CoSense-LLM: Semantics-at-the-Edge with Cost- and Uncertainty-Aware Cloud–Edge Cooperation

Hasan Akgul
Department of Computer Engineering
Istanbul Technical University (ITU)
İTÜ Ayazaga Campus, 34469, Maslak/Istanbul, Turkey
akgulh20@itu.edu.tr

Mari Eplik
Institute of Computer Science
University of Tartu
J. Liivi 2, 50409 Tartu, Estonia
mari.eplik@ut.ee

Javier Rojas
Department of Computer Science
University of Chile
Beauchef 851, Santiago, Chile
jrojas@dcc.uchile.cl

Aina Binti Abdullah
Faculty of Computing
Universiti Teknologi Malaysia
81310 Skudai, Johor, Malaysia
aina.abdullah@utm.my

Pieter van der Merwe
Dept. of Electrical & Electronic Engineering
Stellenbosch University
Banghoek Rd, Stellenbosch, 7600, South Africa
pvdm@sun.ac.za

***Abstract*—We present *CoSense-LLM*, an edge-first framework that converts continuous multimodal sensor streams (e.g., Wi-Fi CSI, IMU, audio, RFID, lightweight vision) into compact, verifiable semantic tokens and coordinates with large language models (LLMs) under explicit latency, energy, bandwidth, and privacy constraints. CoSense-LLM comprises four components: (i) SenseFusion, a lightweight encoder that aligns sensor embeddings with language and compresses them into short, discrete code sequences; (ii) Edge-RAG, a local hybrid retrieval layer that grounds generation in site-specific policies and notes; (iii) PromptRouter, a cost- and uncertainty-aware policy that selects among edge-only generation, edge+retrieval, or compact cloud escalation; and (iv) Secure Execution, an auditable redaction path that enforces data minimization so raw waveforms never leave the device. The system is compatible with modern serving optimizations—including paged/streaming KV caches, FlashAttention-style kernels, speculative decoding, and quantized LoRA adapters—and supports on-device personalization and federated updates under non-IID drift. Across home, office, and clinic deployments with day-long streams, CoSense-LLM delivers grounded explanations while meeting tight service-level objectives: it sustains sub-second (p95) end-to-end latency on edge-dominant paths, reduces inter-tier token and bandwidth costs by preferring local, retrieval-grounded responses, and preserves privacy by transmitting only discrete codes and redacted metadata. Ablations show that Edge-RAG improves factual consistency and reduces contradictions, calibrated uncertainty enables selective abstention and controlled escalations, and KV/decoding accelerators materially lower energy per decision. The results support an edge-first design that treats semantics, privacy, and predictable latency as co-equal goals for large-model deployments in interference-prone environments.**

***Index Terms*—Edge intelligence, large language models, cloud–edge collaboration, retrieval-augmented generation, multimodal sensing, Wi-Fi CSI, RFID, uncertainty-aware routing, quantization, KV cache, speculative decoding, LoRA/QLoRA, federated learning, privacy and auditability**

I. Introduction

Edge intelligence is moving from a vision of offloading a few convolutional layers to a concrete capability of running, splitting, and personalizing large language and multimodal models close to where data is produced. Recent surveys and system papers converge on two forces behind this shift: (i) the need to reduce end-to-end latency and bandwidth while respecting privacy; and (ii) the emergence of algorithm–system co-design that compresses models, manages memory, and orchestrates collaborative execution across device, edge, and cloud [19], [30], [46], [58]. Cloud–edge collaboration for LLMs is now studied not only at the scheduler level but also at the model-graph level, enabling partitioned or cooperative inference and training [22], [26], [43], [70]. Meanwhile, pervasive sensing applications—e.g., Wi-Fi CSI and RFID based activity or gesture understanding—motivate low-leakage, low-latency interpretation pipelines at the network edge rather than sending raw signals to distant servers [9], [29].

A central challenge in edge LLM serving is memory and state management under tight resources. Modern serving stacks rely on paged key–value (KV) caches, dynamic cache admission/eviction, and streaming to decouple prefill and decode without stalling the device [7], [12], [15], [24], [37], [47], [71]. Kernel-level optimizations such as FlashAttention-2 and GPU-side decoding pipelines further increase throughput [6], [11]. To keep conversational quality under constrained context windows, streaming designs maintain "attention sinks" and consistent states across long sessions [44]. System-level disaggregation and scheduling control prefill–decode tradeoffs and cluster utilization [16], [23], [54], while shadow and hierarchical caches improve long-context efficiency [41]. These techniques are complementary to robustness-oriented sensing pipelines that must survive interference or distribution drifts before any language reasoning happens [5], [13].

