International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 6, Issue 2, 229-233, 2025
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V6I2P126

Original Article

Benchmarking Hybrid On-Device and Cloud AI Inference for Android and Embedded Devices: Latency, Memory, Energy, Privacy, and Accuracy Trade-offs

Srikanth Puram
Independent Researcher, Mobile and Embedded Software, Architecture Novi, Michigan, USA.

Received On: 24/04/2025 ***Revised On: 04/05/2025*** ***Accepted On: 19/05/2025*** ***Published on: 10/06/2025***

Abstract - *Android phones, automotive displays, wearables, home devices, retail terminals, industrial gateways, and other embedded gadgets increasingly run artificial intelligence workloads that were previously hosted almost exclusively in the cloud. On-device inference improves privacy, offline behavior, and round-trip latency, but it is constrained by model size, memory, battery, thermal limits, accelerator availability, and fragmented hardware. Cloud inference supports larger models and centralized updates, but it introduces network latency, availability risk, data-transfer cost, privacy exposure, and service dependency. This paper proposes a benchmarking and decision framework for hybrid on-device and cloud AI inference on Android and embedded devices. The framework evaluates five deployment modes: fully on-device inference, cloud-only inference, local prefiltering with cloud escalation, confidence-based hybrid routing, and cloud fallback after local failure. It defines workload classes, device classes, network profiles, privacy levels, and metrics including p50/p95 latency, time-to-first-token, throughput, peak RAM, model size, energy per inference, quantization accuracy loss, network round-trip delay, and privacy exposure. The paper also presents a routing algorithm that selects the inference location based on latency budget, privacy class, model memory footprint, battery state, network quality, and confidence threshold. The proposed framework is aligned with the technology landscape available by April 2025, including LiteRT/TensorFlow Lite foundations, Android on-device AI APIs, Gemini Nano experimental access, Gemma 3 class open models, ONNX Runtime Mobile, and MLPerf-style inference benchmarking.*

Keywords - *Android AI, Embedded AI, On-Device Inference, Cloud Inference, Hybrid AI, LiteRT, TensorFlow Lite, Gemini Nano, Gemma, ONNX Runtime Mobile, Latency Benchmarking, Edge Intelligence.*

1. Introduction

AI workloads are moving closer to users and devices. Mobile and embedded platforms now perform image classification, anomaly detection, keyword spotting, object detection, text classification, summarization, translation, personalization, and context-aware decisioning. This shift is driven by improved mobile processors, NPUs, GPUs, optimized runtimes, quantized models, and growing user expectations for fast and private AI features.

Despite this progress, the correct placement of AI inference remains non-trivial. Running inference on-device can reduce network dependency and protect sensitive data, but large models may exceed memory budgets or drain battery. Running inference in the cloud can provide stronger reasoning, larger context windows, and faster model iteration, but cloud inference depends on network quality and may not satisfy strict privacy or offline requirements. Hybrid inference seeks a balance by combining local pre-processing, on-device inference, confidence-based routing, and cloud fallback.

This paper proposes a benchmark-driven decision framework for Android and embedded devices. The objective is not to claim that one deployment mode is universally superior. Instead, the framework helps engineers select the appropriate inference placement for each workload, device class, privacy requirement, and service-level objective.

2. Technology Evolution up to April 2025

The evolution of on-device AI can be viewed in three waves. The first wave focused on compact neural networks for vision, speech, and sensor workloads using frameworks such as TensorFlow Lite and TensorFlow Lite Micro. The second wave added heterogeneous acceleration through GPUs, DSPs, NPUs, and runtime-specific delegates. The third wave, visible by 2024 and early 2025, introduced mobile and edge access to generative AI capabilities through optimized runtimes, on-device foundation models, and hybrid cloud-device APIs.

By April 2025, Android and embedded AI developers could reasonably consider LiteRT/TensorFlow Lite-style deployment, ONNX Runtime Mobile, hardware acceleration where available, Gemini Nano experimental access on supported Android environments, and Gemma 3 class open models for mobile and edge experimentation. The practical challenge is not merely whether a model can run, but whether it can run within the latency, memory, energy, privacy, and reliability envelope of a product.

3. Problem Statement

Engineering teams need a repeatable way to decide whether an AI workload should run locally, in the cloud, or through a hybrid path. The decision depends on variables that change at runtime: network latency, device memory, battery state, thermal throttling, model confidence, input sensitivity, and user-perceived latency budget. Static architecture choices can become inefficient or unsafe as device conditions change.

