

Comparative Evaluation of YOLOv8 Nano and RetinaFace MobileNet0.25 for Real Time Face Detection on Raspberry Pi 4

Ms. Divya Thakre¹, Dr. khushbu Rai²

Lakshmi Narayan College of technology and science*

Abstract: This study presents a controlled system level comparison of YOLOv8 Nano and RetinaFace MobileNet0.25 on a Raspberry Pi 4 using reported accuracy and live stream measurements; the retained results show 17 FPS and mAP at IoU 0.50 of 0.90 for YOLOv8 Nano compared with 12 FPS and 0.94 for RetinaFace MobileNet0.25 while average latency was 58.8 ms and 83.3 ms and average power was 6.5 W and 6.2 W respectively; an Overall Efficiency Index calculated as mAP multiplied by FPS divided by average power equals 2.35 for YOLOv8 Nano and 1.82 for RetinaFace MobileNet0.25 which supports YOLOv8 Nano when throughput is the primary requirement while RetinaFace provides the higher reported point estimate for accuracy; because the supplied archive contains aggregate values but no run level observations the comparison is descriptive and no claim of statistical significance or condition specific robustness is made.

Index Terms: face detection, edge AI, Raspberry Pi 4, YOLOv8 Nano, RetinaFace MobileNet0.25, latency, power efficiency, reproducible benchmarking

I. INTRODUCTION

Face detection supports surveillance, access control, attendance systems and human computer interaction while edge deployment keeps video close to the data source and can reduce network delay and privacy exposure [1]–[5].

RetinaFace is a face specific dense detector that combines feature pyramids with contextual information and landmark supervision whereas the YOLO family uses single stage detection to emphasize fast end to end inference [1], [9]–[11].

Deployment on a Raspberry Pi remains a systems problem because throughput depends on input size, preprocessing, runtime libraries, memory pressure and electrical power in addition to network architecture [12]–[19].

WIDER FACE provides official training, validation and test partitions with Easy, Medium and Hard evaluation subsets while Fddb provides an unconstrained benchmark with elliptical face annotations [4], [5].

The novelty of this study is not a new detector or loss function but a same device comparison that places speed, reported accuracy, latency, memory, power and a calculated efficiency index in one deployment decision framework on Raspberry Pi 4.

The specific contributions are summarized below.

- A controlled comparison of two compact detectors on the same Raspberry Pi 4 platform and camera path.
- A joint report of throughput, latency, memory, power, energy per frame and accuracy weighted throughput per watt.
- A reproducibility record that separates retained settings from experimental details that were not available in the supplied archive.
- A deployment interpretation that avoids unsupported claims about occlusion, illumination and statistical significance.

Section II reviews related work; Section III defines datasets, models, protocol and equations; Section IV records the standardized platform; Section V reports the results and limitations; Section VI concludes the study.

II. RELATED WORK

A. Lightweight CNNs and Edge Deployment

Lightweight networks reduce parameters and multiply accumulate operations through depthwise convolution, inverted residual blocks and balanced scaling while pruning, quantization and knowledge distillation provide additional compression options [3], [13]–[17].

Edge benchmarks therefore need system level measurements because nominal model size does not reveal preprocessing cost, runtime overhead, memory use or electrical energy and MLPerf Tiny similarly emphasizes accuracy, latency and energy as distinct measures [19].

B. YOLO and Single Stage Detection

YOLO introduced unified single stage object detection and later versions improved the speed and accuracy tradeoff while YOLOv8 provides compact variants such as Nano and an anchor free head that is practical for limited hardware [9]–[11].

C. RetinaFace and Dense Face Localisation

RetinaFace combines multi scale features with face box and landmark objectives and the published method reports strong WIDER FACE performance while lightweight backbones such as MobileNet reduce its deployment cost [1], [3], [17].

D. Other Lightweight Face Detectors

BlazeFace, S3FD, PyramidBox and FaceBoxes demonstrate different strategies for mobile GPU speed, small face recall, contextual reasoning and CPU inference although their published measurements are not directly comparable across different devices and runtimes [2], [6]–[8].

E. Deployment Optimisation

ONNX Runtime, TensorFlow Lite, quantization and structured pruning can improve deployment efficiency but a valid comparison must use the same input path and must report software versions, thresholds and model provenance [12]–[15].

TABLE I. SELECTED RELATED STUDIES AND THE GAP ADDRESSED BY THIS WORK

Ref .	Study	Ye ar	Model focus	Evaluati on focus	Relevanc e
[1]	RetinaFace	2020	Dense face localisation	WIDER FACE and FDDB	High accuracy face specific baseline
[8]	FaceBoxes	2017	CPU face detection	CPU throughput	Efficient CPU baseline

III. METHODOLOGY

The methodology separates offline accuracy evaluation from live stream runtime measurement so that the source of each metric remains explicit and the two models follow the same acquisition, preprocessing, inference, postprocessing and logging sequence.