Offloading and storage-aware execution provide another axis of feasibility for edge scenarios. Single-GPU and heterogeneous CPU/NVMe pipelines amortize model state movement [14], [35], and performance-model-guided offloaders decide tensor placement and parallelism under device limits [28]. In-storage attention and serverless runtimes reduce hot-standby costs while keeping tail latency controlled for bursty workloads at the network edge [8], [50]. Beyond placement,

inference-time acceleration leverages program-style LM execution [1], speculative decoding with verification or multi-head drafting [31], [40], [53], [72], token-utility–aware strategies [36], [73], and kernel-level attention improvements already noted [11]. These advances complement asynchronous and communication-efficient learning frameworks that adapt to edge heterogeneity during training [51], [55].

Personalization and compression are equally decisive at the edge. Low-rank adaptation and its quantized fine-tuning variant enable per-tenant specialization within tight memory budgets [20], [74]. Post-training quantization and activation-aware schemes, combined with serving-time co-design, bring 4–8-bit regimes into production without prohibitive quality loss [2], [4], [10]. Multi-tenant adapter serving avoids cold-start and context bloat by dynamically selecting adapters for tenants and tasks [3], [52]. In parallel, federated and semi-supervised edge learning methods tailor encoders and adapters for user or site-specific distributions [17], [60], [61], [75]–[77]. Such ingredients are crucial when the edge must understand noisy signals or site-specific semantics before handing off compact evidence to an LLM.

Retrieval-augmented generation (RAG) at the edge addresses knowledge staleness and privacy by grounding responses in local corpora and telemetry. Online indexing and cacheable knowledge chunks improve hit rates under intermittent connectivity [18], [33]. Hardware–algorithm co-design shows how computing-in-memory can robustly realize RAG primitives close to sensors [78], while collaborative or decentralized serving amortizes expensive backbone layers across peers [26]. Bridging continuous signals and language further benefits from instruction-tuned multimodal models that map non-textual inputs to textual interfaces [39], [57]. This is especially relevant for Wi-Fi/IMU/RFID/vision scenarios such as attention-based gesture recognition, unobtrusive emotion inference, pulmonary function sensing, open-set gestures, and affective modeling under label noise [21], [25], [32], [34], [45], as well as RFID-based writing and identity sensing in the wild [48].

Security and privacy considerations sharpen the case for edge-first pipelines. Physical-layer authentication can be subverted via fingerprint-level attacks, motivating local verification, data minimization, and verifiable reasoning where possible [42]. Side-channel risks such as acoustic keystroke eavesdropping also argue for on-device processing and careful prompt/trace handling to limit raw data exposure [38]. In response, modern edge intelligence stacks pair robust signal encoders with privacy-preserving learning and adapter isolation, and exploit streaming or shard-based execution to avoid shipping sensitive context to the cloud [14], [22], [43], [70].

Against this backdrop, our work targets low-latency, privacy-preserving event understanding that converts raw, noisy multimodal signals into compact semantic evidence at the edge, and coordinates with large models only when necessary. We build on scheduling and cache-management advances [7], [12], [15], [24], [37], [47], [71], offloading and kernel pipelines [6], [11], [14], [16], [23], [28], [35], [54], and decoding accelerators [31], [36], [40], [53], [72], [73], while weaving in federated and decentralized optimization to cope with dynamic, non-IID edge populations [56], [63], [64], [66]. Our adapter pathway embraces quantized and LoRA-style fine-tuning for personalization [2], [4], [10], [20], [74], with federated variants to mitigate forgetting and communication load [62], [68]. For sampling and pipeline stability, we employ robust token selection and near bubble-free end–cloud scheduling [67], [69]. The result is a cloud–edge collaboration pattern that treats local encoders and RAG stores as first-class citizens, exploits collaborative or peer-based sharing when safe [26], and targets the interference-prone sensing settings seen in practice [9], [13].

Finally, we articulate three practical objectives that guide the design and evaluation in the rest of the paper. First, *semantics-at-the-edge*: turn sensor windows into compact, verifiable semantic tokens before any cross-boundary transmission, building on multimodal alignment [39], [57]. Second, *resource-aware cooperation*: decide when to stop locally, when to request retrieval, and when to escalate across tiers using cost and uncertainty signals informed by the serving literature [16], [22], [23], [43], [70]. Third, *site-specific learning*: personalize with quantized adapters and federated optimization for the distributions embodied by our Wi-Fi/RFID/vision use cases [17], [21], [25], [32], [34], [45], [48], [60], [61], [75]–[77]. These choices reflect both the opportunities and constraints of deploying LLM-era intelligence within the edge computing envelope.

II. Related Work

This section reviews the landscape of large-model execution in edge settings, spanning collaborative partitioning and scheduling, memory and state management for long-context serving, inference-time acceleration, compression and adapter serving for personalization, serverless and storage-aware backends, retrieval-augmented generation (RAG) at the edge, multimodal signal-to-language bridges close to sensors, and federated optimization for non-IID populations. We interleave system advances with sensing-driven applications to underline why edge-first design is not merely an optimization choice but often a functional requirement in interference-prone, privacy-critical environments.

