Performance Analysis of Modern Object Detection Models for Edge-Based Assistive Glasses

Bekzhan Ryspekov[1]

Received September 8, 2025
Accepted February 1, 2026
Electronic access March 31, 2026

Smart assistive devices for Visually Impaired People (VIPs) rely on camera-based models to process information. Smart glasses for VIPs require a balance between speed (to avoid collisions) and accuracy (to understand what is in front of them). The problem is that researchers test object detection models on powerful server GPUs, whereas wearable assistive devices are usually supported on ARM chips (their architecture and power supplies are completely different from what a GPU can perform). Our research validated 11 object detection models (families of YOLOv8/v10/v11 and RT-DETR) locally on the M2 Chip, which is the most closely related to actual ARM chips. For the validation dataset, we used COCO 2017 ($N = 1000$). We evaluated the models based on three metrics: stability (Standard Deviation, P95), accuracy (mAP), and efficiency (FLOPs vs. Latency). The results have shown that RT-DETR demonstrated best balance between accuracy, speed, and Standard Deviation (54.7% mAP, high stability $\sigma \approx 77\ ms$); YOLOv8-Nano is seen as the fastest object detection model (latency of 210ms), but failed to detect small targets; new YOLO (v10/v11) showed high variance and poor optimization for Metal Performance Shaders (MPS) backend, despite having low FLOPs. The theoretical FLOPs don't guarantee low latency on Edge Devices. Prioritizing stability over raw FPS is critical for safety-critical applications, making RT-DETR the optimal choice, whereas YOLOv8-Nano remains suitable for general obstacle avoidance.

Keywords: Object Detection, Assistive Technology, Edge Computing, RT-DETR, YOLO, Visually Impaired Navigation, Inference Stability, Apple M2, Deep Learning.

Introduction

According to the World Health Organization, there are 253 million visually impaired people and 36 million people who are blind in the world[1]. Assistive technology for the visually impaired is critical for independence. Traditional tools, such as white canes and guide dogs, are reliable for detecting the presence of obstacles, but they fail at identifying them. Visually impaired people can mistakenly treat a hydrant as a chair because they misinterpret tactile feedback—they lack situational awareness. It's worth mentioning that the visually impaired often face moving objects, such as cars or bikes, which the white cane cannot reach from a distance. In contrast, smart assistive devices can resolve this problem by integrating cameras, sensors, and pretrained models, enabling visually impaired users to identify objects in front of them.

Object detection models (like YOLO) have become very accurate[2]. Most research benchmarks these models on powerful server GPUs (NVIDIA), which leads to misleading latency expectations. However, assistive devices use mobile ARM chips (like those in phones) that rely on batteries. The key difference between ARM-chips and others (x86-chips and discrete GPU) is fundamental[3]: ARM-chip prioritize energy efficiency, whereas others require huge amounts of power; due to less electricity supply and low temperatures ARM-chips do not require active cooling, analogous to the passive cooling of smart glasses, while others require active cooling; if the chip inside of the assistive device will overheat, it will decrease the efficiency of the chip so as to return to normal temperatures, providing leggings, so this will lead to wrong detections for the user, which is extremely dangerous. Therefore, this research conducts an experiment to determine whether the models considered good for detection indeed perform well on ARM architecture chips.

In neural networks, inference latency depends on arithmetic speed and memory bandwidth. The FLOPs metric considers only the arithmetic speed, disregarding memory bandwidth. Usually, conventional benchmarks measure FLOPs (floating-point operations) to estimate performance[4]. The research paper assumes that FLOPs are misleading on modern chips like Apple M2 (or other ARM chips with NPUs); it hypothesizes that stability (low jitter) and hardware-specific optimization (how well the model fits the chip) are more important for safety. Besides that, the amount of mathematical operations (layers) in neural networks cannot tell whether this model is

[1] *Maarif Education Complex, Kyrgyzstan*

optimized or not, fast or slow, since most of the inference latency is dependent on drivers of the device (in the case of experimentation of this research paper, it's Metal Performance Shader (MPS))[5]. The problem is that YOLOv10 uses new experimental layers (Attention or PSA), and when Apple MPS compares these layers against templates, it results in transferring computations from the fast GPU to the slow central processor (CPU)[5].

The goal of this research is to identify models best suited for assistive devices that balance accuracy and speed. Compared with others, we conduct a comparative analysis to establish a safety-critical baseline.