For example, an on-device image classifier may run with acceptable latency on a high-end phone but exceed memory limits on a low-cost embedded display. A cloud summarization model may produce higher-quality output but fail during poor connectivity. A privacy-sensitive text input may require local processing even if the local model is smaller. Therefore, inference placement should be treated as a policy and benchmarking problem rather than a one-time implementation decision.

4. Workload Taxonomy

The framework groups AI workloads into five classes: sensor and anomaly workloads, vision workloads, speech workloads, language workloads, and context-policy workloads. Each class has different performance and privacy characteristics. Small classification tasks often fit on-device. Large reasoning and long-context generation tasks often favor the cloud. Intermediate tasks may use hybrid routing based on confidence and network conditions.

Table 1: AI Workload Taxonomy for Android and Embedded Devices

Workload class	Examples	Typical local suitability	Common cloud reason
Sensor/anomaly	vibration anomaly, battery anomaly, update-risk classification	High; compact models and small inputs	Centralized fleet analytics or model retraining.
Vision	image classification, object detection, document capture quality	Medium to high depending on model size and accelerator	High-resolution analysis or large multimodal model.
Speech	keyword spotting, short command recognition	High for wake word and command detection	Large-vocabulary ASR or dialogue reasoning.
Language	classification, summarization, rewrite, small assistant actions	Medium for short text and compact models	Long context, stronger reasoning, larger model.
Context-policy	route selection, risk scoring, personalization, UI prediction	High when features are local and privacy-sensitive	Cross-device learning or global policy optimization.

5. Deployment Modes

The framework evaluates five deployment modes. Fully on-device inference keeps the complete model and execution local. Cloud-only inference sends the request to a remote service. Local prefiltering performs a lightweight on-device check before cloud escalation. Confidence-based hybrid routing executes locally first and escalates only if confidence is low or the task requires a larger model. Cloud fallback uses local inference as the default but switches to cloud only when local execution fails, times out, or lacks required capability.

Table 2: Inference Deployment Modes

Mode	Description	Strength	Weakness
On-device	Input, model, and inference remain on the device.	Privacy, offline use, low network dependency.	Memory, energy, thermal, smaller model capacity.
Cloud-only	Input is transmitted to cloud model service.	Larger models, centralized updates, high capacity.	Network latency, privacy exposure, service dependency.
Local prefilter + cloud	Small local model filters or compresses request before cloud.	Reduces bandwidth and cloud calls.	Requires careful threshold tuning.
Confidence hybrid	Local model runs first; cloud used when confidence is low.	Balances latency and quality.	Needs calibrated confidence and policy.
Cloud fallback	Device is default; cloud handles unsupported or failed cases.	Reliable user experience under variable capability.	Fallback may add tail latency.

6. Benchmark Metrics

The benchmark uses metrics that reflect user experience, device constraints, and privacy requirements. Mean latency alone is insufficient because users experience tail latency. p95 latency, time-to-first-token, throughput, peak resident memory, energy per inference, and thermal stability are more useful for production decisions. For generative AI tasks, time-to-first-token and tokens per second should be reported separately. For classification tasks, top-1 accuracy, F1 score, and quantization accuracy loss are appropriate. For hybrid systems, cloud escalation rate and fallback success rate are critical.

Table 3: Metrics for AI Inference Placement Benchmarking

Metric	Definition	Applies to
p50 latency	Median end-to-end inference time.	All workloads.
p95 latency	95th percentile end-to-end inference time.	User-facing and safety-sensitive workloads.
TTFT	Time from request start to first generated token.	Generative language workloads.
Throughput	Images/sec, samples/sec, or tokens/sec.	Batch or streaming workloads.
Peak RAM	Maximum resident memory above baseline.	All on-device workloads.
Energy/inference	Estimated joules or battery percentage per request.	Mobile and embedded devices.
Accuracy loss	Full-precision accuracy minus quantized accuracy.	Compressed local models.
Escalation rate	Requests routed from device to cloud / total requests.	Hybrid routing.
Privacy exposure class	Sensitivity of data leaving device.	Cloud and hybrid modes.
Offline completion rate	Requests completed without network / offline requests.	On-device and fallback modes.

7. Resource and Decision Model

The placement decision is modeled as constrained optimization. A workload W has requirements for latency budget L, privacy class P, accuracy floor A, and maximum resource cost C. A device D has available memory, battery, accelerator capability, and thermal state. A network N has round-trip time, packet loss, bandwidth, and trust class. The router selects the mode that satisfies hard constraints and minimizes weighted cost.

