PS-3:Road Pi: Real-Time Road Hazard Detection Using YOLOv5 Nano on ARM-Based Edge Devices

AMRITA VISHWA VIDYAPEETHAM

Chennai

BY: SANA SAH ,KATIRVELAVAN ,SELVANILA.S

MENTOR: Dr. Rahul S G

Email: selvanila@gmail.com

Abstract:

Road safety remains a major challenge due to the delayed detection of hazards such as potholes, debris, and unexpected obstacles. Cloud-based monitoring systems often introduce latency and depend on stable internet connectivity, making them unsuitable for real-time edge applications. This paper presents RoadPi, a lightweight real-time road hazard detection system deployed on a Raspberry Pi using YOLOv5 Nano and ONNX Runtime. The system performs inference entirely on ARM-based CPU hardware without GPU acceleration. A reduced 320×320 input resolution and optimized inference pipeline enable efficient processing on resource-constrained hardware. Custom Non-Maximum Suppression (NMS), confidence filtering, rate-limited disk logging, and JPEG compression techniques ensure computational efficiency and storage optimization. The system streams detection results through a Flask-based web interface while logging detection metadata in CSV format. Experimental results demonstrate stable real-time performance, validating the feasibility of embedded edge AI deployment for intelligent transportation systems.

Keywords:

Edge AI, YOLOv5 Nano, Raspberry Pi, ONNX Runtime, ARM SoC, Embedded Vision, Real-Time Detection

Introduction:

Road hazard detection is critical for improving transportation safety and enabling intelligent monitoring systems. Traditional surveillance systems rely heavily on centralized cloud processing, which introduces latency and increases dependency on network infrastructure. In contrast, Edge AI systems allow real-time inference directly on embedded hardware devices. Deploying object detection models on ARM-based systems presents computational challenges due to limited processing power and memory constraints. This work proposes a lightweight real-time detection system, RoadPi, deployed entirely on a Raspberry Pi using ONNX Runtime and YOLOv5 Nano. The objective is to achieve stable real-time detection while maintaining low computational overhead and efficient storage utilization.

Related work:

YOLO (You Only Look Once) models have been widely adopted for real-time object detection due to their single-stage detection architecture and high inference speed. Recent advancements focus on lightweight variants such as YOLOv5 Nano to support embedded deployment. ONNX Runtime has emerged as a cross-platform inference engine optimized for CPU and hardware accelerators. Previous studies demonstrate the feasibility of deploying deep learning models on ARM-based systems; however, optimization strategies remain crucial for maintaining real-time performance.

System Architecture:

The RoadPi system consists of five main components: (1) Raspberry Pi hardware, (2) Camera module for image acquisition, (3) YOLOv5 Nano ONNX model for inference, (4) Post-processing module implementing confidence filtering and NMS, and (5) Flask-based web streaming server. Captured frames are resized to 320×320 resolution and normalized before being passed to the ONNX inference session. Post-processed detection results are drawn onto frames and streamed in real-time through a web interface accessible via browser.

Methodology:

A. Image Acquisition: Frames are captured using Picamera2 configured at 320×320 RGB resolution to reduce computational overhead.

B. Preprocessing: The captured frame is normalized by dividing pixel values by 255. The image tensor is reshaped from HWC to CHW format and expanded with a batch dimension to match model input requirements.

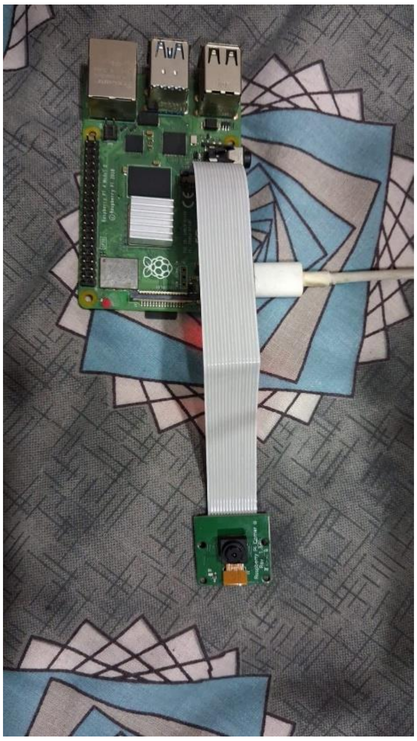
C. Model Inference: YOLOv5 Nano is exported to ONNX format and executed using ONNX Runtime with CPU Execution Provider. The output tensor contains bounding box coordinates, confidence scores, and class indices.

D. Post-Processing: Detections are filtered based on a confidence threshold (0.40). A custom Intersection-over-Union (IoU) function is used to compute overlap between bounding boxes, followed by Non-Maximum Suppression to remove redundant detections.

E. Logging and Storage: When detections occur, frames are saved with timestamp-based filenames. Metadata including bounding box coordinates, confidence score, and detection count are appended to a CSV file. Rate limiting ensures only one frame is saved per second to minimize disk I/O.

Hardware Implementation:

The system operates on a Raspberry Pi powered by ARM architecture. All inference is performed on CPU without GPU acceleration. The YOLOv5 Nano model is selected due to its reduced parameter size and computational efficiency. Memory usage and CPU load are carefully managed to maintain stable performance under continuous real-time operation.