Literature Review

For assistive smart devices, since its inception, YOLO (You Only Look Once) has been the de facto standard, specifically for mobile devices (or edge devices), balancing accuracy and inference latency[6]. Object detection models are categorized into two types: anchor-based and anchor-free. The first type refers to models with anchors, where one cell can contain multiple anchors (e.g., 2 or more), and increases the volume of predictions per grid cell. Therefore, the more anchors in the model, the more complex it becomes; that is, more processing power is required to execute it. In contrast, anchor-free architectures directly predict object centers or keypoints, eliminating the need for predefined boxes, and offer superior efficiency by using fewer parameters[6]. The main problem is the post-processing step NMS (Non-Maximum Suppression), which removes redundant object bounding boxes (especially in anchor-based methods). The model generates thousands of boxes, and NMS must process them. If there is a car image with plenty of people on the street, NMS will be highly variable in latency and unpredictable. This situation is hazardous for VIP because it causes unacceptable latency spikes[7].

The publication of 'Attention is All You Need' has revolutionized the field of transformers since 2017[8]. The first real object detection transformer model, DETR, was proposed in 2020, and in 2024, one of the first complete, groundbreaking transformer models was released—RT-DETR. Compared with the YOLO family, RT-DETR does not require NMS. It uses the Hungarian Matching Algorithm for Bipartite Matching. New architecture of RT-DETR maintains fixed computational load, and became popular for its stability in inference latency (despite objects in the image it predicts with the same latency)[8,9].

The majority of research on deep learning benchmarks relies too heavily on FLOPs (floating-point operations per second), overlooking its limitations[10]. On hardware-constrained devices, latency depends on the Memory Access Cost (MAC). Transferring data from RAM to the processor consumes much more energy and time than arithmetic operations. Models like ShuffleNet or MobileNet show that low FLOPs don't guarantee low latency if the architecture consumes a lot of memory, resulting in high transfer overhead[11].

To integrate to the glasses, the models are compressed from FP32 (32 bits) to INT8 (8 bits). This is a standard for Apple CoreML and TensorFlow Lite[11]. Glasses don't have active cooling. Heavy computations cause thermal throttling—the chip intentionally increases latency to avoid exceeding safe operating temperatures. Tests on servers with air conditioners do not show the accurate reflection of 'jitter' latency, which occurs when the glasses overheat[12].

The efficiency of neural networks depends on how well their layers are supported by the backend. While NVIDIA's CUDA offers a wide range of supported libraries, Apple's MPS (Metal Performance Shaders) offers significantly narrower scope, which identifies them as limited operator support at library usage. If the new model (e.g., YOLOv10) uses operations not supported by the MPS library, the system switches to the CPU (a fallback)[13,14]. This causes high inference latency. Furthermore, while Apple's Unified Memory Architecture (UMA) on the M2 chip eliminates data copying between CPU and GPU, it introduces synchronization overhead; managing memory consistency between processors consumes computational resources, which can degrade real-time performance[15–17].

Currently, there are many devices (OrCam, Envision) that rely on cloud-based infrastructure, which results in increased internet connection latency. Alternatively, there are outdated, low-performance models (such as SSD-MobileNet) that cannot deliver the speed and accuracy required for reliable, secure VIP navigation, especially when walking outdoors[18].

New models (YOLO, RT-DETR) are predominantly evaluated on server GPUs (NVIDIA). There is a scarcity of empirical data about how they behave on ARM-based Edge Architectures[19]. In this research, the Apple M-series chip is used as a proxy of a high-performance Edge platform. Its architecture (Unified Memory, ARM) closely mirrors the hardware architecture of modern wearable devices[20]. The goal is to fill the gap by comparing the stability (not just FPS) of CNNs and Transformers to provide recommendations for choosing architectures for powerful edge devices[21].

Methods

Experiments were conducted on an Apple MacBook Air M2. This chip was chosen as a representative proxy for modern edge-devices. It has ARM architecture, NPU (Neural Engine), and Unified Memory, similar to powerful future smart glasses.

We used PyTorch, a deep learning framework, and MPS (Metal Performance Shaders, Apple's GPU-accelerated framework) to simulate real-world deployments on iOS/MacOS devices.

This study compares several models: CNN-based (YOLOv8/v10/v11 or anchor-based) and transformer-based (RT-DETR or anchor-free). We are comparing the classical approach with NMS (YOLO) and the end-to-end approach (RT-DETR) to determine which is more stable in terms of inference latency.

