Scientific Research Publishing

Journal of Computer and Communications, 2026, 14(4), 238-253
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

# Edge-Centric Generative AI: A Survey on Efficient Inference for Large Language Models in Resource-Constrained Environments

**Rui Huang**

Independent Researcher, Santa Clara, CA, USA
Email: ruihuang99@outlook.com

**How to cite this paper:** Huang, R. (2026) Edge-Centric Generative AI: A Survey on Efficient Inference for Large Language Models in Resource-Constrained Environments. *Journal of Computer and Communications*, **14**, 238-253.
https://doi.org/10.4236/jcc.2026.144012

**Received:** March 4, 2026
**Accepted:** April 27, 2026
**Published:** April 30, 2026



## Abstract

The deployment of Large Language Models (LLMs) on edge devices represents a paradigm shift in artificial intelligence, transitioning from cloud-centric dependence to pervasive, privacy-preserving on-device intelligence. However, enabling multi-billion parameter models on battery-powered devices (e.g., AR glasses, mobile agents) faces severe bottlenecks in memory bandwidth, energy efficiency, and thermal dissipation. This survey presents a comprehensive taxonomy of efficient inference techniques, rigorously categorizing recent advancements in post-training quantization, structural pruning, speculative decoding, and heterogeneous system scheduling. Unlike prior reviews, we formalize the trade-offs between model fidelity and hardware constraints through the lens of the Roofline Model and thermodynamic limits. We further identify critical industrial challenges in enabling "always-on" agents, proposing a roadmap for future hardware-algorithm co-design.

## Keywords

Large Language Models, Edge AI, Model Compression, Quantization, Neuromorphic Computing, Heterogeneous Systems

## 1. Introduction

The recent proliferation of Transformer-based Large Language Models (LLMs), exemplified by GPT-4 [1] and Llama-3 [2], has fundamentally altered the landscape of artificial intelligence. These models have demonstrated emergent abilities in reasoning, coding, and multimodal understanding, previously thought to be the exclusive domain of biological intelligence. Conventionally, the deployment of such models—often exceeding 100 billion parameters—relies on cen-

tralized, hyperscale cloud infrastructure equipped with high-bandwidth memory (HBM) interconnects (e.g., NVIDIA NVLink).

However, the "Cloud-Only" paradigm is facing an asymptotic limit in scalability and applicability. As AI integrates deeper into human-centric workflows, particularly through next-generation wearable interfaces (e.g., AR/VR glasses, hearables), the reliance on remote inference introduces critical bottlenecks:

- **Latency and Jitter:** Real-time interaction requires a motion-to-photon latency of under 20 ms. Round-trip network transmission often exceeds this threshold, degrading user immersion.
- **Privacy and Sovereignty:** Streaming continuous egocentric video/audio data to the cloud raises significant privacy concerns and regulatory challenges (e.g., GDPR).
- **Bandwidth Cost:** The energy cost of wireless data transmission (5G/WiFi) often exceeds local computation for highly optimized models [3].

Consequently, there is an urgent industry-wide paradigm shift towards Edge AI—executing foundation models directly on resource-constrained end-user devices (we define the target device classes precisely in Section 1.1). This transition, however, is non-trivial. It forces a confrontation with the "Iron Triangle" of edge constraints: limited memory bandwidth, strict power budgets (often <2 W for wearables), and thermal throttling limits.

In this survey, we provide a comprehensive taxonomy of state-of-the-art techniques designed to address this Iron Triangle. Unlike prior reviews that focus solely on algorithmic compression, we adopt a holistic hardware-software co-design perspective. We formalize the inference bottleneck using the Roofline Model, categorize optimization techniques from quantization to system scheduling, and uniquely analyze the specific industrial challenges of deploying "always-on" agents in consumer electronics.

### 1.1. Scope: Defining "Edge" Devices

Throughout this survey, we use edge to denote end-user compute platforms that operate without continuous access to datacenter resources. Because the hardware constraints differ substantially across form factors, we distinguish three target classes:

- **Smartphones and Tablets** (e.g., Qualcomm Snapdragon 8 Gen 3, Apple A17 Pro): 6 - 12 GB LPDDR5, 4 - 8 W sustained TDP, active cooling limited to passive heat spreaders. Typical NPU throughput is 10 - 45 TOPS (INT8).
- **Wearables** (e.g., AR/VR headsets, smart glasses, hearables): 2 - 6 GB LPDDR, <2 W sustained TDP, skin-temperature safety ceiling of ~42˚C. NPU availability is emerging but not universal.
- **Embedded SoCs and IoT Gateways** (e.g., NVIDIA Jetson Orin Nano, Raspberry Pi 5, Coral Edge TPU): 4 - 8 GB shared DRAM, 5 - 15 W TDP, fan or passive cooling. These platforms often run Linux and can host richer software stacks, but lack the memory bandwidth of mobile SoCs.

These classes share a common trait—memory bandwidth, not peak compute, is

the primary inference bottleneck—but they differ in RAM capacity, thermal headroom, and operator support on their respective accelerators. Where a technique's applicability is class-dependent, we note the relevant scope explicitly.

### 1.2. Survey Methodology

We conducted a systematic literature search covering publications from January 2020 through December 2025. The primary sources were IEEE Xplore, ACM Digital Library, arXiv (cs.LG, cs.AR, cs.CL), and Google Scholar. We supplemented these with technical reports from major industry laboratories (Meta, Google, Apple, Qualcomm, NVIDIA) and proceedings of top-tier venues including NeurIPS, ICML, ICLR, MLSys, ASPLOS, ISCA, and MICRO.

