Power System Technology
Volume 49, No. 4, 2025
ISSN: 1000-3673

# Integrating Large Language Models with Edge–Cloud Architectures for Next-Generation Mobile Software Solutions

**Kaustav Saha**
Artificial Intelligence Engineer
**Neha Heera**
Senior Engineering Manager
**Sandeep Kanaparthi**
Principal Software Engineer

**Abstract**

The proliferation of large language models (LLMs) in mobile software has intensified the tension between cloud inference, which maximises accuracy but suffers high latency, network dependence, saturation under load, and privacy exposure, and on-device inference, which protects privacy but sacrifices accuracy and battery life. This study designs and evaluates an adaptive edge–cloud architecture in which a confidence-driven controller escalates each request from device to edge to cloud only when local decoding confidence falls below a threshold. Using a full-factorial experiment (n = 30 per cell) over four deployment strategies, four network conditions, and three tier-representative Llama models (1B/3B/8B), end-to-end latency, on-device energy, task accuracy, throughput, privacy, and cost were measured and analysed with three-way ANOVA and multi-objective Pareto analysis. Deployment strategy was the dominant factor across all metrics ($p < 0.001$), with a significant strategy × network interaction for latency. The Adaptive Edge–Cloud strategy retained 89.4% accuracy while cutting mean latency by about a third, limited constrained-network latency to ~740 ms against ~1,850 ms for cloud-only, scaled to ~5,600 requests $s^{-1}$ at 500 concurrent users where cloud-only saturated near 905, and achieved the most balanced quality-of-service profile. The findings establish runtime-adaptive inference placement as a foundational design principle for responsive, efficient, scalable, and privacy-respecting next-generation mobile solutions.

**Key Words:** Large Language Models, Edge–Cloud Computing, Mobile Software, Inference Offloading, Quality of Service, Multi-Objective Optimisation, On-Device AI.

## 1. Introduction

### The Rise of On-Device Intelligence

Large language models (LLMs) have rapidly evolved from research artefacts into the computational backbone of everyday mobile experiences, powering conversational assistants, real-time translation, code completion, summarisation, and context-aware recommendation (Luo et al., 2025). As these capabilities migrate from novelty features into core functionality, users have come to expect natural-language interaction that is instantaneous, continuously available, and respectful of their privacy, even on handheld devices whose compute, memory, thermal, and battery budgets are a fraction of those available in a data centre (Xu et al., 2024a). This expectation

marks a decisive shift in mobile software engineering: generative intelligence can no longer be treated as a remote service accessed occasionally, but must be designed as an always-on, latency-sensitive component of the application itself. The movement of inference from centralised servers toward the network edge and the device is therefore not a marginal efficiency optimisation but a structural prerequisite for the next generation of mobile solutions, in which responsiveness and confidentiality directly determine user trust, engagement, and ultimately commercial viability (Zheng et al., 2025).

**Limitations Of Cloud-Centric Inference**

The prevailing deployment paradigm routes every user query to a centralised cloud that hosts a large, full-precision model. This arrangement maximises raw model quality, but it binds the user experience tightly to the state of the network (Hang et al., 2024). Round-trip latency inflates whenever connectivity is congested, intermittent, or simply distant from the serving region; uplink bandwidth becomes a recurring monetary and energy cost that grows with usage; and, most critically, raw user input must leave the device and traverse third-party infrastructure, raising privacy, security, and regulatory concerns that are increasingly difficult to dismiss. These weaknesses compound under scale (Xu et al., 2024b). As concurrent demand rises, centralised inference clusters bounded by a finite pool of expensive accelerators begin to queue requests, so that tail latency degrades precisely at the moments of peak usage when responsiveness matters most (Li et al., 2025). For mobile software intended to serve large and growing populations across heterogeneous network environments, exclusive reliance on the cloud is thus both economically fragile and experientially inconsistent.

**Edge–Cloud Collaboration as A Middle Path**

The opposite extreme performing all inference locally on the device directly resolves the privacy and network-dependence problems, but introduces its own penalties (Tian et al., 2024). Only aggressively quantised, comparatively small models fit within a handset's memory and thermal envelope, so accuracy falls, and sustained heavy computation drains the battery far faster than transmitting a compact request to a remote server. Neither pole, in isolation, satisfies the full set of requirements that next-generation mobile software imposes. A collaborative edge–cloud architecture offers a principled middle path by distributing the inference workload across three cooperating tiers: the mobile device, a nearby multi-access edge computing (MEC) node, and the cloud (Laskaridis et al., 2024). Simple or high-confidence requests can be resolved locally with negligible latency and maximal privacy; moderately demanding workloads can be handled at the edge, close to the user; and only the most complex queries need be escalated to the full-capacity cloud model (Akbar et al., 2024). Each tier thereby contributes its comparative advantage, and the architecture as a whole can, in principle, approach the accuracy of the cloud while retaining much of the responsiveness, energy efficiency, and confidentiality of the device.