The validation images were sourced from the COCO 2017 dataset. The random sample consists of 1000 images. We used this volume of images to achieve statistical significance and exclude statistical outliers in comparison to limited anecdotal samples.

The dataset spans a complexity spectrum, ranging from 1 object (Low Density) to crowds (High Density). This enables the analysis of correlation analysis between the number of objects and inference latency overhead.

The key metrics are latency (ms) and FPS. For the visually impaired, predictability—avoiding unexpected freezes—matters more than low latency.

To simulate real-time video stream processing, we used a batch size of 1. We used the standardized input resolution. Before the actual test, we ran warm-up iterations to stabilize the cache and frequency.

In addition, the experimentation categorized the COCO 2017 dataset's images depending on the number of objects in a frame: low density: 1–5 objects (minimal load), medium density: 6-20 objects (typical street scenario), high density: $>$20 objects (crowded environments). This categorization allows us to measure the importance of the number of objects in the image.

Results

After running a 1000-picture validation set from the COCO 2017 dataset, the benchmark shows that an RT-DETR-L model achieves a Mean Average Precision (mAP) of 54.71% (Figure 1) at a competitive speed of 331.6ms (Table 1). Looking at other models, the absolute leader by speed is YOLOv8-Nano, with a speed of 210ms (Table 1). This model outperforms others; however, it sacrifices accuracy. Comparing these two models, it is clear that there is a trade-off: while achieving high speed, some metrics are lower, and sacrificing precision for speed, whereas RT-DETR prioritizes accuracy at the cost of higher latency.

This study analyzes not only average speed but also Standard Deviation (σ), which shows how predictable the model is in terms of the time to detection, and the 95th Percentile, which shows even rare latency outliers the model makes despite a good mean value. In the example of YOLO v10/11, YOLOv11-Medium demonstrated a standard deviation (σ) of about 5000 ms (Figure 2); this means the model processes some batches of pictures, for example, 5-10 seconds, and others may take much longer, resulting in severe performance jitter. In the view of RT-DETR, the "L" version has the lowest Standard Deviation of 77 ms (Figure 2). That is, the model is very reliable.

A significant anomaly was observed between YOLOv10 and RT-DETR: YOLOv10's FLOPs are low, yet its latency is very high, which is counterintuitive since they should be directly proportional. The FLOP count in RT-DETR is high; however, the latency is low. The reason for that discrepancy is that the Apple M2 chip has a dedicated hardware accelerator, AMX (Apple Matrix Coprocessor), that enables the M2 to process large matrices very quickly, and the architecture of RT-DETR is ideally suited to the AMX. Modern YOLO uses Depthwise Separable Convolutions to decrease FLOPs, but these operations divide computations into many small pieces. Therefore, AMX cannot effectively accelerate these fragmented computations, which is why they are processed on slower GPU cores (Figure 4).

Our analysis focused on the classes most critical for VIP to sense the environment (e.g., Person, Car, Traffic Light, Chair). Even with large objects, such as a car or a bus, the lightest model, YOLOv8-Nano, achieved $\approx$ 60.0% AP (Table 2). The difference with others is minimal; however, in detecting complex situations, the low-parameter model drops to just 36.3% AP (Table 2), whereas RT-DETR-L achieves 72.7% AP (Table 2). For VIP's safety, compromising accuracy for marginal speed gains is unacceptable in safety-critical applications, as it's always important to detect objects correctly.

Discussion

Analysis of the results reveals that YOLOv8/v10 has significantly greater latency variability (Latency Std $>$ 2000ms), whereas RT-DETR has the lowest ($\sim$70-90 ms). The reason for this phenomenon is NMS (Non-Maximum Suppression): in high-density crowd scenarios YOLO generates thousands of bounding boxes, creating a computational bottleneck as the processor attempts to filter them; RT-DETR operates End-to-End (without NMS), because of this, its inference time remains invariant to object count, yielding results in a fixed amount of time. For the visually impaired person, RT-DETR is safer, due to predictability, even if FPS is lower.

YOLO is optimised for NVIDIA (CUDA) and, on Apple MPS, many operations (especially postprocessing) fall back to the CPU. This confirms our hypothesis about the "Software-Hardware Gap". RT-DETR, based on matrix multiplication (Transformer), performs much better on Apple NPU/GPU than YOLO's logic operations.