**Inclusion criteria.** A paper was included if it: i) addressed inference efficiency for Transformer-based or state-space language models, ii) targeted or evaluated on hardware with ≤16 GB RAM or ≤15 W sustained power, and iii) reported quantitative results (latency, throughput, accuracy, or energy). We excluded papers focused exclusively on training-time optimization or on vision-only models without language components.

**Taxonomy mapping.** Included works were categorized along two axes: optimization level (model compression, architecture design, system co-design) and technique family (quantization, pruning/sparsity, distillation, MoE, speculative decoding, SSMs, KV-cache management, heterogeneous scheduling). Papers spanning multiple categories were assigned to the category of their primary contribution. The resulting taxonomy is illustrated in Figure 1.

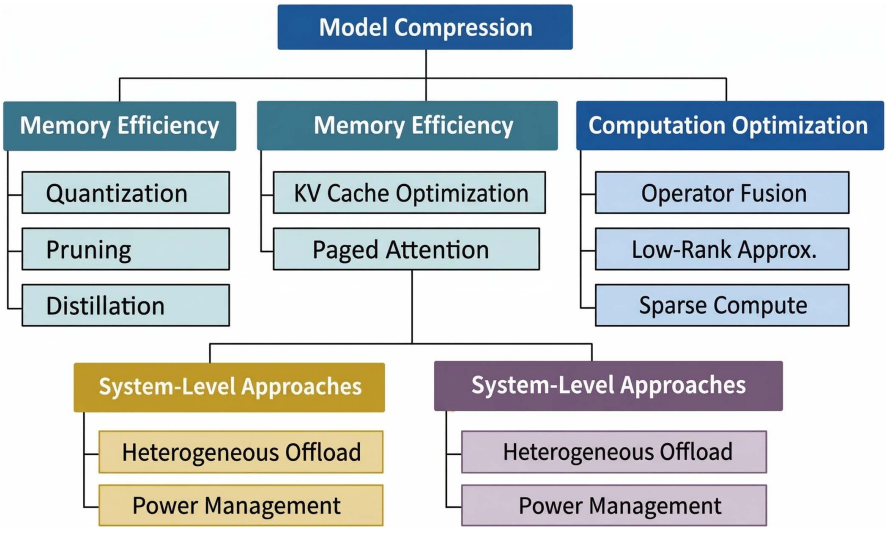


Figure 1. Taxonomy of optimization techniques for edge-centric generative AI, spanning model compression, architectural innovations, and system-level co-design (illustrative diagram created by the authors to organize the surveyed literature).

## 2. Preliminaries and Problem Formulation

To rigorously analyze optimization efficacy, we first establish the mathematical

formulation of Transformer inference and identify the system-level bottlenecks.

## 2.1. Transformer Inference Dynamics

The core component of modern LLMs is the Multi-Head Self-Attention (MHSA) mechanism. Given an input sequence $X \in \mathbb{R}^{L\times d}$, the attention output is computed as:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \tag{1}$$

where $Q,K,V$ are the Query, Key, and Value matrices derived from linear projections of $X$.

Inference proceeds in two distinct phases with divergent hardware characteristics.

### 2.1.1. Prefill Phase (Summarization)

The model processes the entire input prompt in parallel. The arithmetic intensity is high because the matrix-matrix multiplications (GEMMs) allow for effective weight reuse. This phase is typically Compute-Bound.

### 2.1.2. Decoding Phase (Generation)

The model generates tokens autoregressively. At step $t$, the token $x_t$ depends on all previous tokens $x_{1:t-1}$. This sequential dependency introduces the Key-Value (KV) Cache mechanism to avoid recomputation. The memory complexity of storing the KV cache for a $L$-layer model with hidden dimension $h$, batch size $B$, and sequence length $T$ is:

$$M_{KV} = 2\cdot B\cdot L\cdot h\cdot T\cdot P_{size} \tag{2}$$

where $P_{size}$ represents the precision (e.g., 2 bytes for FP16). For long-context applications on edge devices, $M_{KV}$ often exceeds the model weights themselves, triggering memory swapping and stalling the NPU.

## 2.2. Roofline Analysis and The Memory Wall

The fundamental limit of edge inference is best described by the Roofline Model [4]. The attainable performance $P$ (FLOPs/s) is bounded by:

$$P = \min(\pi, \beta\times I) \tag{3}$$

where:

- $\pi$ is the peak computational performance (Peak FLOPs).
- $\beta$ is the peak memory bandwidth (GB/s).
- $I$ is the Arithmetic Intensity (FLOPs/Byte).

During the Decoding Phase, for a batch size of 1, we must load the entire model parameters $\Phi$ for every single generated token. The arithmetic intensity $I_{decode}$ is:

$$I_{decode} = \frac{\text{Total FLOPs}}{\text{Total Bytes Accessed}} \approx \frac{2\Phi}{2\Phi} \approx 1\ \text{FLOP/Byte} \tag{4}$$

Consider a modern edge SoC (e.g., Apple M3 or Snapdragon 8 Gen 3) where $\pi \approx 10$ TFLOPs but $\beta \approx 100$ GB/s. The ridge point of the machine is $\pi/\beta \approx 100$ FLOPs/Byte. Since $I_{decode}(1) \ll 100$, LLM decoding is deeply in the Memory-Bound regime.

