Technical Report

Project: Real-Time Road Anomaly Detection on Raspberry Pi 5

Name: Mohd Nouman Ahmed

Category: ECE / Embedded AI

Challenge: Bharat AI-SoC Student Challenge 2026

This report explains model selection, ARM optimizations, and performance benchmarks for the real-time road anomaly detection system deployed on the Raspberry Pi 5. The solution is designed as a CPU-only edge AI pipeline without any AI hats or hardware accelerators optimized for embedded deployment.

1) Model Selection: YOLO with NCNN Runtime

For real-time road safety applications, an object detection architecture is required for which I used **YOLOv8n** due to its performance and accuracy for embedded applications enabling real-time inference on live camera streams.

The trained YOLO model was exported to a best.pt file which was later on optimized to a more quantized model known as **NCNN (FP16)** — a lightweight, dependency-free inference engine specifically designed for embedded ARM CPUs and the best inference model for Raspberry Pi 5 for its optimization and accuracy.

Why NCNN?

- Highly optimized ARM kernels.
- Designed for CPU-only deployment.
- This makes NCNN an ideal inference backend for Raspberry Pi 5, providing a more reliable option for embedded AI deployments.

---

2) Optimization/Quantization Strategy

To achieve real-time performance on a CPU-only device, two levels of optimization were implemented:

2.1 FP16 (Half-Precision) Quantization

Standard deep learning models use FP32 weights, which double memory bandwidth requirements and reduce cache efficiency on embedded CPUs. By converting the model to FP16, the system achieves:

- **50% reduction** in model size.
- **Reduced memory bandwidth.**

- **Improved cache utilization.**
- **Better SIMD throughput on ARM NEON.**

This allows the Raspberry Pi 5's Cortex-A76 cores to process multiple pixels per instruction cycle using NEON SIMD registers, significantly increasing FPS.

2.2 Multi-Threaded Producer–Consumer Pipeline

A major drawback in embedded vision systems is disk I/O latency, especially when writing MP4 video to an SD card as we have to provide required clips for detected anomalies.

Problem: Video encoding and file writes block the main inference loop, which causes frame drops and delayed detections leading to inaccurate results.

Solution: The pipeline is split into two asynchronous threads:

- **Inference Thread (Producer):** Captures frames, runs YOLO detection, and maintains a **2-second sliding RAM buffer** so that even the footage available 2 seconds before detecting the anomaly is shown. It pushes frames to a queue upon detection.
- **File Writer Thread (Consumer):** Encodes video using FFmpeg for smaller video size and codec format, writing MP4 clips to disk fully asynchronously in a different thread.

This design ensures no frame drops during save operations, deterministic latency, continuous real-time detection, and robust operation on low-I/O bandwidth storage.

---

3) System Architecture & Implementation

The deployed system performs the following steps:

1. **Capture** video streams in versatile modes (USB webcam / rpicamera0 / MP4 input).
2. **Run** YOLOv8n inference using NCNN (FP16).
3. **Filter** anomaly classes: potholes, speed bumps, obstacles.
4. **Maintain** rolling RAM buffer (pre-event frames).
5. **Trigger** event logging on detection.
6. **Asynchronously** encode and store MP4 clips.
7. **Log** timestamp, class, confidence, and clip path to CSV.

This ensures evidence-preserving anomaly detection, where the cause of the anomaly is captured before the event itself.

---

4) Performance & Thermal Benchmarking

Test Conditions: Raspberry Pi 5 (8GB), 640×480 resolution, Active Cooling, 64-bit OS.

- **Inference Speed:** Stable **9.0 FPS** (Average).
- **Thermal Stability:** Peak CPU temperature recorded at **62°C** under continuous load.

---

5) Summary & Evaluation

- **Technical Correctness:** The NCNN FP16 pipeline achieves stable real-time inference with correct class filtering, bounding-box overlay, CSV telemetry logging, and synchronized video evidence generation.
- **Use of ARM Tools:** The project explicitly targets ARM Cortex-A76 by leveraging **ARM NEON SIMD** acceleration, **FP16** compute paths, and **NCNN ARM-optimized kernels**.
- **Novelty:** The **2-second pre-event buffering system** captures the cause of anomalies — solving a critical limitation in conventional dashcam and black-box systems that only record after an event occurs.
- **Documentation & Demo Readiness:** Includes a structured project repository, reproducible setup instructions, logged telemetry (CSV), timestamped video evidence, and real-time on-screen FPS overlay.

6) Conclusion

This project demonstrates that high-performance real-time computer vision is achievable on CPU-only ARM SoCs using the right combination of efficient model architecture (YOLOv8n), embedded inference runtime (NCNN), quantization (FP16), and multi-threaded design. The system is suitable for deployment in low-cost ADAS prototypes, smart city monitoring nodes, and edge AI safety systems.

7) References

1. Dataset used for training yolov8n model https://www.kaggle.com/datasets/rohitsuresh15/radroad-anomaly-detection/
2. Yolov8n ultralytics documentation
3. opencv documentation
4. ffmpeg
5. tencent ncnn