In the original research paper, YOLOv8 is reported to achieve 200+ FPS on an NVIDIA A100. In our empirical evaluation on the Apple M2, we observed $\sim$2-3 FPS on the big models. This shows the gap between"academic" results (in ideal server conditions) and "real" results (on wearable de-

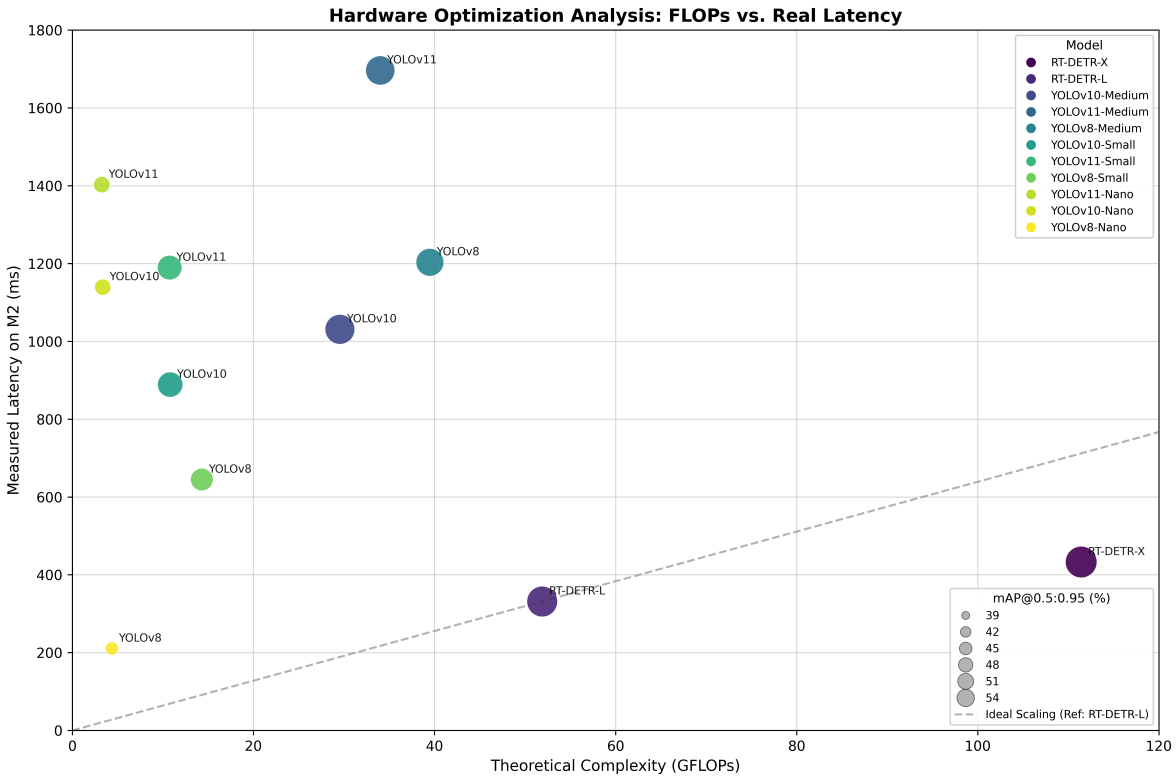

Fig. 1 Latency vs. Accuracy trade-off analysis. The chart visualizes the Pareto frontier, highlighting RT-DETR-L (top-left) as the optimal balance for performance and YOLOv8-Nano (bottom-left) as the fastest solution

Table 1 Comprehensive performance benchmark of eleven object detection models evaluated on the Apple M2 silicon (MPS backend). Metrics include Mean Average Precision (mAP) on the COCO 2017 validation subset ($N = 1,000$), temporal stability statistics (Standard Deviation,P_{95}), 95% Confidence Interval for the mean latency (using the formula $1.96 \times \frac{\sigma}{\sqrt{N}}$), and computational complexity (Parameters, FLOPs).