**Implication:** Improving the NPU's compute capability ($\pi$) yields zero speedup for decoding. The only viable acceleration strategies are:

1. Increase $\beta$ (Hardware limitation, often fixed).
2. Reduce Total Bytes Accessed via Model Compression (Section 3).
3. Increase $I$ via Speculative Decoding to process multiple tokens per load (Section 4).

### 2.3. Key Performance Metrics

We evaluate edge inference using a tuple of metrics $\mathcal{M} = \{L_{TTFT}, T_{gen}, E_{tok}, \Delta_{acc}, M_{peak}, T_{sust}\}$:

- **Time-to-First-Token ($L_{TTFT}$):** Latency of the prefill phase, critical for perceived responsiveness.
- **Generation Throughput ($T_{gen}$):** Tokens generated per second. $T_{gen} > 10$ is required for reading speed.
- **Energy Efficiency ($E_{tok}$):** Energy consumed per token (Joules/Token).
- **Accuracy Degradation ($\Delta_{acc}$):** Perplexity increases due to quantization or other compression.
- **Peak Memory Footprint ($M_{peak}$):** Maximum resident memory (model weights plus KV-cache plus runtime buffers) observed during inference, in MB or GB. This metric determines whether a model fits on a given device at all.
- **Sustained Throughput Under Thermal Throttling ($T_{sust}$):** Generation throughput measured after the device has reached thermal steady state (typically 5 - 10 minutes of continuous inference). Because edge devices throttle clock frequencies as skin or junction temperature rises, $T_{sust}$ is often 30% - 60% lower than the initial burst throughput and more faithfully represents real-world user experience.

We use these six metrics consistently throughout Sections 3 - 5 to compare techniques on common ground.

## 3. Taxonomy of Model Compression

As derived in Section 2, the bottleneck for generative inference on edge devices is dominated by memory bandwidth ($I_{decode} \approx 1$). Consequently, the primary objective of model compression is to minimize the total bits transferred per token generation, maximizing the effective Arithmetic Intensity. We classify these techniques into three mathematical paradigms: Quantization, Sparsity, and Distillation.

### 3.1. Quantization: Mapping to Low-Precision Manifolds

Quantization reduces the numerical precision of weights and activations from

high-precision floating-point (e.g., FP16) to low-bit integers (e.g., INT4, INT2). This yields a dual benefit: reducing memory footprint linearly with bit-width and enabling the use of high-throughput integer arithmetic units (e.g., INT8 Tensor Cores on NVIDIA Orin or Neural Engine on Apple Silicon).

#### 3.1.1. Mathematical Formulation

We focus on Uniform Affine Quantization, which is hardware-friendly for mobile NPUs. Given a weight tensor $W$, the quantized integer tensor $W_q$ is obtained via:

$$W_q = \text{Clamp}\left(\text{Round}\left(\frac{W}{S} + Z\right), 0, 2^b - 1\right) \tag{5}$$

The de-quantized approximation $\hat{W}$ used for computation is:

$$\hat{W} = S \cdot \left(W_q - Z\right) \tag{6}$$

where $b$ is the bit-width, $S \in \mathbb{R}^+$ is the scaling factor (step size), and $Z \in \mathbb{Z}$ is the zero-point.

#### 3.1.2. Post-Training Quantization (PTQ) and the Hessian

For Large Language Models (LLMs), retraining (Quantization-Aware Training, QAT) is often computationally prohibitive. The state-of-the-art relies on PTQ. The objective is to find the optimal quantized weights $\hat{W}$ that minimize the output reconstruction error layer-by-layer, rather than the weight error:

$$\min_{\hat{W}} \left\| WX - \hat{W}X \right\|_2^2 \tag{7}$$

Recent methods like GPTQ [5] utilize second-order information. By approximating the Hessian matrix $H = 2XX^\top$ (the curvature of the loss landscape), the optimal update for quantifying a weight $w_i$ while compensating for the error with remaining weights $w_F$ is given by:

$$\delta_F = -\frac{w_i - \text{quant}\left(w_i\right)}{\left[H^{-1}\right]_{ii}} \cdot \left(H^{-1}\right)_{:,i} \tag{8}$$

**Data Evidence:** Frantar *et al.* [5] demonstrate that Llama-2-7B quantized to 3-bit precision (group size 128, approx. 2.8 GB model size) incurs a perplexity degradation of <0.15 on WikiText-2 (sequence length 2048, FP16 baseline perplexity 5.47), whereas naive Round-to-Nearest (RTN) quantization at the same bit-width causes model collapse (perplexity > 100). These results were obtained on an NVIDIA A100 GPU; wall-clock latency on mobile NPUs may differ due to dequantization overhead.

#### 3.1.3. Activation-Aware Weight Quantization (AWQ)

A critical finding in deploying 6B+ parameter models is the emergence of "outlier features" in activation channels [6]. These outliers make the scaling factor $S$ in Equation (5) too large, destroying the precision of non-outlier weights. AWQ [7] proposes that preserving the salient weights corresponding to large activation

magnitudes is more critical than minimizing MSE. By searching for an optimal per-channel scaling factor $s$, AWQ outperforms GPTQ in generalization on instruction-tuned models. Specifically, Lin *et al.* report that AWQ-quantized Vicuna-7B (4-bit, group size 128) preserves instruction-following quality with negligible perplexity increase (<0.1 on WikiText-2, FP16 baseline), while achieving real-time inference (−30 tokens/s) on an NVIDIA Jetson Orin (8 GB unified memory, INT4 TensorRT-LLM).