The following cost model can be used for decisioning: Cost(m) = alpha * Latency(m) + beta * Memory(m) + gamma * Energy(m) + delta * PrivacyRisk(m) + epsilon * FailureRisk(m) - zeta * Accuracy(m). Hard constraints such as privacy class and memory availability are applied before cost minimization.

Algorithm 1. Hybrid inference placement decision

```
Input: workload W, input x, device state D, network state N, privacy class P, model catalog C
Output: inference mode m and model k
1: candidates <- getCompatibleModels(W, C)
2: remove candidates where model_size > available_memory(D) - safety_margin
3: remove cloud modes if P prohibits off-device transfer
4: remove modes whose expected_latency exceeds W.max_latency unless no alternative exists
5: if network_unavailable(N) then choose best on-device candidate or return DEFER
6: for each remaining candidate c:
7:     estimate latency, energy, RAM, accuracy, privacy risk, and failure risk
8:     score[c] <- weightedCost(c)
9: choose c_min with minimum score
10: if c_min is local and confidence(x) < tau_conf then route to cloud if allowed
11: return selected mode and model
```

8. Benchmark Design

A credible benchmark should combine device classes, workload classes, model families, and network profiles. It should report raw measurements rather than only aggregate scores. The same workload should be measured across on-device, cloud, and hybrid modes so that the trade-off is visible. Device classes should include low-memory Android gadgets, mid-range mobile devices, high-end mobile devices, and Android-based embedded or automotive-style systems. Network profiles should include offline, poor cellular, average cellular, Wi-Fi, and managed low-latency network conditions.

Table 4: Reproducible Benchmark Matrix

Dimension	Benchmark levels
Device class	Low-memory Android device; mid-tier phone; high-end phone; Android embedded/automotive device.
Network profile	Offline; 150-300 ms RTT cellular; 50-100 ms RTT cellular; Wi-Fi; managed edge network.
Model family	Tiny classifier; MobileNet/EfficientNet class vision; keyword spotting; anomaly model; compact language model.
Runtime path	CPU; GPU delegate; NPU/accelerator where available; cloud endpoint.
Precision	FP32 baseline; FP16; INT8; optional lower-bit quantized model if supported.
Measurement	30 warm runs; 100 measured runs; cold-start and warm-start separated.

Table 5: Workload-Specific Reporting Plan

Workload	Input	Local metrics	Cloud metrics	Hybrid metrics
Image classification	224x224 image	latency, RAM, energy, top-1 accuracy	RTT, total latency, accuracy	local confidence, escalation rate
Object detection	camera frame	p95 latency, thermal behavior, FPS	network upload time, latency	local detection + cloud verification rate
Keyword spotting	audio window	latency, energy, false wake rate	not preferred for privacy/latency	local wake + cloud dialogue
Anomaly detection	sensor vector	latency, RAM, F1, offline rate	fleet analysis latency	local triage + cloud retraining
Text summarization	short text	TTFT, tokens/sec, RAM, quality score	RTT, TTFT, quality score	local short summary + cloud fallback

9. Quantization, Memory, and Thermal Constraints

On-device AI often depends on compression and quantization. INT8 quantization can reduce model size and memory bandwidth, but it may reduce accuracy. FP16 can reduce memory compared with FP32 while preserving more numeric fidelity, especially on supported accelerators. For language models, memory pressure includes weights, activation buffers, tokenizer state, key-value cache, and runtime overhead. Therefore, model file size alone is not a sufficient measure of feasibility. Thermal behavior should also be reported because repeated inference may throttle CPU, GPU, or NPU performance. A model that meets latency for a single request may fail p95 latency after sustained use. The benchmark should therefore include cold-start, warm-start, and sustained-run measurements.

10. Privacy and Security Considerations

Privacy classification should be a first-class input to routing. Inputs containing payment details, health information, authentication data, private messages, or precise location should default to on-device processing or require explicit policy approval before cloud transfer. Cloud processing may still be appropriate for non-sensitive or consented data, but the routing system should not treat all inputs equally. Security controls include model integrity validation, secure model storage, authenticated cloud endpoints, transport encryption, token protection, prompt and input minimization, and local redaction before cloud escalation. A hybrid architecture should log routing decisions without storing sensitive raw inputs.