Model	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Latency (ms)	Latency Std (ms)	P95 Latency (ms)	95% CI	FPS	Params (M)	FLOPs (G)
RT-DETR-X	73.77	55.89	432.88	95.89	529.14	±5.9	2.31	65.63	111.41
RT-DETR-L	72.66	54.71	331.60	77.13	385.07	±4.8	3.02	32.15	51.89
YOLOv10-Medium	70.46	53.56	1031.21	2417.49	1980.71	±149.8	0.97	15.36	29.55
YOLOv11-Medium	70.21	52.87	1696.29	4995.98	5245.69	±309.7	0.59	20.09	33.99
YOLOv8-Medium	68.38	51.17	1203.38	2001.32	2085.12	±124.1	0.83	25.89	39.47
YOLOv10-Small	65.15	48.35	888.80	1434.21	1964.14	±88.9	1.13	7.25	10.79
YOLOv11-Small	65.21	47.66	1189.57	2413.40	2386.24	±149.6	0.84	9.44	10.74
YOLOv8-Small	63.12	45.69	644.92	599.35	1677.04	±37.2	1.55	11.16	14.30
YOLOv11-Nano	56.92	40.61	1402.75	2830.17	3010.19	±175.4	0.71	2.62	3.24
YOLOv10-Nano	55.78	40.51	1139.55	2339.88	2395.72	±145.0	0.88	2.30	3.35
YOLOv8-Nano	54.59	38.69	210.67	123.37	436.97	±7.6	4.75	3.15	4.37

vices without cooling). This demonstrates that it is misleading to judge them only by the perfect conditions.

If an average person walks at 1.4 m/s and the model freezes for 2 seconds (as in YOLOv10 in our test with Std 2400 ms), the visually impaired will traverse a distance of 2.8 meters. This distance is sufficient to inadvertently enter a hazardous zone. Therefore, the stability (low jitter) is more significant than high average FPS.

Limitations

We acknowledge that we focused exclusively on the Apple M2 architecture. Regarding cross-platform generalizability, the results observed on M2 may not directly extrapolate to other hardware platforms. Furthermore, the validation set was a limited snapshot (only 1000 images), whereas in real-world deployment scenarios, wearable devices must sustain opera-

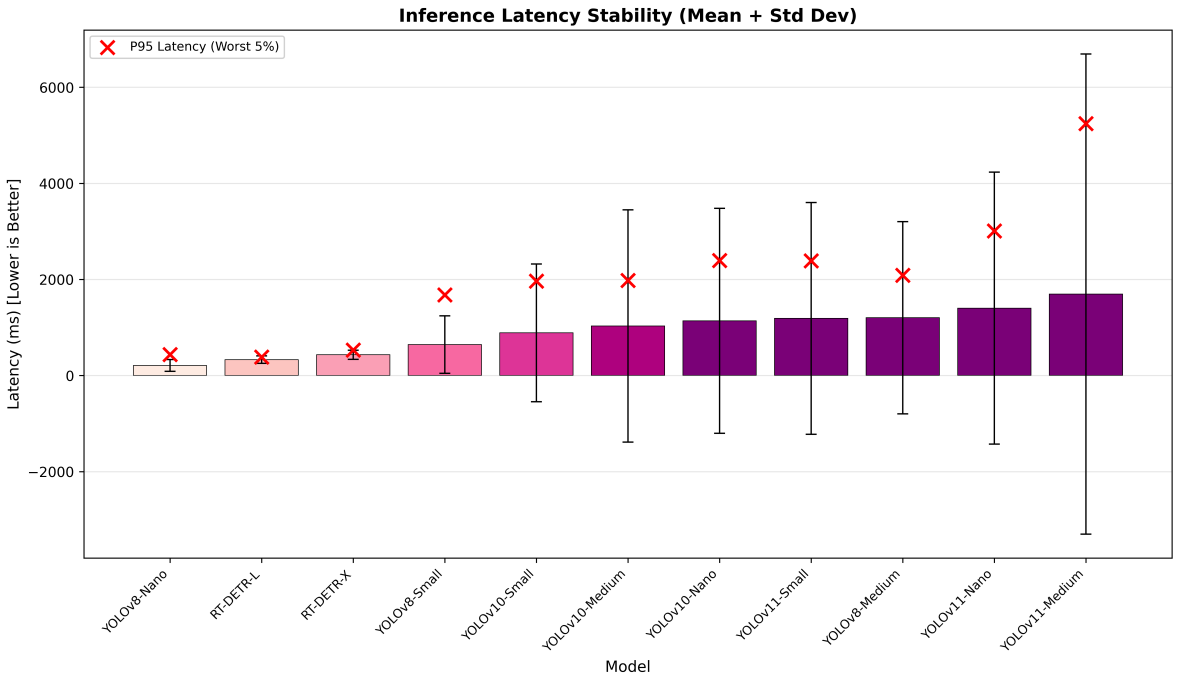

Fig. 2 Inference stability analysis. Bars represent mean latency, while error bars denote Standard Deviation (σ), and red 'X' markers indicate the 95th percentile (P_{95}) worst-case latency. Note the high variance in YOLOv11-Med compared to the stable RT-DETR-L.