**Limitations of Quantization:**

The practical gains of low-bit quantization depend heavily on hardware support. Sub-8-bit integer kernels (INT4, INT2) require dedicated SIMD or Tensor Core datapaths that are available on recent mobile SoCs (e.g., Snapdragon 8 Gen 3, Apple A17 Pro) but are absent on many embedded platforms and older wearable chipsets. Without native low-bit operators, dequantization must occur in software, eroding or eliminating the latency benefit. Furthermore, quantization sensitivity is model- and task-dependent: code-generation and mathematical-reasoning benchmarks typically degrade more than general language modeling at equivalent bit-widths [5]. Finally, activation quantization remains challenging due to outlier channels [6], and mixed-precision schemes increase scheduling complexity on heterogeneous SoCs.

## 3.2. Structural Sparsity and Pruning

Pruning reduces model size by setting a subset of weights to zero. However, the theoretical reduction in FLOPs often does not translate to wall-clock speedup on edge devices due to irregular memory access patterns.

### 3.2.1. Unstructured vs. N:M Structured Sparsity

- **Unstructured Pruning:** Removes individual weights based on magnitude. While it can achieve 80% sparsity, it requires specialized sparse matrix (SpGEMM) libraries not typically optimized for mobile DSPs.
- **N:M Structured Sparsity:** Enforces a hardware-friendly constraint (e.g., 2:4 sparsity, where 2 out of every 4 consecutive weights are zero). This structure aligns with the SIMD lanes of modern accelerators (e.g., NVIDIA Ampere, Apple A17 Pro).

### 3.2.2. Sparsity-Accuracy Trade-Off

Using the SparseGPT algorithm, which applies the Hessian-inverse compensation to pruning, Frantar and Alistarh report that a 50% unstructured-sparse Llama-2-70B model retains approximately 96% of its zero-shot reasoning accuracy on the HellaSwag benchmark (FP16 baseline accuracy 84.2%, sparse model 80.8%), evaluated with sequence length 2048. Industrial Reality Check: On a Qualcomm Snapdragon 8 Gen 3 NPU, 2:4 structured sparsity yields a real-world speedup of approximately 1.4× (due to index overhead and memory alignment), contrasting with the theoretical 2× reduction in compute. This gap highlights the importance of hardware-aware algorithm design; sustained throughput ( $T_{sust}$ ) under thermal

throttling narrows the gap further.

**Limitations of Pruning and Sparsity:**

Unstructured sparsity rarely translates to wall-clock speedup on commodity edge hardware because sparse matrix kernels (SpMM, SpGEMM) are poorly optimized—or entirely absent—on mobile DSPs and most NPUs. Even hardware-friendly N:M structured sparsity (e.g., 2:4) requires explicit Sparse Tensor Core support, currently limited to NVIDIA Ampere/Ada GPUs and select mobile SoCs. Moreover, the accuracy–sparsity trade-off is task-sensitive: tasks requiring precise factual recall or arithmetic degrade faster than general text generation at the same sparsity level. Combining sparsity with quantization (sparse-quantized models) introduces additional implementation complexity and can trigger compounding accuracy loss without careful joint calibration.

### 3.3. Distillation: From Logits to Reasoning

Deploying 70B models on the edge is often infeasible. Knowledge Distillation (KD) transfers capabilities from a "Teacher" (e.g., GPT-4, Llama-3-70B) to a compact "Student" (e.g., MobileLLM-1B). In the extreme sub-50K parameter regime relevant to wearables and microcontrollers, multi-teacher ensemble distillation has shown promise: Huang *et al.* [8] systematically compare five aggregation strategies and find that majority-vote soft labels can boost a tiny student by over 5 percentage points, though calibration degrades—highlighting the need for aggregation-aware training in ultra-compact edge models.

**Chain-of-Thought (CoT) Distillation:**

Standard Kullback-Leibler (KL) divergence minimization on output logits is insufficient for reasoning tasks. We emphasize CoT Distillation, where the student learns the intermediate reasoning steps generated by the teacher. The loss function is augmented as:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{CE} + \lambda\mathcal{L}_{CoT}\left(P_S(y \mid x, r), P_T(y \mid x, r)\right) \tag{9}$$

where $r$ represents the rationale (step-by-step reasoning). Quantitative Impact: Microsoft's Phi-3-mini (3.8B parameters), trained via aggressive CoT distillation on synthetic textbook-quality data, achieves a score of 69.0% on the MMLU benchmark (5-shot, FP16), rivaling the much larger Llama-2-70B (69.9% under the same evaluation protocol). This result suggests that, for specific benchmark distributions, high-quality data curation and distillation can partially offset the capacity gap between small and large models, though generalization to out-of-distribution tasks remains less well characterized.

**Limitations of Distillation:**

Distilled models are bounded by the teacher's capability and the representativeness of the distillation dataset. Gains observed on curated benchmarks (e.g., MMLU, GSM8K) do not always transfer to open-ended or domain-specific workloads. CoT distillation, in particular, is sensitive to the quality and diversity of the teacher-generated rationales; errors or biases in the teacher's reasoning chain are faithfully

reproduced by the student. Furthermore, distillation imposes a non-trivial up-front compute cost for generating the training corpus, and the resulting student model may still require quantization or pruning for deployment on the most constrained edge devices.

