Edge AI Benchmark Suite PRD

Product Name

Edge AI Benchmark Suite (working title)

Overview

The Edge AI Benchmark Suite is a standardized, automated benchmarking framework designed to evaluate and compare AI inference capabilities across popular edge AI platforms. The initial scope targets the NVIDIA Jetson Nano Developer Kit, Raspberry Pi with AI HAT+, and Raspberry Pi with AI HAT+ 2. The suite focuses on two high-value workloads for edge developers: real-time computer vision using YOLO and local large language model (LLM) inference using Ollama.

The benchmark is designed to be reproducible, transparent, and extensible, producing publication-quality metrics, tables, and visualizations suitable for technical evaluation and showcase.

Target Audience

- Edge AI developers
- Embedded systems engineers
- Developers evaluating hardware trade-offs for on-device AI

Goals

- Provide a fair, repeatable comparison of AI performance across edge platforms
- Measure multiple performance dimensions beyond raw speed
- Automate setup, execution, and result collection
- Produce clear visual outputs (graphs, tables, dashboard)
- Serve as a reference benchmark developers can rerun or extend

Non-Goals

- Training or fine-tuning models
- Cloud-based inference benchmarking
- Benchmarking non-AI workloads
- Providing purchasing recommendations or pricing analysis

Supported Hardware Platforms (v1)

- NVIDIA Jetson Nano Developer Kit
- Raspberry Pi + AI HAT+
- Raspberry Pi + AI HAT+ 2

Each benchmark run will explicitly record: - CPU model - Accelerator (GPU / NPU) - RAM size - Storage type (SD / SSD) - Cooling configuration - Power mode / performance profile - OS version and kernel

Benchmark Workloads

1. Computer Vision: Ultralytics YOLO Benchmarks

The benchmark suite will evaluate the full official Ultralytics YOLO model families across three major generations, covering multiple computer vision task types.

Supported YOLO Versions and Tasks

YOLOv8

- Detection: yolov8n.pt, yolov8s.pt, yolov8m.pt, yolov8l.pt, yolov8x.pt
- Instance Segmentation: yolov8n-seg.pt, yolov8s-seg.pt, yolov8m-seg.pt, yolov8l-seg.pt, yolov8x-seg.pt
- Pose / Keypoints: yolov8n-pose.pt, yolov8s-pose.pt, yolov8m-pose.pt, yolov8l-pose.pt, yolov8x-pose.pt, yolov8x-pose-p6.pt
- Oriented Bounding Boxes (OBB): yolov8n-obb.pt, yolov8s-obb.pt, yolov8m-obb.pt, yolov8l-obb.pt, yolov8x-obb.pt
- Classification: yolov8n-cls.pt, yolov8s-cls.pt, yolov8m-cls.pt, yolov8l-cls.pt, yolov8x-cls.pt

YOLOv11

- Detection: yolo11n.pt, yolo11s.pt, yolo11m.pt, yolo11l.pt, yolo11x.pt
- Instance Segmentation: yolo11n-seg.pt, yolo11s-seg.pt, yolo11m-seg.pt, yolo11l-seg.pt, yolo11x-seg.pt
- Pose / Keypoints: yolo11n-pose.pt, yolo11s-pose.pt, yolo11m-pose.pt, yolo11l-pose.pt, yolo11x-pose.pt
- Oriented Bounding Boxes (OBB): yolo11n-obb.pt, yolo11s-obb.pt, yolo11m-obb.pt, yolo11l-obb.pt, yolo11x-obb.pt
- Classification: yolo11n-cls.pt, yolo11s-cls.pt, yolo11m-cls.pt, yolo11l-cls.pt, yolo11x-cls.pt

YOLOv26

- Detection: yolo26n.pt, yolo26s.pt, yolo26m.pt, yolo26l.pt, yolo26x.pt
- Instance Segmentation: yolo26n-seg.pt, yolo26s-seg.pt, yolo26m-seg.pt, yolo26l-seg.pt, yolo26x-seg.pt
- Pose / Keypoints: yolo26n-pose.pt, yolo26s-pose.pt, yolo26m-pose.pt, yolo26l-pose.pt, yolo26x-pose.pt
- Oriented Bounding Boxes (OBB): yolo26n-obb.pt, yolo26s-obb.pt, yolo26m-obb.pt, yolo26l-obb.pt, yolo26x-obb.pt
- Classification: yolo26n-cls.pt, yolo26s-cls.pt, yolo26m-cls.pt, yolo26l-cls.pt, yolo26x-cls.pt

All models will be evaluated strictly in inference mode using official pretrained weights.

Dataset

- Default Ultralytics-provided validation/test datasets per task type
- Fixed input resolution per model family and task
- Identical preprocessing and postprocessing across all platforms

Metrics

- Latency (ms): First inference latency, P50, P95
- Throughput: FPS
- Accuracy: mAP / precision metrics as provided by Ultralytics
- Resource utilization: CPU %, accelerator %, memory usage
- Power consumption: idle and inference power (if supported)

2. Local LLM Inference (Ollama)

Model Groups

- **7B**: llama2:7b, mistral:7b, olmo2:7b
- **8B**: llama3.1:8b, dolphin3:8b, dolphin-llama3:8b
- **9B**: gemma2:9b

Versions, hashes, and quantization formats will be recorded to ensure reproducibility.

Prompts

- Fixed, version-controlled prompt set
- Identical prompts across all platforms and models
- Deterministic generation settings (temperature, top-p, top-k)

Metrics

- Time to first token (TTFT)
- Tokens per second (throughput)
- Total completion latency
- Output token count consistency
- Resource utilization: CPU %, accelerator %, memory usage
- Power consumption (if supported)

Accuracy will not be semantically scored; consistency and performance metrics are the focus.

Benchmark Methodology

- Warm-up runs: 3 iterations per test (excluded from results)
- Measured runs: 10 iterations per test
- Aggregation: mean, standard deviation, min/max
- Background services minimized
- Stable thermal conditions enforced

Environment Setup & Installation

- Fully automated, native execution
- Python virtual environment for isolation
- Version-pinned dependencies
- Ollama installation automated
- Idempotent and reproducible scripts

Result Collection & Reporting

- Structured JSON and CSV per run
- Aggregation script produces summary CSVs
- Auto-generated dashboard (HTML / notebook) with charts and tables
- Raw and aggregated data downloadable

Extensibility

- Modular workloads
- Easy addition of new models or platforms
- Clear interface for adding benchmarks

Comparison Dashboard Design

- Static dashboard (HTML or notebook)
- Global filters: platform, workload, YOLO version, task, model size, LLM group
- Sections: system overview, YOLO performance, YOLO scaling, LLM performance, LLM efficiency, stability/variance, raw data access
- Charts: latency, throughput, accuracy, efficiency, box plots for variance
- Visual principles: neutral palette, one color per platform, clear axes and units

Success Criteria

- Benchmarks run end-to-end with a single command per platform
- Results are reproducible across multiple runs
- Clear performance differences observable across platforms
- Developers can easily rerun or extend the benchmark suite