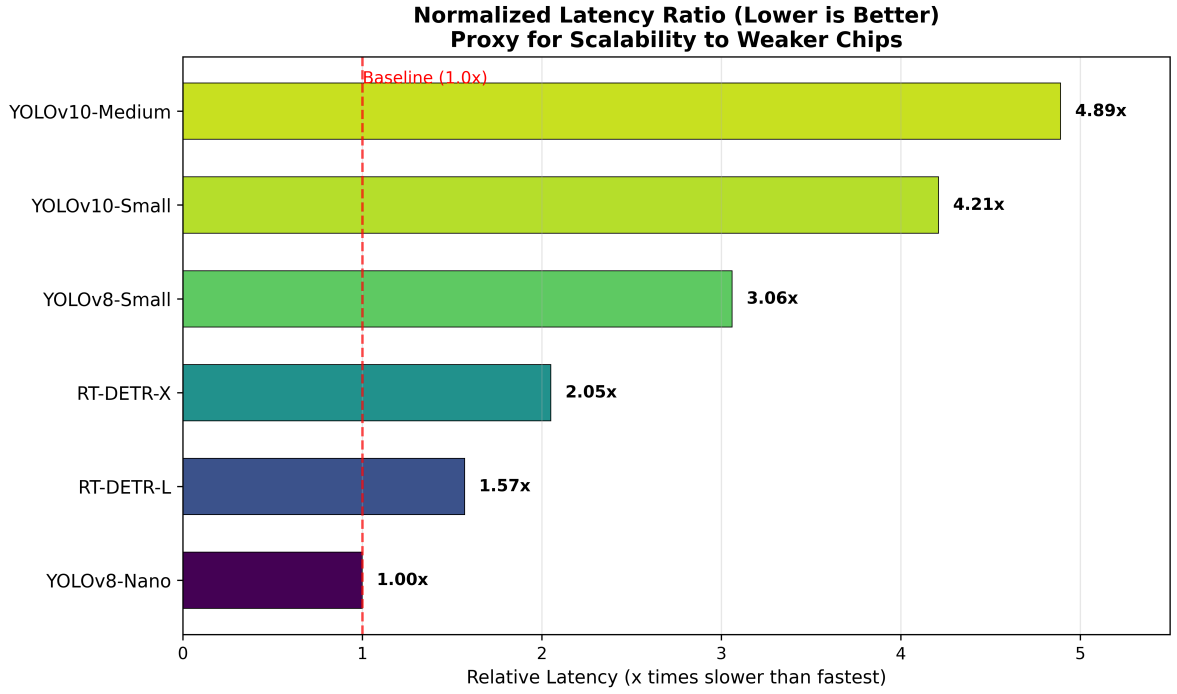

Fig. 3 Normalized inference latency ratios relative to the fastest baseline (YOLOv8-Nano). This metric serves as a scalability proxy, indicating that relative performance rankings are likely to persist across different ARM-based hardware platforms.