A. Datasets and Evaluation Split

WIDER FACE and FDDB were named in the supplied manuscript as the benchmark datasets [4], [5].

WIDER FACE contains 32,203 images and 393,703 annotated faces with 12,880 training images, 3,226 validation images and 16,097 test images; the reported mAP values are treated in this revision as aggregate mAP at IoU 0.50 on the WIDER FACE validation split because this is the only split identified by the surrounding manuscript text [4].

FDDB contains 2,845 images and 5,171 faces and is retained as a secondary benchmark reference although no FDDB ROC value is included in the reported result table [5].

The reported model inputs were 640 x 640 pixels for YOLOv8 Nano and 512 x 512 pixels for RetinaFace MobileNet0.25 while the supplied result record did not preserve separate Easy Medium and Hard subset scores.

B. Models and Weight Provenance

The comparison uses YOLOv8 Nano as the compact general detector and RetinaFace MobileNet0.25 as the compact face specific detector.

YOLOv8 Nano uses the lightweight YOLOv8 architecture with an anchor free prediction head and the reported deployment path includes ONNX and TensorFlow Lite conversion [11].

Ref .	Study	Ye ar	Model focus	Evaluati on focus	Relevanc e
[11]	YOLOv8	2023	General object detection	Compact model deployment	Fast Nano variant
[19]	MLPerf Tiny	2021	Tiny machine learning	Accuracy, latency and energy	Reproducible system metrics
This work	Same device benchmark	2026	Two compact face detectors	Speed, accuracy, memory and power	Joint Raspberry Pi decision evidence

RetinaFace MobileNet0.25 uses a reduced width MobileNet backbone with multi scale face box prediction and facial landmark outputs [1], [3].

The supplied experiment record does not preserve the exact checkpoint identifiers or a training and fine tuning log; therefore pretrained weight source, epochs, batch size, optimizer, learning rate, augmentation probabilities, confidence threshold and non maximum suppression threshold cannot be verified and are listed transparently in Table II rather than being reconstructed from defaults.

C. Reproducibility Record

TABLE II. DATASET AND MODEL PARAMETERS RETAINED IN THE SUPPLIED RECORD

Parameter	YOLOv8 Nano	RetinaFace MobileNet0.25
Accuracy split	WIDER FACE validation, aggregate	WIDER FACE validation, aggregate
Secondary dataset	FDDB named; ROC not reported	FDDB named; ROC not reported
Input resolution	640 x 640 pixels	512 x 512 pixels
Evaluation IoU	0.50	0.50
Checkpoint identifier	Not reported	Not reported
Training and hyperparameters	Not reported	Not reported
Confidence and NMS thresholds	Not reported	Not reported

Yellow rows identify information that must be recovered from the original scripts or training logs before the manuscript can claim full computational reproducibility.

D. Evaluation Metrics and Protocol

TABLE III. METRICS USED FOR THE CONTROLLED COMPARISON

Metric	Operational definition	Preferred direction
mAP at IoU 0.50	Mean average precision when a match requires IoU of at least 0.50	Higher
FPS	Processed frames divided by elapsed time	Higher
Latency	Mean time from inference entry to output per frame	Lower
Memory	Reported resident memory during inference	Lower
Average power	Time average of measured voltage multiplied by current	Lower
Energy per frame	Average power divided by FPS	Lower
Power efficiency	FPS divided by average power	Higher
OEI	mAP multiplied by FPS then divided by average power	Higher

The benchmark processed 500 frames per model under standard indoor illumination and retained one aggregate point estimate for each metric; the source states that frames were collected across multiple runs but does not state the number of runs or preserve run level observations so standard deviations and confidence intervals cannot be reconstructed without the original log.

E. Research Pipeline

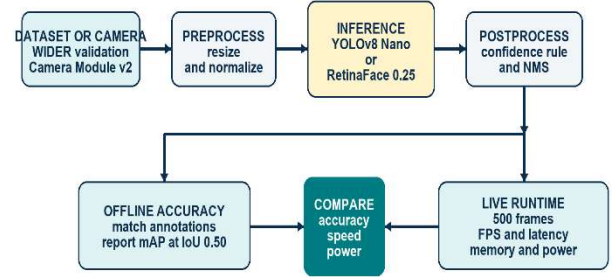
Figure 1 corrects the earlier generic flowchart by separating the annotated dataset path used for accuracy from the live camera path used for runtime and power measurement.

