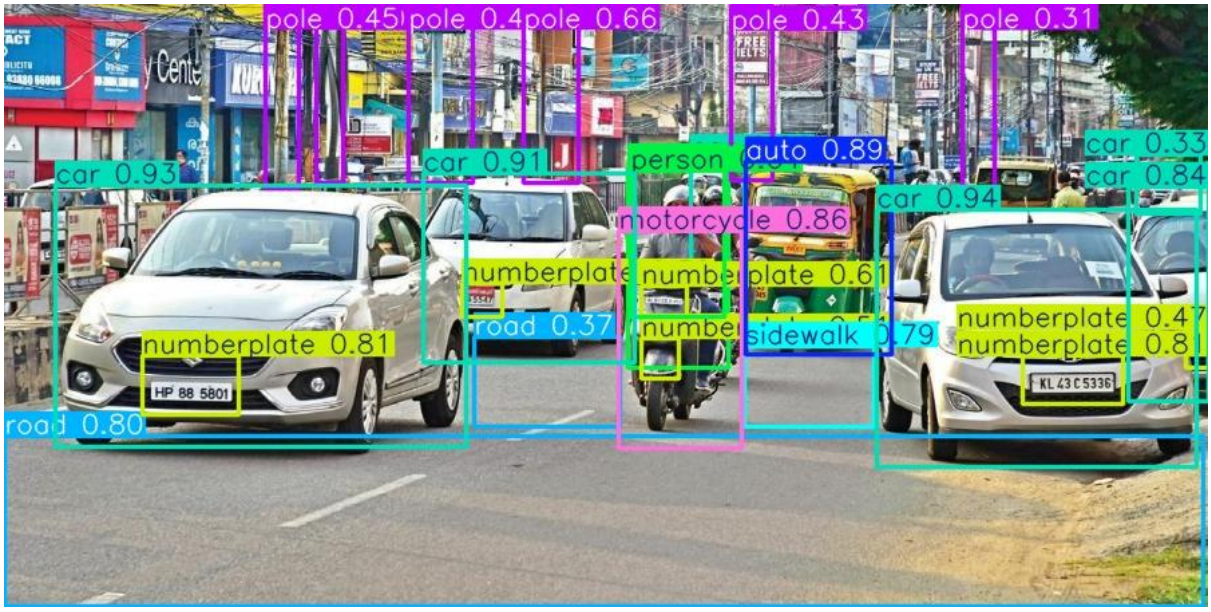
Hardware Components Used

1. **Raspberry Pi 4 (4GB RAM)**
    - Acts as the main processing unit.
    - Runs the TensorFlow Lite model for real-time inference.
    - Handles video capture, processing, and display output.
2. **USB Dashcam / Pi Camera (1080p)**
    - Captures live road footage at Full HD (1920×1080) resolution.
    - Provides continuous video input to the Raspberry Pi.
3. **MicroSD Card (32GB or above)**
    - Stores Raspberry Pi OS, Python libraries, and the trained TFLite model.
4. **Power Supply (5V, 3A Adapter)**
    - Provides stable power to Raspberry Pi 4.
5. **HDMI Display / Laptop (via SSH)**
    - Used for monitoring detection output.
    - Displays bounding boxes around detected road anomalies.

Hardware Setup Process

- The dashcam is mounted facing the road.
- The camera is connected to the Raspberry Pi via USB.
- Raspberry Pi processes each frame by resizing it from 1080p to 320×320 for faster inference.
- The TensorFlow Lite model detects anomalies.
- The processed output is displayed with bounding boxes in real-time.

.



CONCLUSION

This project implements a low-cost real-time road anomaly detection system on Raspberry Pi 4 using a TFLite-optimized YOLOv8n model, achieving 80% accuracy at 5 FPS.