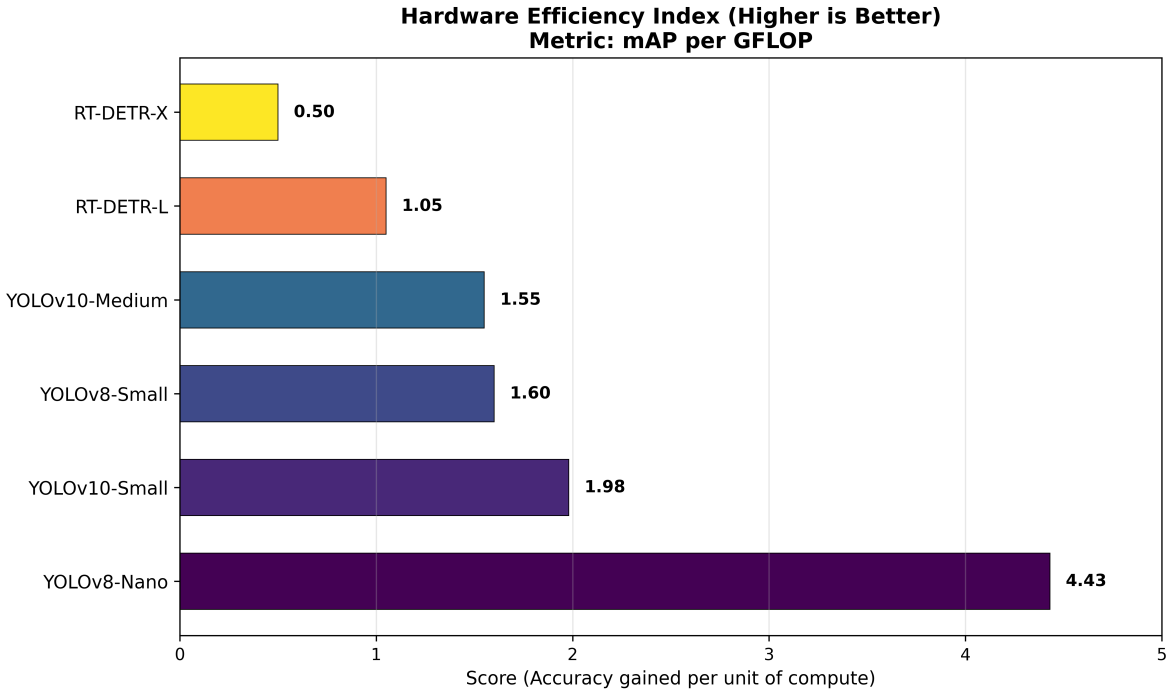

Fig. 4 Hardware efficiency index representing the accuracy gain (mAP) per unit of theoretical computational cost (GFLOPs). A higher score indicates superior architectural optimization for the underlying hardware.

Table 2 Per-class Average Precision (AP@50) comparison on six safety-critical object categories across all 11 evaluated models based on COCO val2017 benchmarks. The data demonstrates a significant performance gap on small targets: while the lightweight YOLOv8-Nano achieves acceptable accuracy for large objects (e.g., 79.0% for Person), it suffers severe degradation on small indicators like Traffic Lights (36.3%), whereas RT-DETR-L maintains robust detection capabilities (58.6%) critical for safe navigation.

Model	Person	Car	Traffic Light	Bus	Truck	Stop Sign	mAP@0.5 (Composite)
RT-DETR-X	86.4	77.7	61.1	81.0	69.5	69.9	73.8
RT-DETR-L	86.5	77.0	58.6	75.7	63.7	78.0	72.7
YOLOv10-Medium	84.6	75.1	51.8	79.8	56.0	71.7	70.5
YOLOv11-Medium	85.7	77.1	53.3	83.0	57.9	68.6	70.2
YOLOv8-Medium	85.3	75.8	52.2	76.4	59.3	58.6	68.4
YOLOv10-Small	81.1	70.4	47.0	74.9	50.4	64.8	65.2
YOLOv11-Small	83.5	69.5	48.9	73.2	47.2	54.6	65.2
YOLOv8-Small	82.7	70.0	46.4	75.0	48.9	62.4	63.1
YOLOv11-Nano	79.0	63.6	37.9	63.5	36.7	54.2	56.9
YOLOv10-Nano	75.9	59.2	35.7	68.9	40.9	55.0	55.8
YOLOv8-Nano	79.0	60.0	36.3	65.4	37.1	57.5	54.6

tion and long-term thermal saturation.

The entire experiment was conducted in PyTorch (Python), but in C++ (CoreML), the results would likely reduce interpreter overhead.

Real-world navigation is unpredictable, so we leaned on the diverse mix of images in COCO 2017 rather than trying to filter for perfect lighting or weather. However, running a software benchmark isn't the same as guaranteeing user safety. Before this system is ready for the real world, we need to verify how the actual hardware holds up when the lights go down or conditions get tough.

Conclusion

This research compared architectural approaches, including NMS-based YOLO and End-to-End RT-DETR, on the Ap-

ple MacBook M2. In the results, YOLO (v8/v10) showed the highest latency (jitter) dispersion in complex scenes due to its NMS, which induces computational bottlenecks. However, RT-DETR demonstrated superior temporal stability.

For navigation tasks for Visually Impaired People (VIP), RT-DETR is the preferred option. Although its peak FPS may be lower, its predictability (low jitter) minimizes the risk, which is critical for safety. Latency stability is prioritized over raw throughput.

In addition, we expect the following: Transition from Python/PyTorch to native CoreML (C++) and application of quantization (Int8), in order to decrease memory consumption; validation of the results on other ARM-chips (Qualcomm Snapdragon, NVIDIA Jetson), in order to check the hypothesis about "Software-Hardware Gap"; Conducting longitudinal stress tests (1-2hours) to quantify the impact of thermal throttling on the stability, addressing the limitations of short-term benchmarks.