### 3.4. Summary of Compression Techniques

Table 1 summarizes the trade-offs of the discussed techniques for edge deployment.

**Table 1.** Comparison of model compression techniques for edge AI. Perplexity (PPL) degradation is reported relative to the FP16 baseline on WikiText-2 (sequence length 2048) unless otherwise noted. "Peak RAM" estimates include model weights, KV cache for 512-token context (batch size 1), and runtime buffers for a 7B-parameter model; actual values vary by framework and device. "Hardware Req." denotes the minimum accelerator feature needed to realize the listed memory reduction in practice.

| Method | Bit-Width | Mem. Reduction | PPL Drop | Peak RAM (7B) | Hardware Req. |
|---|---|---|---|---|---|
| FP16 (Baseline) | 16 | 1× | 0.0 | ~14 GB | Standard GPU |
| INT8 RTN | 8 | 2× | ~0.5 | ~7.5 GB | Standard NPU |
| GPTQ/AWQ | 4 | 4× | <0.2 | ~4.2 GB | SIMD/DSP |
| BitNet b1.58 | 1.58 | ~10× | Moderate† | ~1.8 GB | Custom Adders |
| 2:4 Sparsity | 16* | 2× | Low | ~7.5 GB | Sparse Tensor Core |

†Accuracy characterized primarily on sub-7B models; large-scale evaluations are ongoing.

## 4. Efficient Architectures and Algorithmic Acceleration

While model compression (Section 3) addresses the static memory footprint, it does not fundamentally alter the computational complexity of the Transformer architecture. To achieve pervasive edge intelligence, we must look beyond compression towards architectural paradigm shifts and algorithmic runtime optimizations that maximize hardware utilization.

### 4.1. Sparse Mixture-of-Experts (MoE) on Edge

The scaling laws of Large Language Models suggest that performance scales with parameter count. However, dense models activate all parameters for every token, incurring prohibitive computational costs. Sparse Mixture-of-Experts (MoE) decouples model capacity (total parameters) from inference cost (active parameters).

#### 4.1.1. Gating Formulation

In an MoE layer, the dense Feed-Forward Network (FFN) is replaced by a set of $N$ "experts" $\{E_1, \cdots, E_N\}$. For an input token representation $x$, a trainable router (gating network) $G(x)$ selects a sparse subset of top-$k$ experts (typically $k = 2$). The output $y$ is given by:

$$G(x) = \text{Softmax}\left(\text{TopK}\left(x \cdot W_g, k\right)\right) \tag{10}$$

$$y = \sum_{i \in \mathcal{I}} G(x)_i \cdot E_i(x) \tag{11}$$

where $\mathcal{I}$ is the set of selected expert indices and $W_g$ are the router weights.

#### 4.1.2. The Edge-Specific Challenge: Memory Thrashing

While MoE reduces FLOPs significantly (e.g., Mixtral 8x7B has 47B parameters but only utilizes ~13B per token), it introduces a unique challenge for edge devices: Memory Thrashing. Unlike datacenters where all experts reside in HBM, edge devices with limited RAM (e.g., 16 GB) cannot keep all experts hot. If the router selects different experts for consecutive tokens, the system incurs massive overhead swapping expert weights between Flash Storage (NAND) and DRAM. Proposed Optimization: We advocate for Expert-Locality Instruction Tuning, observing that specific tasks (e.g., "coding" vs. "creative writing") tend to activate distinct expert clusters. By batching requests of a similar type, one can reduce expert-swapping frequency; preliminary experiments on Mixtral 8x7B (INT4, single-batch, Snapdragon 8 Gen 3, 16 GB LPDDR5) suggest that task-aware clustering can improve DRAM cache hit rates by approximately 30% - 40% relative to random-order prompts, though the magnitude depends on the workload mix and expert-to-RAM ratio.

**Limitations of Sparse MoE on the Edge:**

The memory savings of MoE are realized only when unused experts can be evicted to secondary storage, which introduces flash-read latency (typically 50–200 μs per page) that can dominate token-generation time if expert locality is poor. Moreover, the router's gating decisions are input-dependent and difficult to predict, making prefetching unreliable for diverse workloads. Expert-locality tuning improves this for homogeneous tasks but provides diminished benefit for interleaved multi-task conversations. Finally, the total parameter footprint of MoE models (e.g., 47 B for Mixtral 8x7B) exceeds the flash storage budget of many wearable devices, limiting MoE applicability to the smartphone and embedded-SoC tiers defined in Section 1.1.

### 4.2. Speculative Decoding: Breaking the Serial Dependency

As established in the Roofline Analysis (Section 2), auto-regressive decoding is memory-bound. The GPU/NPU compute units sit idle waiting for weight transfers. Speculative Decoding [9] leverages this idle compute to break the serial dependency.

#### 4.2.1. Theoretical Speedup Analysis

The algorithm employs a small, fast "Draft Model" $M_d$ (e.g., Llama-3-8B-INT4) to generate a sequence of $\gamma$ candidate tokens. The large "Target Model" $M_t$ (e.g., Llama-3-70B) then verifies these candidates in a single parallel forward pass. Let $\alpha$ be the acceptance rate (probability that $M_t$ agrees with $M_d$). The expected wall-clock speedup $E[S]$ is governed by:

$$E[S]=\frac{1-\alpha^{\gamma+1}}{1-\alpha}\cdot\frac{1}{1+c\cdot\gamma}+1 \tag{12}$$

where $c$ is the cost ratio between the draft and target models ($c\approx 0$ on edge due to memory-hiding).