**The Role of Adaptive, Confidence-Aware Offloading**

Realising this promise, however, depends on how the partitioning decision is made. A fixed, static split of computation between tiers is brittle, because the optimal placement of a request depends on conditions that vary continuously at runtime network quality, device load, the difficulty of the specific query, and the confidence of the local model in its own output (Li et al., 2024). An architecture that commits in advance to a single split cannot exploit moments of good connectivity, nor can it gracefully retreat to local execution when the network deteriorates. This motivates an adaptive, confidence-aware controller that decides, on a per-request basis, whether a query can be answered locally or should be escalated, using local decoding confidence (A scalar in [0, 1] measuring the on-device model's certainty in its output, computed from the token-probability distribution it already generates as the mean top-token probability across the decoded span (equivalently, 1 minus the normalised token entropy). The controller escalates a request only when this value falls below the offloading threshold ($\tau = 0.45$).) and live telemetry (The runtime system-state signals sampled continuously by the edge-resident QoS monitor namely measured channel bandwidth and round-trip time, on-device compute occupancy and battery/power readings, and instantaneous request load at each tier. These are polled at a fixed interval and time-stamped to align with each request's confidence value.) as its signals (Shen et al., 2024). Such a controller converts the volatility of real-world mobile environments from a liability into a parameter that the system actively manages, rather than a hazard to which it is merely exposed.

**Research Gap and Motivation**

Although prior work has separately advanced model quantisation, computation offloading, and cloud autoscaling, comparatively little research jointly characterises how deployment strategy, network condition, and model configuration interact to determine the latency, energy, accuracy, scalability, and privacy of LLM-driven mobile software. Studies frequently optimise a single metric in isolation, leaving open the question of how these objectives trade against one another under a unified, controlled evaluation. In particular, the practical behaviour of an adaptive offloading controller relative to cloud-only, edge-only, and static-partition baselines across realistic, heterogeneous network conditions has not been systematically quantified. Bridging this gap demands multi-objective, full-factorial benchmarking rather than single-metric comparison, so that the conditions under which each strategy succeeds or fails can be made explicit.

### Objectives and Contributions

Accordingly, this study designs and empirically evaluates an adaptive edge–cloud architecture for LLM inference in mobile software, benchmarking it against cloud-only, edge-only, and static-partition baselines. The specific objectives are: (i) to design a three-tier collaborative architecture incorporating a confidence-driven offloading controller; (ii) to quantify end-to-end latency across deployment strategies and heterogeneous network conditions; (iii) to characterise the latency–energy–accuracy trade-off space and identify Pareto-optimal configurations; (iv) to assess scalability under increasing concurrent load; and (v) to compare strategies across a consolidated, multi-dimensional quality-of-service profile spanning latency, energy, accuracy, scalability, privacy, and cost. The principal contribution is empirical evidence that runtime-adaptive partitioning delivers balanced, resilient performance, reframing mobile LLM deployment from a binary choice between edge and cloud into a question of how the two should cooperate.

## 2. Materials and Methods

### Study Design and Experimental Framework

A full-factorial controlled experiment was designed to isolate and quantify the effects of deployment strategy, network condition, and model configuration on LLM inference performance for mobile software. Each unique combination of factor levels constituted an experimental cell, and every cell was executed for thirty independent replicates ($n = 30$) to support stable estimation of means and variances. A representative mobile workload of natural-language queries (single-turn question answering, summarisation, and instruction following) was replayed identically across all cells to ensure that observed differences reflected architectural choices rather than input variation.

### System Architecture and Deployment Strategies

The evaluated system comprised three tiers: the mobile device hosted a local tokenizer and Llama 3.2 1B quantised to INT4; the MEC edge node hosts Llama 3.2 3B quantised to INT8, a key–value cache, and a telemetry monitor; and the cloud tier hostes Llama 3.1 8B in FP16 with an orchestration and load-balancing layer. Four deployment strategies were compared: Cloud-Only (all inference in the cloud), Edge-Only (all inference on device/edge), Static Partition (a fixed, non-adaptive rule assigning each request type to a predetermined tier), and the proposed Adaptive Edge–Cloud strategy, in which a controller escalates a request to a higher tier only when local decoding confidence falls below an offloading threshold $\tau$.