11. Benchmark Reporting Template

To support reproducibility, benchmark results should be reported in a structured format. At minimum, every run should include device model, OS version, runtime, accelerator path, model identifier, precision, input size, network profile, p50 latency, p95 latency, peak RAM, energy estimate, accuracy, fallback reason, and failure class. The benchmark should distinguish measured results from policy thresholds. For example, a product may set a p95 latency budget of 500 ms for interactive classification or 2 s for a short summarization task, but those values are service-level requirements rather than measured results.

Table 6: Benchmark Result Schema

Field	Example value format	Notes
device_class	mid_tier_android / embedded_android	Avoid relying only on brand name.
runtime_path	CPU / GPU / NPU / cloud	Report delegate or accelerator.
model_precision	FP32 / FP16 / INT8	Include quantization method.
network_profile	offline / cellular_200ms / wifi_40ms	Needed for cloud and hybrid modes.
p50_ms and p95_ms	numeric milliseconds	Report end-to-end and model-only if possible.
peak_ram_mb	numeric MB above baseline	Use consistent measurement method.
energy_j	joules or estimated battery delta	State measurement device or approximation.
privacy_class	public / internal / sensitive / restricted	Used by routing policy.
decision_mode	local / cloud / hybrid / fallback	Final selected execution path.

12. Discussion

The strongest practical result of a hybrid framework is not always the lowest average latency. It is the ability to meet privacy, latency, reliability, and energy constraints under variable conditions. In many mobile and embedded products, the correct design is a cascade: local model for fast and private first-pass inference, cloud escalation for low-confidence or high-complexity cases, and offline fallback when connectivity is unavailable.

The decision framework also helps avoid overloading devices with models that are impressive but unsuitable for sustained use. A compact local model that completes 95 percent of requests under the latency budget may be better than a larger model that only works on high-end devices. Conversely, cloud inference may be justified when accuracy or reasoning quality is materially higher and privacy policy allows transfer. By separating metrics, constraints, and routing policy, the framework allows product teams to evolve models without rewriting application architecture. New model families, accelerators, or cloud services can be added

to the model catalog and compared using the same benchmark schema.

13. Limitations and Future Work

This paper defines a benchmark and decision framework rather than publishing device-specific measurement results. A complete empirical study should run the proposed protocol across real devices and release raw results, model artifacts, scripts, and network profiles. Future work may include thermal-aware routing, adaptive confidence thresholds, local retrieval-augmented generation, federated personalization, and safety evaluation for on-device generative models.

14. Conclusion

This paper proposed a benchmarking and decision framework for hybrid on-device and cloud AI inference in Android and embedded devices. The framework compares on-device, cloud, and hybrid modes using latency, memory, energy, accuracy, privacy, network dependency, and reliability metrics. It also provides a routing algorithm that uses hard privacy and memory constraints before optimizing weighted cost. The approach is designed for the April 2025 technology landscape and provides a practical method for engineering teams to select inference placement without assuming that either local or cloud AI is universally superior.

References

[1] Google, "TensorFlow Lite is now LiteRT," Google Developers Blog, Sep. 2024.
[2] Google, "LiteRT: High-Performance On-Device Machine Learning," Google AI Edge Documentation, 2024.
[3] Android Developers Blog, "Gemini Nano experimental access available on Android," Oct. 2024.
[4] Google AI for Developers, "Gemma releases: Gemma 3 model card and release notes," Mar. 2025.
[5] ONNX Runtime, "ONNX Runtime Mobile Documentation," Microsoft, 2024.
[6] Android Developers, "Neural Networks API," Android NDK Documentation, 2024.
[7] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," arXiv:2010.08678, 2020.
[8] C. Banbury et al., "MLPerf Tiny Benchmark," arXiv:2106.07597, 2021.
[9] V. J. Reddi et al., "MLPerf Inference Benchmark," arXiv:1911.02549, 2019.
[10] J. Lee et al., "On-Device Neural Net Inference with Mobile GPUs," arXiv:1907.01989, 2019.
[11] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML, 2019.
[12] A. Howard et al., "Searching for MobileNetV3," ICCV, 2019.
[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT," NeurIPS Workshop, 2019.
[14] Z. Sun et al., "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices," ACL, 2020.
[15] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023.
[16] OWASP Foundation, "Mobile Application Security Verification Standard," 2024.
[17] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018.
[18] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," OSDI, 2016.
[19] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," NeurIPS, 2019.
[20] National Institute of Standards and Technology, "Secure Software Development Framework (SSDF) Version 1.1," NIST SP 800-218, Feb. 2022.