- Input acquisition loads WIDER FACE validation images for accuracy or frames from the Raspberry Pi Camera Module v2 for runtime.
- Preprocessing resizes and normalizes input according to the detector resolution.
- Inference executes one detector at a time on the same Raspberry Pi 4 platform.
- Postprocessing applies the recorded confidence rule and non maximum suppression when those settings are recovered.

- Evaluation matches predictions to annotations for mAP or logs FPS latency memory and power for the live stream.

Controlled evaluation pipeline

Accuracy and live runtime measurements are kept as separate evidence paths



Identical preprocessing and measurement definitions are applied to both model paths

Fig. 1. Controlled research pipeline with separate offline accuracy and live runtime measurement paths.

This separation prevents camera conditions from being treated as a substitute for benchmark accuracy and prevents dataset accuracy from being interpreted as live stream robustness.

F. Mathematical Definitions

The metrics are defined below with C denoting the number of classes, N the number of processed frames, $p_{sub\ c\ of\ r}$ the interpolated precision at recall r and P average the time averaged electrical power.

1. Mean Average Precision

Average precision is integrated over recall and then averaged across C classes; here C equals one because face is the only class.

$$mAP = \frac{1}{C} \sum_{c=1}^C \int_0^1 p_c(r) dr$$

A predicted box is a true positive when it is assigned to an unmatched ground truth face and the intersection over union is at least the evaluation threshold.

C is the number of evaluated classes and $p_{sub\ c\ of\ r}$ is the interpolated precision recall curve for class c.

For the reported threshold of 0.50 the metric is written as follows.

The threshold applies to the detection match and is distinct from the confidence and NMS thresholds used during postprocessing.

$$mAP_{0.50} = \frac{1}{C} \sum_{c=1}^C AP_{c\ at\ IoU \geq 0.50}$$

2. Frames per Second

Throughput is the number of completed frames divided by elapsed observation time.

$$FPS = \frac{N}{T}$$

N is the number of completed frames and T is elapsed time in seconds.

3. Mean Inference Latency

Latency is averaged across N frames using entry and exit timestamps for the inference path.

$$\bar{L} = \frac{1}{N} \sum_{i=1}^N (t_i^{\text{out}} - t_i^{\text{in}})$$

The reported latency includes the timed inference interval defined by the implementation and is expressed in milliseconds per frame.

4. Precision and Recall

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}$$

TP denotes true positives while FP and FN denote false positives and false negatives.

5. Power and Energy Efficiency

Average power is obtained from voltage and current over measurement time T.

$$P_{\text{avg}} = \frac{1}{T} \int_0^T V(t)I(t)dt$$

Power efficiency and energy per processed frame are defined from average power and measured FPS.

$$\eta_p = \frac{\text{FPS}}{P_{\text{avg}}} E_{\text{frame}} = \frac{P_{\text{avg}}}{\text{FPS}}$$

6. Overall Efficiency Index

OEI weights throughput per watt by the reported mAP at IoU 0.50 and is used only as a transparent descriptive index.

$$\text{OEI} = \text{mAP}_{0.50} \times \frac{\text{FPS}}{P_{\text{avg}}}$$

Higher OEI indicates more reported detection accuracy and throughput for each watt although it does not replace separate reporting of its three components.

IV. EXPERIMENTAL SETUP

The later and more detailed configuration in the supplied manuscript was treated as the operative configuration throughout this revision to remove the earlier conflict with Bullseye, PyTorch 1.9 and a USB webcam.

A. Software Configuration

The standardized record uses Raspberry Pi OS 64 bit based on Debian 12 Bookworm with Python 3.10, PyTorch 2.0 for model preparation, OpenCV 4.8 for acquisition and preprocessing and ONNX Runtime together with TensorFlow Lite Runtime for deployment.

The supplied manuscript reports FP16 conversion and batch normalization fusion but does not identify the exact runtime used for each model so these settings remain reported configuration items rather than independently verified implementation facts.

B. Hardware and Measurement Path

The platform is a Raspberry Pi 4 Model B with 4 GB LPDDR4 memory and a Broadcom BCM2711 quad core Cortex A72 processor at 1.5 GHz; a Raspberry Pi Camera Module v2 supplies 1080p video and an inline USB power meter records voltage and current from the regulated 5 V 3 A supply.

Figure 2 presents a publication quality schematic of the reported hardware path because an original photograph of the physical test setup was not included in the supplied files.