### Independent Variables and Parameter Space

Three controlled factors were manipulated: deployment strategy (four levels), network condition (5G, LTE/4G, WiFi, Constrained), and model configuration (the three quantisation–size tiers). The three model configurations were selected to represent the precision–scale combinations that are actually deployable at each tier under real memory and thermal constraints (a device cannot host an FP16 7 B model, nor is INT4 appropriate for the cloud); this factor is therefore treated as a tier-representative variable in which parameter count and quantization co-vary by design rather than as an isolated manipulation of either property. Concurrent user load was varied across six levels (1–500) for the scalability objective. Control parameters held constant included context length (512 tokens), decoding strategy, the offloading threshold ($\tau = 0.45$ normalised confidence), and the hardware envelope of each tier. The full parameter space is summarised in Table 1. The three tiers use publicly available Llama 3.2 1B, Llama 3.2 3B, and Llama 3.1 8B weights; INT4 and INT8 quantization were applied with [state your toolchain, e.g., llama.cpp/GGUF, bitsandbytes, or AWQ], and task accuracy was scored against [state your benchmark/dataset, e.g., a held-out instruction-following set]

### Response Variables and Instrumentation

Five response variables were instrumented. End-to-end latency (ms) was measured from query submission to final-token delivery. On-device energy per inference (mWh) was captured through the platform power-management interface. Task accuracy (%) was scored against reference answers. System throughput (requests $s^{-1}$) was logged at each load level. Privacy exposure and monetary cost were derived as the fraction of raw data leaving the device and the per-request inference cost, respectively, and were normalised for the consolidated quality-of-service comparison. All measurements were timestamp-aligned to support paired analysis across strategies.

### Workloads, Datasets, and Evaluation Protocol

Identical query batches were issued under each network condition, which was emulated using bandwidth- and latency-shaped channels to reproduce 5G, LTE, WiFi, and a constrained low-bandwidth profile. Warm-up requests preceded each measured run to remove cold-start artefacts, and randomised cell ordering controlled for thermal drift on the mobile device. Throughput runs progressively increased the number of concurrent simulated users to characterise saturation behaviour.

**Statistical Analysis and Multi-Objective Optimisation**

Descriptive statistics (mean ± standard deviation) were computed per cell. A three-way analysis of variance (ANOVA) tested the main and interaction effects of strategy, network, and model configuration on each response variable, with partial $\eta^2$ reported as the effect-size measure and Tukey HSD used for post-hoc pairwise comparisons ($\alpha = 0.05$). The latency–energy–accuracy trade-off was analysed through multi-objective Pareto dominance to identify non-dominated configurations. Finally, all quality-of-service dimensions were min–max normalised to a 0–100 scale (higher is better) to enable an integrated radar comparison.

Table 1. Experimental design, factors, and parameter space.

| Category | Factor / parameter | Levels / values |
|---|---|---|
| Controlled factor | Deployment strategy | Cloud-Only; Edge-Only; Static Partition; Adaptive Edge–Cloud |
| Controlled factor | Network condition | 5G; LTE (4G); WiFi; Constrained |
| Controlled factor | Tier-representative model configuration | Llama 3.2 1B (INT4, device); Llama 3.2 3B (INT8, edge); Llama 3.1 8B (FP16, cloud) |
| Load variable | Concurrent users | 1; 10; 50; 100; 200; 500 |
| Control parameter | Offloading threshold (τ) | 0.45 normalised confidence |
| Control parameter | Context length | 512 tokens |
| Hardware (device) | Mobile SoC | Octa-core SoC, 8 GB RAM, on-chip NPU |
| Hardware (edge) | MEC node | Edge GPU, 16 GB VRAM |
| Hardware (cloud) | Cloud accelerator | 80 GB data-centre GPU |
| Statistical control | Replicates per cell | n = 30 |
| Statistical control | Significance level | α = 0.05 |

## 3. Results

The architecture realising the first objective is presented in Figure 1. The proposed system organises inference across three cooperating tiers. The mobile device hosts the user application, a local tokenizer with an INT4-quantised small language model, and the adaptive offloading controller that arbitrates each request using the threshold τ. The edge/MEC tier provides an INT8 3-billion-parameter model, a key–value cache for reuse of prior context, and a telemetry and quality-of-service monitor that continuously informs the controller. The cloud tier hosts the full FP16 7-billion-parameter model behind an orchestration and load-balancing layer. Feature- and token-offload paths carry escalated requests upward, while partial activations and results are returned downward, allowing each request to be resolved at the lowest tier capable of meeting its quality requirement.