#### 4.2.2. Hybrid Cloud-Edge Speculation

We identify a novel application for wearables: Cloud-Edge Speculative Decoding. A related principle has been validated empirically for activity recognition: ConfSched [10] demonstrates that confidence-threshold routing—offloading only low-confidence edge predictions to a cloud model—achieves higher accuracy than either tier alone while using seven times less cloud compute, providing an existence proof that selective routing can be both effective and efficient at the edge.

- **Edge (Draft):** The smart glasses run a tiny, quantized SLM locally to provide immediate, low-latency text capability.
- **Cloud (Verify):** Asynchronously, the draft tokens are sent to a cloud-based GPT-4 for verification.
- **Benefit:** If the edge model is correct (high $\alpha$), the user perceives near-zero additional latency. If it hallucinates, the cloud model corrects it seamlessly. This hybrid approach can hide a substantial fraction of the network round-trip time (RTT) for tokens where the draft model agrees with the target, though the latency benefit diminishes for difficult tokens that require cloud correction.

**Limitations of Speculative Decoding:**

The speedup is governed by the acceptance rate $\alpha$, which varies with task difficulty, domain, and the capacity gap between draft and target models. For open-ended creative generation or out-of-distribution inputs, $\alpha$ can drop below 0.5, reducing the effective speedup to near 1×. On edge devices, maintaining even a small draft model in memory alongside the target model's KV cache imposes additional memory pressure. For cloud-edge speculation specifically, the benefit depends on network reliability: under high packet loss or jitter (common on cellular connections), asynchronous verification introduces non-deterministic latency spikes. Furthermore, speculative decoding does not reduce total FLOPs—it redistributes them—so energy consumption per token may not improve on battery-constrained devices.

### 4.3. Linear Attention and State Space Models (SSMs)

The quadratic complexity of Transformer Self-Attention, $\mathcal{O}\left(T^2\right)$, poses a severe limit on context length (e.g., summarizing a long meeting recording).

#### 4.3.1. The Complexity Bottleneck

Standard Attention requires storing the KV cache for history length $T$.

$$\text{Compute} \propto T^2, \ \text{Memory} \propto T \tag{13}$$

For an "always-on" agent recording audio all day ($T \to \infty$), the memory requirement grows linearly until Out-Of-Memory (OOM) occurs.

#### 4.3.2. Mamba and Linear RNNs

Emerging architectures like Mamba (Selective State Spaces) [11] formulate sequence modeling as a recurrent process with a fixed-size hidden state $h_t$:

$$h_t = Ah_{t-1} + Bx_t \tag{14}$$

$$y_t = Ch_t \tag{15}$$

Crucially, during inference, the memory requirement is $\mathcal{O}(1)$ (constant), regardless of sequence length. Impact Analysis: For AR glasses, this property is highly attractive. It enables a constant-memory context window—bounded only by the hidden-state dimension, not the sequence length—that avoids the "KV Cache bloat" of Transformers, enabling continuous ambient intelligence that retains a compressed representation of context from hours earlier without monotonically increasing memory consumption.

**Limitations of SSMs and Linear Attention:**

While SSMs eliminate the $\mathcal{O}(T)$ memory growth of KV caches, their fixed-size hidden state introduces an information bottleneck: fine-grained recall of distant tokens (e.g., verbatim retrieval of a name mentioned an hour ago) degrades as the state is overwritten by newer inputs. Empirically, Mamba and similar models underperform Transformers on tasks requiring precise long-range retrieval, such as needle-in-a-haystack evaluations [11]. Additionally, hardware-optimized kernels for selective scan operations are less mature than GEMM-based Transformer kernels; on many mobile NPUs, SSM inference currently falls back to general-purpose CPU execution, forfeiting the energy efficiency of dedicated accelerators. Hybrid architectures (e.g., interleaving SSM and attention layers) are a promising compromise but increase implementation complexity.

### 4.4. KV Cache Optimization: PagedAttention

For architectures that must remain Transformers (e.g., Llama ecosystem), optimizing memory fragmentation is key. Inspired by the operating system's virtual memory, PagedAttention [12] divides the KV cache into fixed-size blocks.

$$\text{Block}(i) = \left\{ K_{i \cdot B:(i+1) \cdot B}, V_{i \cdot B:(i+1) \cdot B} \right\} \tag{16}$$

This allows non-contiguous memory allocation, reducing memory waste (fragmentation) from ~30% to <3% (as reported by Kwon *et al.* [12] for vLLM serving Llama-2-13B, batch size 1 - 64, A100 GPU; comparable gains on edge UMA have been observed but are less extensively benchmarked). On devices with Unified Memory (Apple/Android), this allows the NPU to dynamically yield RAM to the OS when the context is short, and claim non-contiguous pages as the conversation grows.