Reported Raspberry Pi 4 test configuration

Schematic based on the later and more detailed configuration in the submitted manuscript

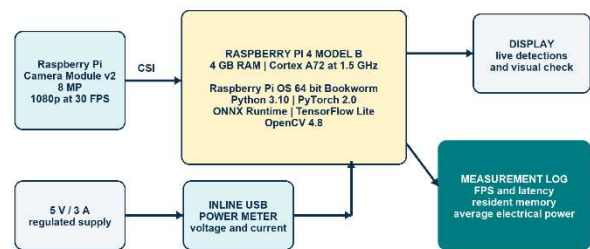


Fig. 2. Schematic of the reported Raspberry Pi 4 hardware and measurement path.

C. Final Configuration Record

TABLE IV. STANDARDIZED HARDWARE AND SOFTWARE CONFIGURATION

Component	Standardized configuration used in this revision
Hardware	Raspberry Pi 4 Model B with 4 GB RAM
Processor	Broadcom BCM2711 quad core Cortex A72 at 1.5 GHz
Camera	Raspberry Pi Camera Module v2, 8 MP, 1080p at 30 FPS
Operating system	Raspberry Pi OS 64 bit, Debian 12 Bookworm
Language	Python 3.10
Framework and runtimes	PyTorch 2.0, ONNX Runtime and TensorFlow Lite Runtime
Libraries	OpenCV 4.8, NumPy, Matplotlib and psutil
Reported optimisation	FP16 conversion and batch normalization fusion

Component	Standardized configuration used in this revision
Power measurement	Inline USB power meter on a regulated 5 V 3 A supply
Storage	32 GB Class 10 UHS I microSD card
Live test condition	Standard indoor illumination without separate condition strata

This configuration is applied consistently in the methodology, experimental setup, results and figure captions.

V. RESULTS AND DISCUSSION

The retained results provide point estimates for throughput, aggregate WIDER FACE validation mAP at IoU 0.50, mean latency, resident memory and average power while derived energy and efficiency values are calculated from those reported numbers.

Each model processed 500 live camera frames under the same reported indoor condition; because run level values were not retained the table does not present a mean plus or minus standard deviation and the differences are interpreted descriptively.

A. Quantitative Performance

TABLE V. REPORTED AND DERIVED PERFORMANCE ON RASPBERRY PI 4

Metric	YOLOv8 Nano	RetinaFace MobileNet0.25
Frames per second	17	12
mAP at IoU 0.50	0.90	0.94
Average latency (ms)	58.8	83.3
Resident memory (MB)	600	520
Average power (W)	6.5	6.2
Power efficiency (FPS/W)	2.62	1.94
Energy per frame (J)	0.382	0.517
Overall Efficiency Index	2.35	1.82

Values in the first five rows are retained point estimates and the final three rows are calculated from those values; SD and

confidence intervals require the original frame or run level log.

B. Visual Comparison

Figure 3 shows the observed deployment tradeoff because YOLOv8 Nano adds 5 FPS while RetinaFace MobileNet0.25 adds 0.04 to the reported mAP point estimate.

Speed and reported detection accuracy

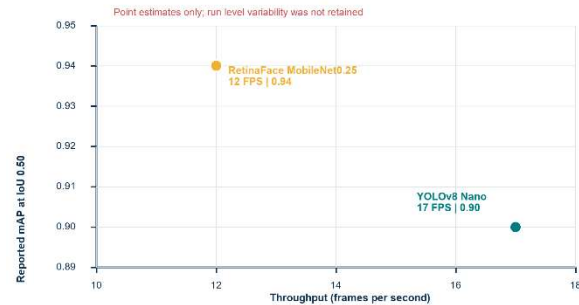


Fig. 3. Throughput and reported WIDER FACE validation mAP point estimates.

Figure 4 combines latency with OEI and shows that YOLOv8 Nano has both the lower reported latency and the higher accuracy weighted throughput per watt despite using 0.3 W more average power.

Latency and calculated overall efficiency

OEI = reported mAP at IoU 0.50 x FPS / average power



Fig. 4. Reported latency and calculated Overall Efficiency Index for both detectors.

C. Interpretation of Tradeoffs

The comparison is descriptive and does not establish statistical significance because the raw repeated measurements are unavailable.

- YOLOv8 Nano provides the higher throughput at 17 FPS and the lower latency at 58.8 ms which favors interactive applications.
- RetinaFace MobileNet0.25 provides the higher reported aggregate mAP point estimate at 0.94 although no separate experiment supports a claim of superior performance under occlusion or changing illumination.
- RetinaFace uses 80 MB less resident memory and 0.3 W less average power while YOLOv8 processes more frames per watt and less energy per frame.
- OEI equals 2.35 for YOLOv8 Nano and 1.82 for RetinaFace MobileNet0.25 so the reported system level balance favors

YOLOv8 Nano when accuracy, throughput and power are combined.