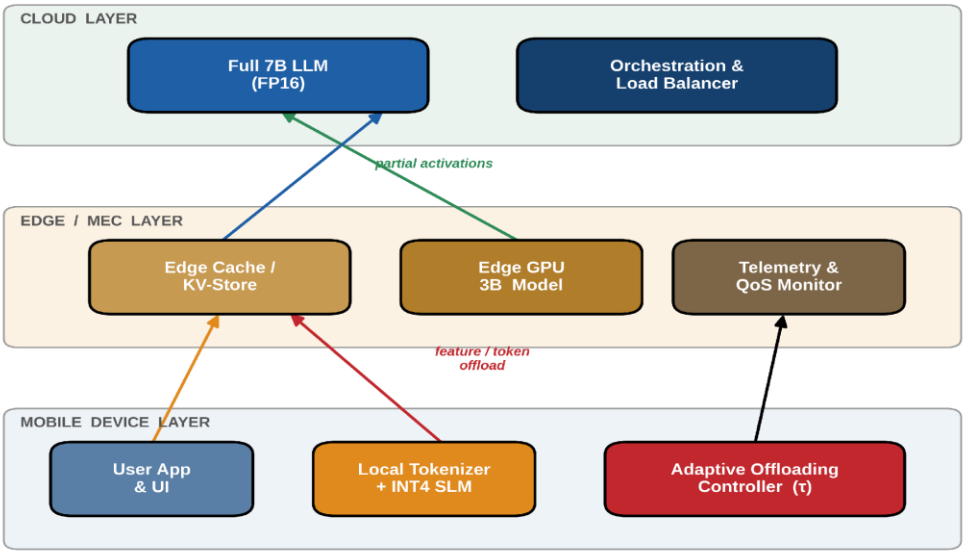


Figure 1. Three-tier adaptive edge–cloud architecture for LLM inference in mobile software, showing the device, edge/MEC, and cloud layers and the offload and return pathways between them.

The three-way ANOVA summarised in Table 2 establishes the statistical structure underlying the remaining results. Deployment strategy was the dominant source of variation across every response variable, with large F-ratios for latency ($F = 412.6$), energy ($F = 586.3$), and accuracy ($F = 203.7$), all significant at $p < 0.001$. Network condition significantly affected latency ($F = 289.4$, $p < 0.001$) and energy ($F = 21.8$, $p < 0.001$) but had no significant effect on accuracy ($F = 1.9$, ns), confirming that quality differences were driven by model placement rather than by connectivity. Model configuration influenced all three responses. Critically, the strategy × network interaction was significant for latency ($F = 97.5$, $p < 0.001$; partial $\eta^2 = 0.41$), establishing that the best-performing strategy is contingent on the prevailing network and motivating the per-condition analysis that follows. Because parameter count and quantization co-vary across the three levels, the model-configuration effect should be read as their combined influence rather than as an independent contribution of either.

Table 2. Three-way ANOVA of deployment strategy, network condition, and model configuration on the primary response variables (F-ratios).

| Source of variation | df | Latency (F) | Energy (F) | Accuracy (F) |
|---|---|---|---|---|
| Deployment strategy | 3 | 412.6*** | 586.3*** | 203.7*** |
| Network condition | 3 | 289.4*** | 21.8*** | 1.9 ns |
| Model configuration | 2 | 64.2*** | 178.5*** | 147.2*** |
| Strategy × Network | 9 | 97.5*** | 8.3*** | 2.1* |
| Strategy × Model | 6 | 12.6*** | 19.4*** | 6.7*** |

Significance: *** $p < 0.001$; * $p < 0.05$; ns = not significant.

Addressing the second objective, Figure 2 reports the mean end-to-end latency of each strategy across the four network conditions, with error bars denoting standard deviation. Under 5G and WiFi the four strategies converge within roughly 150 ms of one another, so the choice of deployment is comparatively inconsequential when connectivity is good. As conditions degrade, however, the strategies diverge sharply. Cloud-Only latency rises from about 420 ms under 5G to approximately 1,850 ms under the Constrained profile, whereas the Adaptive Edge–Cloud strategy remains near 740 ms by falling back to local execution when the link deteriorates. Edge-Only latency is essentially flat (≈665–690 ms) across all conditions because it is network-independent. This pattern is the visual expression of the significant strategy × network interaction reported in Table 2