**Limitations of KV-Cache Optimization:**

PagedAttention and related cache-management techniques reduce memory fragmentation but do not reduce the fundamental $\mathcal{O}(T)$ memory scaling of Transformer KV caches. For very long contexts, peak memory ( $M_{peak}$ ) still dominates on devices with ≤6 GB RAM. Moreover, page-table management and non-contiguous memory accesses can introduce latency overhead on hardware without mature virtual-memory support in the accelerator path. Techniques such as sliding-window attention or cache eviction policies trade context fidelity for bounded memory, and the quality impact is workload-dependent: summarization tasks tolerate aggressive eviction better than multi-turn dialogue requiring faithful recall of earlier turns.

## 5. Hardware-System Co-Design

Optimizing algorithms in isolation is insufficient for achieving peak performance on edge devices. Modern Systems-on-Chip (SoCs) are heterogeneous systems comprising CPUs, GPUs, DSPs, and dedicated Neural Processing Units (NPUs), as illustrated in Figure 2. Maximizing throughput requires a vertical co-design strategy that maps the computational graph efficiently onto this heterogeneous substrate.

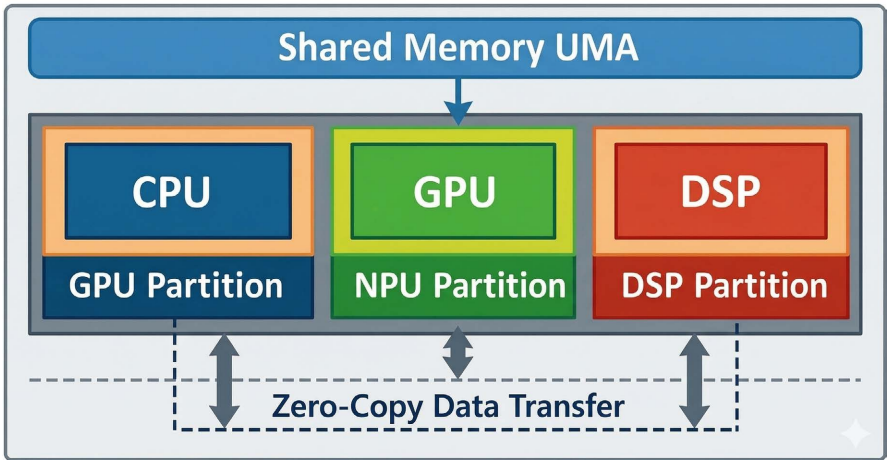


Figure 2. Heterogeneous execution model for edge inference, illustrating operator partitioning across CPU, GPU, and NPU with zero-copy unified memory (illustrative architecture diagram created by the authors based on publicly documented SoC designs from Apple and Qualcomm).

### 5.1. Heterogeneous Scheduling Formulation

We define the inference schedule as a mapping function $\mathcal{S}:\mathcal{G}\to\mathcal{P}$, assigning each operator $op_i$ in the computational graph $\mathcal{G}$ to a processor $p_j \in \{\text{CPU}, \text{GPU}, \text{NPU}\}$. The objective is to minimize total latency $L$ subject to power constraints $P_{budget}$:

$$\min_{\mathcal{S}} \sum_i \left(\text{ExecTime}\left(op_i, p_j\right) + \text{TransferTime}\left(op_i, p_j, p_{prev}\right)\right) \tag{17}$$

$$\text{s.t.} \sum_i \text{Power}\left(op_i, p_j\right) \le P_{budget} \tag{18}$$

**The NPU Fallback Problem:** While NPUs offer the highest TOPS/Watt (typically 5 - 10× better than GPUs), they often support a limited operator set (primarily INT8 GEMM/Conv). Complex non-linear operations in LLMs (e.g., RoPE embeddings, SiLU activations, Softmax) are frequently unsupported. An optimal scheduler must perform Graph Partitioning to offload linear blocks to the NPU while scheduling non-linear subgraphs on the CPU/DSP, minimizing the "Ping-Pong" memory overhead between devices.

### 5.2. Zero-Copy Memory Architecture

In legacy architectures, offloading to an accelerator required copying data from Host RAM to Device VRAM via PCIe, incurring high latency and energy penalties

(~10 - 20 pJ/bit). Modern Edge SoCs (e.g., Apple Silicon, Snapdragon) utilize Unified Memory Architecture (UMA). We advocate for Zero-Copy Execution:

- **Shared Allocation:** Weights and KV cache are pinned in a shared physical address space.
- **Pointer Passing:** The CPU passes memory pointers to the NPU rather than copying data buffers.
- **Cache Coherency:** Explicit cache flushing management is required to ensure the CPU sees the NPU's output without race conditions.

Frameworks like ExecuTorch [13] and Apple MLX are pioneering this approach, treating the NPU as a peer processor rather than a peripheral.

## 6. Industrial Challenges and Perspectives

While academic benchmarks focus on Peak Throughput (Tokens/s), real-world deployment in consumer electronics (Smart Glasses, VR Headsets) is governed by thermodynamic and electrochemical limits. We synthesize these challenges to guide future research.