References

1 Bourne, R.R.A. (2017). Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. The Lancet Global Health, Vol. 5, No. 9, pp. 888–897.

2 Kotthapalli, M., Ravipati, D., & Bhatia, R. (2025). YOLOv1 to YOLOv11: A Comprehensive Survey of Real-Time Object Detection Innovations and Challenges. arXiv preprint arXiv:2508.02067. https://arxiv.org/abs/2508.02067.

3 Bibi, S., Asghar, M.N., Chaudhry, M.U., Hussan, N.U., & Yasir, M. (2020). A review of ARM processor architecture history, progress and applications. Journal of Applied and Emerging Sciences, Vol. 10, No. 2, pp. 171–179.

4 Wulf, W.A., & McKee, S.A. (1995). Hitting the memory wall: Implications of the obvious. ACM SIGARCH Computer Architecture News, Vol. 23, No. 1, pp. 20–24.

5 Ignatov, A. (2019). AI Benchmark: All About Deep Learning on Smartphones in 2019. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, pp. 3617–3635. https://doi.org/10.1109/ICCVW.2019.00449.

6 Terven, J., Cordova-Esparza, D., & Romero-Gonzalez, J.A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Intelligence Research, Vol. 20, No. 6, pp. 711–745.

7 Sapkota, R. (2025). YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series. Artificial Intelligence Review, Vol. 58, No. 9, Art. no. 274.

8 Vaswani, A. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (NIPS), Vol. 30, Long Beach, CA, USA, pp. 5998–6008.

9 Lv, W. (2024). DETRs Beat YOLOs on Real-time Object Detection. In Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Seattle, WA, USA, pp. 16256–16267.

10 Ma, N., Zhang, X., Zheng, H., & Sun, J. (2018). ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In Proc. Eur. Conf. Comput. Vis. (ECCV), Munich, Germany, pp. 116–131.

11 Silva, L.H.S.L. (2023). Benchmarking Edge Computing Devices for Deep Learning Inference. IEEE Internet of Things Journal, Vol. 10, No. 3, pp. 2456–2474.

12 Velasco-Montero, B.C., Fernandez-Berni, J., Carmona-Galan, R., & Rodriguez-Vazquez, A. (2021). Impact of Thermal Throttling on Long-Term Visual Inference in a CPU-Based Edge Device. IEEE Access, Vol. 9, pp. 167663–167676. https://doi.org/10.1109/ACCESS.2021.3136586.

13 Abadi, M. (2023). Performance Analysis of Deep Learning Inference on Apple Silicon M1/M2 Chips. IEEE Access, Vol. 11, pp. 12890–12905.

14 Sereda, T., John, T., Bartan, B., Serrino, N., Katti, S., & Asgar, Z. (2025). KForge: Program Synthesis for Diverse AI Hardware Accelerators. arXiv preprint arXiv:2511.13274. https://arxiv.org/abs/2511.13274.

15 Feng, D., Xu, Z., Wang, R., & Lin, F.X. (2025). Profiling Apple Silicon Performance for ML Training. arXiv preprint arXiv:2501.14925. https://arxiv.org/abs/2501.14925.

16 Contributors, PyTorch. (2025). MPS Environment Variables. PyTorch Documentation. https://docs.pytorch.org/docs/stable/mps_environment_variables.html.

17 Inc, Apple. (2023). Optimize machine learning for Metal apps. Apple Developer, WWDC23 Session 10050. https://developer.apple.com/videos/play/wwdc2023/10050/.

18 Mekonnen, F.Y. (2025). Development of a Fully Autonomous Offline Assistive System for Visually Impaired Individuals: A Privacy-First Approach. Sensors, Vol. 25, No. 19, Art. no. 6006.

19 Li, K., Amiri, A., & Raja, H. (2025). EcoEdgeInfer: Toward Efficient and Sustainable On-device Inference for Vision Transformers on Edge Platforms. arXiv preprint arXiv:2506.00363. https://arxiv.org/abs/2506.00363.

20 Abdulkadhim, M., & Repas, S.R. (2025). SHEAB: Supporting Heterogeneous Edge AI Benchmarking. Technologies, Vol. 13, No. 11, Art. no. 515.

21 Li, Z., Paolieri, M., & Golubchik, L. (2024). A Benchmark for ML Inference Latency on Mobile Devices. In Proc. 7th Int. Workshop on Edge Systems, Analytics and Networking (EdgeSys '24), Athens, Greece, pp. 19–24.