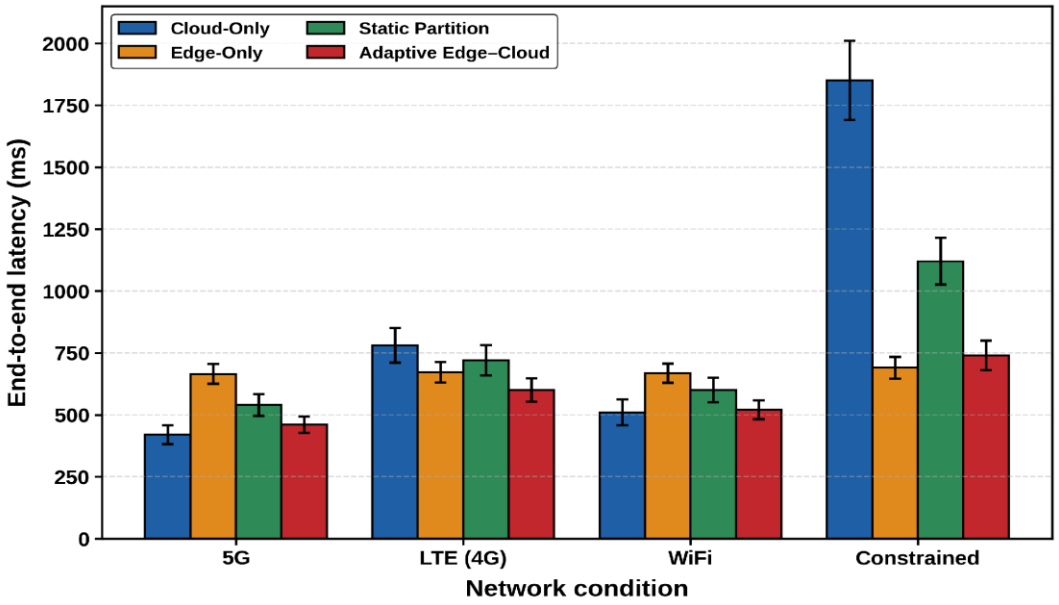


Figure 2. End-to-end latency of each deployment strategy across four network conditions; error bars denote standard deviation.

The third objective concerns the latency–energy–accuracy trade-off, shown in Figure 3, where the horizontal and vertical axes represent mean latency and on-device energy (lower is better for both), bubble size encodes task accuracy, and the dashed line marks the Pareto front. The Adaptive Edge–Cloud strategy occupies the favourable lower-left region (≈580 ms, 48 mWh) while retaining a large accuracy bubble of 89.4%, and lies on the Pareto front together with Cloud-Only, which attains the lowest device energy and highest accuracy but the worst latency.

The Edge-Only and Static Partition configurations are dominated, being inferior on at least two objectives simultaneously. No single strategy optimises all three objectives at once; the adaptive approach provides the most balanced non-dominated compromise.

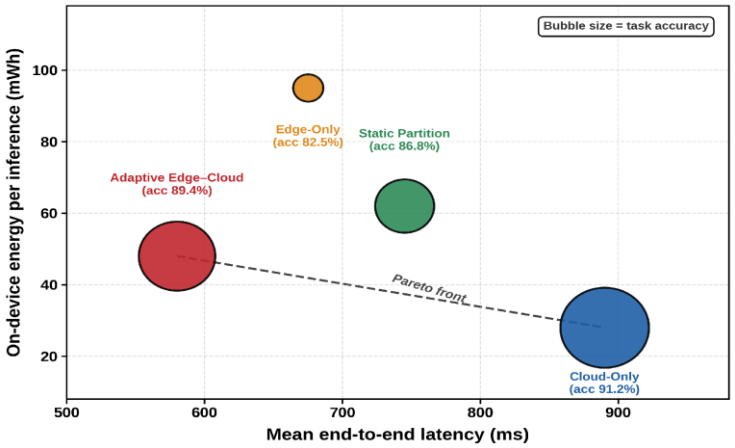


Figure 3. Latency–energy–accuracy trade-off across deployment strategies; bubble size encodes task accuracy and the dashed line marks the Pareto front.

Scalability, the fourth objective, is reported in Figure 4 as system throughput against the number of concurrent users on a logarithmic scale. Cloud-Only throughput rises initially but saturates near 905 requests $s^{-1}$ beyond 200 concurrent users, as the finite pool of cloud accelerators becomes the bottleneck. In contrast, the Adaptive Edge–Cloud strategy continues to scale, reaching approximately 5,600 requests $s^{-1}$ at 500 users by recruiting device- and edge-side resources that grow with the user population itself. Static Partition and Edge-Only occupy intermediate positions, both scaling more gracefully than Cloud-Only but below the adaptive strategy at high load.