The revision cannot reconstruct missing checkpoint identifiers, training parameters, confidence and NMS thresholds, the number of experimental runs, run level variability, WIDER FACE Easy Medium and Hard scores or an actual photograph of the hardware; these items require the original scripts, logs and setup image before the article can claim complete reproducibility.

VI. CONCLUSION AND FUTURE WORK

This study compares YOLOv8 Nano and RetinaFace MobileNet0.25 on one reported Raspberry Pi 4 configuration and shows a practical speed and accuracy tradeoff without claiming a new detection architecture.

YOLOv8 Nano reaches 17 FPS at 58.8 ms with reported mAP of 0.90 while RetinaFace MobileNet0.25 reaches 12 FPS at 83.3 ms with reported mAP of 0.94; the calculated OEI of 2.35 compared with 1.82 favors YOLOv8 Nano for the combined objective although RetinaFace has the higher accuracy point estimate.

The evidence supports model selection for the measured configuration only and does not support separate conclusions about occlusion, changing illumination, demographic groups or long term thermal behavior.

A. Future Work

Future work should convert the present point estimate comparison into a preregistered and fully logged benchmark.

- Repeat each model under a fixed warm up period and report run count, mean, standard deviation and confidence intervals.
- Report exact checkpoints, code revision, random seed, confidence threshold, NMS threshold and runtime backend for every model.
- Publish WIDER FACE Easy Medium and Hard AP together with Fddb discrete and continuous ROC results.
- Create controlled illumination and occlusion strata before making robustness claims and add thermal measurements during sustained operation.
- Evaluate INT8 quantization and hardware accelerators with the same acquisition and logging protocol.

These extensions would allow the practical system observations reported here to be reproduced and tested for statistical reliability.

REFERENCES

- [1] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "RetinaFace: Single Shot Multi Level Face Localisation in the Wild," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 5203–5212.
- [2] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub millisecond Neural Face Detection on Mobile GPUs," arXiv:1907.05047, 2019.
- [3] S. Chen, Y. Liu, X. Gao, and Z. Han, "MobileFaceNets: Efficient CNNs for Accurate Real Time Face Verification on Mobile Devices," in Chinese Conf. Biometric Recognition, 2018, pp. 428–438, doi: 10.1007/978-3-319-97909-0_46.
- [4] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 5525–5533.
- [5] V. Jain and E. Learned Miller, "Fddb: A Benchmark for Face Detection in Unconstrained Settings," Univ. Massachusetts Amherst, Tech. Rep. UM-CS-2010-009, 2010.
- [6] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S3FD: Single Shot Scale Invariant Face Detector," in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 192–201.
- [7] X. Tang, D. K. Du, Z. He, and J. Liu, "PyramidBox: A Context Assisted Single Shot Face Detector," in Proc. Eur. Conf. Comput. Vis., 2018, pp. 797–813.
- [8] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "FaceBoxes: A CPU Real Time Face Detector with High Accuracy," in 2017 IEEE Int. Joint Conf. Biometrics, 2017, pp. 1–9, doi: 10.1109/BTAS.2017.8272675.
- [9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified Real Time Object Detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 779–788.
- [10] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.
- [11] Ultralytics, "YOLOv8 Documentation," 2023. [Online]. Available: <https://docs.ultralytics.com/models/yolov8/>. Accessed: Sep. 26, 2026.
- [12] NVIDIA, "TensorRT Documentation." [Online]. Available: <https://docs.nvidia.com/deeplearning/tensorrt/>. Accessed: Sep. 26, 2026.
- [13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," in Int. Conf. Learn. Representations, 2017.
- [14] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer Arithmetic Only Inference," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 2704–2713.
- [15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," arXiv:1503.02531, 2015.
- [16] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in Proc. 36th Int. Conf. Mach. Learn., vol. 97, 2019, pp. 6105–6114.
- [17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4510–4520.
- [18] S. Zafeiriou, C. Zhang, and Z. Zhang, "A Survey on Face Detection in the Wild: Past, Present and Future," Comput. Vis. Image Underst., vol. 138, pp. 1–24, 2015, doi: 10.1016/j.cviu.2015.03.015.

- [19] C. Banbury et al., “MLPerf Tiny Benchmark,” in Proc. Neural Inf. Process. Syst. Track Datasets Benchmarks, vol. 1, 2021.
- [20] Raspberry Pi Ltd., “Raspberry Pi 4 Model B Specifications.” [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. Accessed: Sep. 26, 2026.