Optimization Techniques:

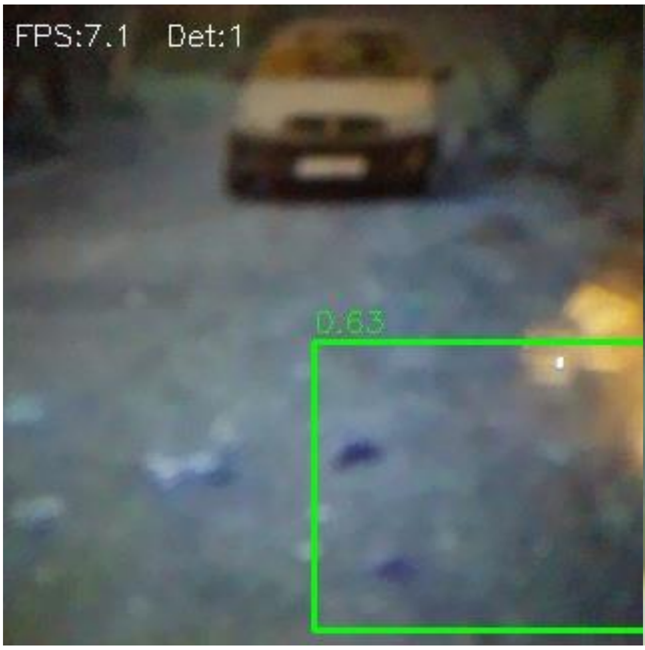
Several optimization techniques were implemented to ensure efficient embedded deployment:

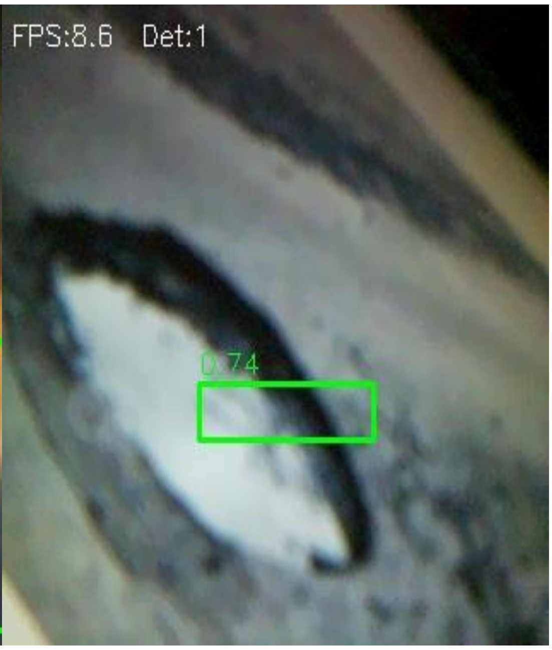
- Reduced input resolution (320×320) to lower computation.
- Lightweight YOLOv5 Nano architecture.
- CPU-only ONNX Runtime inference.
- Custom lightweight Non-Maximum Suppression implementation.
- JPEG compression (quality 70) for reduced storage usage.
- Threaded Flask server for non-blocking streaming.
- Rate-limited image saving to minimize disk overhead.

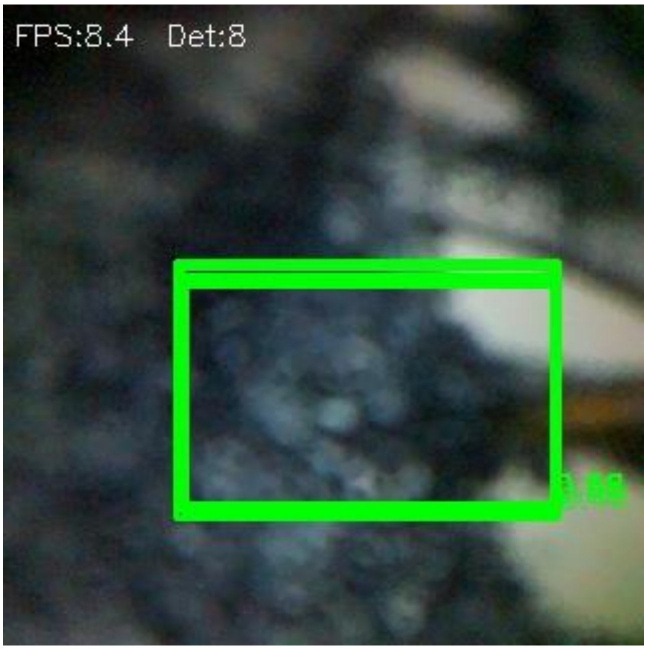
These optimizations collectively enable stable real-time detection performance on resource-constrained hardware.

Experimental Results:

Experimental evaluation demonstrates stable frame processing under CPU-only execution. Average FPS remains consistent during continuous operation. Detection metadata is successfully logged in CSV format, and saved images confirm bounding box accuracy. The system maintains reliable performance suitable for edge-based road monitoring applications.







Discussion:

While the system achieves real-time detection, performance may vary under extreme lighting conditions. Scaling to multi-class detection or higher resolutions may increase computational load. Further hardware acceleration could significantly enhance throughput.

Future Scope:

Future improvements include INT8 quantization for further model compression, integration with hardware accelerators (NPU), multi-class hazard detection expansion, GPS-based tagging, and cloud synchronization for centralized analytics.

Conclusion:

Road Pi demonstrates the feasibility of deploying real-time object detection systems on low-cost ARM-based edge devices. Through careful optimization and lightweight model selection, stable embedded AI inference is achieved. The proposed system serves as a scalable foundation for intelligent road safety monitoring solutions.

References:

[1] J. Redmon et al., 'YOLOv3: An Incremental Improvement,' 2018.
[2] Ultralytics, 'YOLOv5 Documentation.'
[3] Microsoft, 'ONNX Runtime Documentation.'
[4] Raspberry Pi Foundation, Official Documentation.