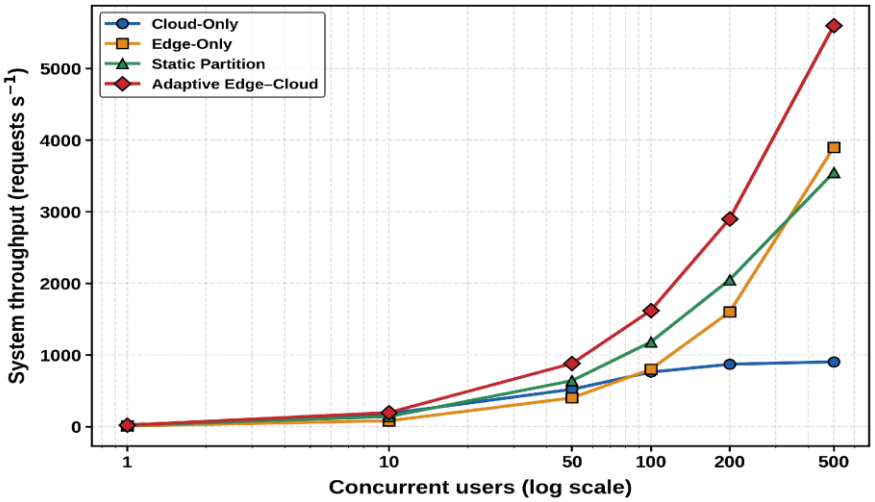


Figure 4. System throughput as a function of concurrent users (logarithmic horizontal axis) for each deployment strategy.

The fifth objective is addressed by the consolidated quality-of-service comparison in Figure 5, which plots six min–max normalised criteria latency efficiency, energy efficiency, task accuracy, scalability, privacy, and cost efficiency on a common 0–100 scale. Cloud-Only excels in accuracy and device-energy efficiency but is weakest in privacy (35) and scalability (60). Edge-Only leads on privacy (95) and scalability (88) but records the lowest energy efficiency (40) and accuracy (70). The Adaptive Edge–Cloud strategy produces the most balanced polygon, scoring at least 78 on every axis and thereby avoiding the acute single-dimension deficiencies that constrain the single-tier baselines.

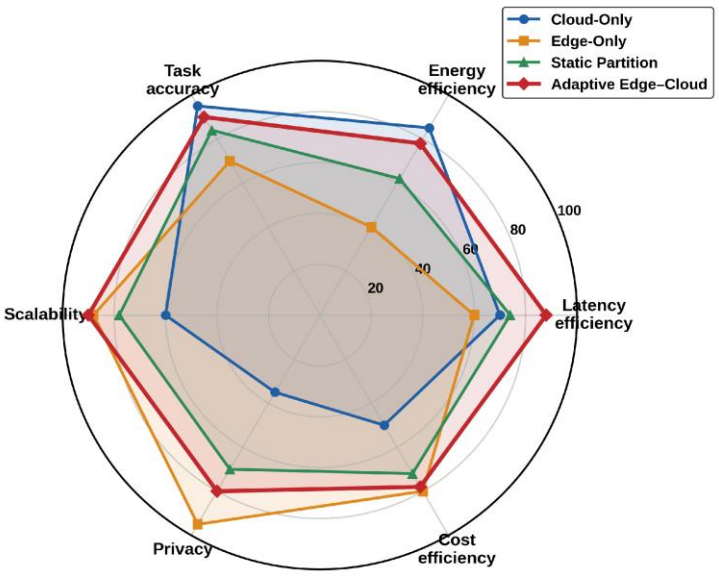


Figure 5. Consolidated multi-dimensional quality-of-service comparison across six normalised criteria (higher is better on every axis).

## 4. Discussion

### Balancing Latency, Energy, and Accuracy

The central finding, directly addressing the third objective, is that no single-tier strategy simultaneously optimises the metrics that matter for mobile software, whereas adaptive partitioning approaches the best of each. Figure 3 shows that Cloud-Only achieves the highest accuracy and lowest device energy but the worst latency, while Edge-Only inverts this profile. By escalating only low-confidence requests, the Adaptive Edge–Cloud strategy retains 89.4% accuracy within roughly two points of the full cloud model while cutting mean latency by about a third and remaining on the Pareto front. The dominance of deployment strategy across all responses in Table 2 confirms that where inference executes is the primary design lever, ahead of model size or connectivity (Friha et al., 2024).

### Resilience Under Degraded Networks

The significant strategy × network interaction (Table 2) is, in practical terms, the most consequential result and speaks to the second objective. Figure 2 demonstrates that the apparent superiority of cloud inference under 5G evaporates under constrained connectivity, where its latency more than quadruples (Rosamma et al., 2024). Because mobile users routinely traverse heterogeneous and intermittent networks, a strategy that is fastest only under ideal conditions is unreliable in deployment. The adaptive controller's graceful fallback to local execution converts network volatility from a failure mode into a managed contingency, which is essential for software that must stay responsive in transit, indoors, and in low-coverage regions (Ale et al., 2024).