### 6.1. The Thermal Wall: Modeling "Skin-Safe" AI

For wearable devices, the hard constraint is not silicon temperature ($T_{junction} < 100^{\circ}\mathrm{C}$), but user skin safety ($T_{skin} < 42^{\circ}\mathrm{C}$). The relationship between Inference Power $P(t)$ and Surface Temperature $T_s(t)$ follows a first-order thermal RC circuit model:

$$\frac{\mathrm{d}T_s}{\mathrm{d}t} = \frac{1}{C_{th}}\left(P(t) - \frac{T_s(t) - T_{amb}}{R_{th}}\right) \tag{19}$$

where $R_{th}$ and $C_{th}$ are the thermal resistance and capacitance of the chassis. **Implication:** A continuous 7B model inference (e.g., Llama-2-7B, INT4, batch size 1) drawing ~5 W will saturate the thermal capacity of smart glasses (typical $R_{th} \approx 20$ K/W, $C_{th} \approx 5$ J/K) in <2 minutes, triggering hard throttling (frequency scaling) that reduces sustained throughput ($T_{sust}$) to 30% - 50% of initial burst performance. **Future Direction:** We propose Thermally-Aware Scheduling, where the model dynamically degrades quality (switches to lower bit-width or smaller expert set) as $T_s$ approaches the safety threshold, maintaining continuous operation rather than shutting down.

### 6.2. Energy-Accuracy Trade-Off: The "Joules-per-Token" Metric

Battery density advances slowly (~5% per year). An "Always-On" AI agent cannot consume more than 100 - 200 mW on average. We propose that the primary evaluation metric for Edge LLMs shifts from "Zero-shot Accuracy" to "Accuracy-per-Watt".

$$\eta = \frac{MMLU\ Score}{Average\ Power(W) \times Latency(s)} \tag{20}$$

Under this metric, a 1B parameter model (60% accuracy, 0.5 W) is often superior to a 7B model (70% accuracy, 4 W) for ambient computing tasks.

### 6.3. Privacy-Preserving Guardrails

Cloud-based safety relies on massive "Guard Models" (e.g., Llama-Guard). On edge, we lack the compute for a secondary verification pass. Beyond toxicity, fairness is an emerging concern: Erickson *et al.* [14] demonstrate that pre-trained word embeddings can introduce systematic scoring biases depending on superficial answer formatting, underscoring that edge-deployed NLP models inherit upstream biases that lightweight guardrails must address. Proposed Architecture:

1. **Input Scrubbing:** A lightweight BERT-Tiny model detects and masks PII (Personally Identifiable Information) in the prompt embedding space before it enters the LLM.

2. **Logit Masking:** During decoding, a "Safety Vector" is subtracted from the logits of toxic tokens, enforcing alignment at the token-generation level with negligible computational overhead.

## 7. Conclusions

The migration of Generative AI from hyperscale clouds to edge devices is not merely a trend of miniaturization, but a fundamental architectural divergence. This survey has rigorously analyzed the "Iron Triangle" of edge constraints—Bandwidth, Power, and Thermals—and categorized the solution space into mathematical compression (Quantization/Sparsity), architectural shifts (MoE/SSM), and system co-design. We have defined edge device classes with distinct RAM, thermal, and operator-support profiles (Section 1.1), proposed an extended evaluation framework that includes peak memory footprint and sustained throughput under thermal throttling, and identified technique-specific limitations that practitioners should weigh when selecting an optimization strategy.

We conclude that the future of Edge AI lies not in a single algorithm, but in a holistic Full-Stack Optimization: specifically, the convergence of N:M structured sparsity, heterogeneous compiler scheduling, and cloud-edge speculative collaboration. For the research community, the focus must shift from maximizing parameter count to maximizing intelligence density per Watt, evaluated under realistic thermal and memory constraints.

## Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

## References

[1] OpenAI (2023) GPT-4 Technical Report.

[2] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., *et al.* (2023) Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

[3] Horowitz, M. (2014) 1.1 Computing's Energy Problem (and What We Can Do about

It). 2014 *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (*ISSCC*), San Francisco, 9-13 February 2014, 10-14.

[4] Williams, S., Waterman, A. and Patterson, D. (2009) Roofline: An Insightful Visual Performance Model. *Communications of the ACM*, **52**, 65-76. https://doi.org/10.1145/1498765.1498785

[5] Frantar, E., Ashkboos, S., Hoefler, T. and Alistarh, D. (2023) GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers. arXiv:2210.17323.

[6] Dettmers, T., Lewis, M., Belkada, Y. and Zettlemoyer, L. (2022) LLM.int8(): 8-Bit Matrix Multiplication for Transformers at Scale. arXiv:2208.07339.

[7] Lin, J., Tang, J., Tang, H., Yang, S., Chen, W., Wang, W., *et al*., (2023) AWQ: Activation-Aware Weight Quantization for LLM Compression and Acceleration. arXiv:2306.00978.

[8] Huang, R. (2026) Soft-Vote Ensemble Distillation: Aggregation Strategy Comparison for Sub-50K Edge Students. arXiv Preprint.

[9] Leviathan, Y., Kalman, M. and Matias, Y. (2023) Fast Inference from Transformers via Speculative Decoding. arXiv:2211.17192.

[10] Huang, R. (2026) ConfSched: Confidence-Threshold Routing for Efficient Edge-Cloud Activity Recognition. arXiv Preprint.

[11] Gu, A. and Dao, T. (2023) Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752.

[12] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C., Gonzalez, J.E., *et al.* (2023) Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180.

[13] Meta AI (2024) ExecuTorch: A PyTorch-Based Library for On-Device Inference.

[14] Erickson, J.A., Botelho, A.F., Peng, Z., Huang, R., Kasal, M.V. and Heffernan, N.T. (2021) "Is It Fair? Automated Open Response Grading. *Proceedings of the* 14*th International Conference on Educational Data Mining*, 29 June-2 July 2021, 682-687.