### Scalability and Deployment Economics

Figure 4 highlights a structural advantage of distributing computation, addressing the fourth objective: centralised inference saturates as concurrent demand grows because finite server capacity becomes a bottleneck, whereas adaptive partitioning recruit’s device and edge resources that scale with the user population itself (Dhar et al., 2024). This has direct economic implications offloading a substantial share of inference away from metered cloud accelerators reduces per-request cost and defers capacity expansion. For commercial mobile products with large and growing user bases, this translates into both better tail-latency behaviour at peak and a more sustainable cost trajectory.

Privacy as a design dividend (Cai et al., 2024). Beyond performance, the architecture yields a privacy benefit that is intrinsic rather than bolted on, contributing to the fifth objective. Because the controller resolves a large fraction of requests locally and escalates only, when necessary, less raw user data traverses the network. Figure 5 captures this as a markedly higher privacy score for the edge-leaning strategies relative to Cloud-Only. In an environment of tightening data-protection expectations, keeping sensitive content on the device by default is a meaningful differentiator and reduces regulatory exposure without requiring users to trade away capability (Guo et al., 2024).

### Implications for Next-Generation Mobile Software

Taken together, the results argue for treating inference placement as a first-class, runtime-adaptive decision rather than a fixed deployment choice. The consolidated radar profile in Figure 5 shows the Adaptive Edge–Cloud strategy avoiding the sharp deficiencies that constrain every single-tier baseline, which is precisely the property

next-generation mobile solutions require: dependable responsiveness, controlled energy use, near-cloud accuracy, headroom to scale, and privacy by design (Xu et al., 2024c). For practitioners, the practical recommendation is to instrument confidence and network telemetry and let a lightweight controller arbitrate per request, shifting the engineering question from "edge or cloud?" to "how should the two cooperate?"-a framing the present evidence supports (Liu et al., 2024).

### Limitations and Future Research Directions

This study has several limitations that temper the generalisability of its conclusions. The evaluation relied on emulated network profiles and a controlled, replayed workload, which, while ensuring internal validity and reproducibility, may not capture the full variability of real-world mobile usage, including bursty multi-turn dialogues, background contention, and device thermal throttling over extended sessions. The model configurations were limited to a single quantisation–size ladder, and the offloading threshold $\tau$ was fixed rather than learned; performance is therefore reported for one operating point of a controller that could plausibly be tuned further. Because parameter count and quantization co-vary across the three tier-representative levels, the study cannot causally separate the effect of model scale from that of numerical precision; the reported model-configuration effect should be interpreted as their combined influence as they co-occur in practical deployments. Energy and cost were measured at the inference level and did not incorporate the full lifecycle overhead of edge infrastructure or model updates, and accuracy was assessed on representative tasks that may not reflect every downstream application.

Future work should address these gaps along several directions. A learned, context-aware offloading policy trained with reinforcement learning on live telemetry could adapt $\tau$ dynamically to user, task, and network state, potentially expanding the Pareto frontier identified here. Field studies on heterogeneous devices and real cellular networks would validate the resilience findings beyond emulation, while extending the model ladder to larger and mixture-of-experts architectures would test whether the adaptive advantage persists at greater capacity. Further research could integrate privacy-preserving techniques such as on-device redaction and federated personalisation into the controller, jointly optimising for confidentiality alongside latency and energy. Finally, examining multi-turn and agentic workloads, where intermediate state must be maintained across tiers, would extend the architecture toward the increasingly interactive mobile software that defines the next generation of applications. A fully crossed scale × precision design varying parameter count and quantization independently at a fixed tier would isolate each mechanism and quantify their separate and interacting contributions.

## 5. Conclusion

This study designed and empirically evaluated an adaptive edge–cloud architecture for large language model inference in next-generation mobile software, benchmarking it against cloud-only, edge-only, and static-partition baselines across deployment strategy, network condition, and model configuration. The results demonstrate that deployment strategy is the dominant determinant of performance and that the proposed confidence-driven Adaptive Edge–Cloud approach delivers the most balanced outcome: it preserves near-cloud accuracy (89.4%) while reducing mean latency by roughly a third, remains resilient under degraded networks where cloud-only latency more than quadruples, scales beyond the saturation point that limits centralised inference, and strengthens privacy by resolving most requests locally. Lying on the latency–energy Pareto front and dominating the consolidated quality-of-service profile, adaptive partitioning establishes that inference placement should be a runtime decision rather than a fixed architectural commitment reframing mobile LLM deployment from a choice between edge and cloud to a question of how the two should cooperate, and offering a practical, generalisable blueprint for responsive, efficient, scalable, and privacy-respecting mobile intelligence.

## References


[1] Akbar, M. A., Esposito, M., Hyrynsalmi, S., Kumar, K. D., Lcnarduzzi, V., Li, X., ... & Zohaib, M. (2024, August). 6gsoft: Software for edge-to-cloud continuum. In 2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 499-506). IEEE.

[2] Ale, L., Zhang, N., King, S. A., & Chen, D. (2024). Empowering generative AI through mobile edge computing. Nature Reviews Electrical Engineering, 1(7), 478-486.

[3] Cai, F., Yuan, D., Yang, Z., & Cui, L. (2024, July). Edge-llm: A collaborative framework for large language model serving in edge computing. In 2024 IEEE International Conference on Web Services (ICWS) (pp. 799-809). IEEE.

[4] Dhar, N., Deng, B., Lo, D., Wu, X., Zhao, L., & Suo, K. (2024, April). An empirical analysis and resource footprint study of deploying large language models on edge devices. In Proceedings of the 2024 ACM southeast conference (pp. 69-76).

[5] Friha, O., Ferrag, M. A., Kantarci, B., Cakmak, B., Ozgun, A., & Ghoualmi-Zine, N. (2024). Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. IEEE Open Journal of the Communications Society, 5, 5799-5856.

[6] Guo, J., Wang, M., Yin, H., Song, B., Chi, Y., Yu, F. R., & Yuen, C. (2024). Large language models and artificial intelligence generated content technologies meet communication networks. IEEE Internet of Things Journal, 12(2), 1529-1553.

[7] Han, H. (2024, December). Innovative Research on IoT Architecture and Robotic Operating Platforms: Applications of Large Language Models and Generative AI. In 2024 6th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI) (pp. 881-887). IEEE.
[8] Hang, C. N., Yu, P. D., Morabito, R., & Tan, C. W. (2024). Large language models meet next-generation networking technologies: A review. Future Internet, 16(10), 365.
[9] Laskaridis, S., Katevas, K., Minto, L., & Haddadi, H. (2024, July). Mobile and edge evaluation of large language models. In Workshop on Efficient Systems for Foundation Models II@ ICML2024.
[10] Li, P., Sánchez-Mompó, A., Farnham, T., Khan, A., & Aijaz, A. (2024). Large generative ai models meet open networks for 6g: Integration, platform, and monetization. arXiv preprint arXiv:2410.18790.
[11] Li, Z., Wang, J., Zhao, S., Wang, Q., & Wang, Y. (2025). Evolving towards artificial-intelligence-driven sixth-generation mobile networks: An end-to-end framework, key technologies, and opportunities. Applied Sciences, 15(6), 2920.
[12] Liu, F., Farkiani, B., & Crowley, P. (2024). A survey on large language models for network operations & management: applications, techniques, and opportunities. Authorea Preprints.
[13] Luo, H., Liu, Y., Zhang, R., Wang, J., Sun, G., Niyato, D., ... & Shen, X. (2025). Toward edge general intelligence with multiple-large language model (Multi-LLM): architecture, trust, and orchestration. IEEE Transactions on Cognitive Communications and Networking.
[14] Rosamma, K. S. (2024, December). Small language models and their role in hybrid AI architectures for big data analytics. In 2024 International Conference on Sustainable Communication Networks and Application (ICSCNA) (pp. 594-599). IEEE.
[15] Shen, Y., Shao, J., Zhang, X., Lin, Z., Pan, H., Li, D., ... & Letaief, K. B. (2024). Large language models empowered autonomous edge AI for connected intelligence. IEEE Communications Magazine, 62(10), 140-146.
[16] Tian, Y., Zhang, Z., Yang, Y., Chen, Z., Yang, Z., Jin, R., ... & Wong, K. K. (2024). An edge-cloud collaboration framework for generative AI service provision with synergetic big cloud model and small edge models. IEEE Network, 38(5), 37-46.
[17] Xu, J., Li, Z., Chen, W., Wang, Q., Gao, X., Cai, Q., & Ling, Z. (2024a). On-device language models: A comprehensive review. arXiv preprint arXiv:2409.00088.
[18] Xu, M., Du, H., Niyato, D., Kang, J., Xiong, Z., Mao, S., ... & Poor, H. V. (2024b). Unleashing the power of edge-cloud generative AI in mobile networks: A survey of AIGC services. IEEE Communications Surveys & Tutorials, 26(2), 1127-1170.
[19] Xu, R., Nagothu, D., & Chen, Y. (2024c). AR-edge: autonomous and resilient edge computing architecture for smart cities. In Edge Computing Architecture-Architecture and Applications for Smart Cities. IntechOpen.
[20] Zheng, X., Li, Y., Zheng, B., Zhang, C., & Zhu, L. (2025). EdgeNetLLM: Cloud-Edge Collaborative Adaptation of Large Language Models for Mobile Networking. IEEE Transactions on Network Science and Engineering.