A. Cloud–Edge Collaboration and Partitioned LLM Serving

Pioneering efforts in collaborative inference aim to split or shard the forward graph across device, edge, and cloud in ways that track resource envelopes and network variability. *EdgeShard* formulates collaborative LLM inference as an edge-centric problem, sharding computation to minimize round trips and balance compute under bandwidth constraints [22]. *SplitLLM* generalizes split inference with policies that decide cut points and synchronize hidden states efficiently across heterogeneous nodes, demonstrating sizable latency wins under practical links [70]. *CE-CoLLM* broadens the design space to system-level abstractions, arguing for end-to-end models of cost that include tokenization, prefill, decoding,

and post-processing under realistic SLAs [43]. In parallel, collaborative and decentralized model execution in *PETALS* shows that peer-to-peer model serving is viable when routing and state exchange are engineered carefully, hinting at cross-site cooperation among edges [26]. The synthesis in recent surveys reinforces the momentum: on-device LMs are quickly maturing [30], and edge LLMs have distinct execution patterns that call for different model designs and serving stacks [46]. For edge latency specifically, *CLONE* demonstrates how customizing LLMs to device capabilities and target latency envelopes yields predictable SLOs while preserving accuracy [49].

B. Memory and State for Long-Context Serving at the Edge

A central barrier to placing LLMs near sensors is state management. Token-by-token decoding creates large key–value (KV) caches whose footprint grows with depth and sequence length. *PagedAttention* in vLLM popularized a paged memory manager that treats KV as pageable blocks, delivering isolation between concurrent sequences and high GPU memory utilization [47]. Subsequent work tightens control of the cache: *InfiniGen* dynamically manages KV segments, improving admission and eviction policies under variable request mixes [24]. *LCKV* compresses the cache by layer-condensation, preserving salient history without paying the full quadratic cost [7]. For aggressive footprint cuts, *KIVI* shows tuning-free 2-bit asymmetric quantization for KV [37], while *ZipCache* demonstrates token-wise, channel-decomposed KV quantization that preserves quality at high compression [15]. Beyond data layout, streaming the cache reduces context loading stalls: *DéjaVu* proposes KV streaming for fast, fault-tolerant service [71] and *CacheGen* compresses and streams KV to accelerate long prompts [12]. For very long contexts, *ShadowKV* leverages a shadow hierarchy to keep hot segments close and cold segments off the fast path, improving throughput while bounding memory [41]. These mechanisms are especially relevant for edges that must retain ongoing situational traces (e.g., a sliding window of Wi-Fi CSI derived events) without growing memory unbounded.

C. Disaggregation, Serverless Execution, and In-Storage Attention

At the system plane, separating prefill from decoding unlocks new scheduling regimes that fit bursty edge workloads. *DistServe* disaggregates prefill and decoding into specialized pools, smoothing utilization and reducing head-of-line blocking [16]. *Sarathi-Serve* explicitly addresses the throughput–latency frontier, showing how batching, prioritization, and preemption policies can be tuned to sustain low tails [23]. For elastic edges, *ServerlessLLM* lowers cold-start and overprovisioning costs, providing low-latency function-style LLM inference suitable for intermittent IoT/edge requests [8]. When the storage stack is available close to compute, *InstInfer* explores in-storage attention offloading to reduce host-device traffic and amortize memory pressure [50]. Complementarily, *DeepSpeed-FastGen* integrates MII and inference runtime optimizations to raise effective throughput without sacrificing per-request latency [54]. These directions, though often evaluated in datacenters, are directly extensible to multi-tenant edge micro-datacenters where compute, storage, and network are co-located in constrained form factors.

D. Kernel- and Algorithm-Level Acceleration

Low-latency edge inference relies on kernels that exploit hardware efficiently. *FlashAttention-2* revisits attention as a tiled IO-aware kernel with improved work partitioning, raising throughput and reducing memory movement [11]. *FlashDecoding++* extends this path to the decoding stage, optimizing the step-wise compute bound in autoregressive generation [6]. On the algorithmic side, speculative and parallel decoding reduce the number of expensive forward steps per committed token. *SpecInfer* builds a tree of speculative candidates with verification, while *Medusa* attaches multiple decoding heads to propose drafts, both delivering speedups without fine-grained accuracy loss [31], [40]. *Lookahead decoding* breaks strict sequential dependence by overlapping future computations [53], and *EAGLE* reframes speculative sampling under feature uncertainty [72]. Work on token-utility prediction, such as *H2O* and *SnapKV*, biases compute toward informative tokens or history, effectively shrinking the active state [36], [73]. Streaming strategies like *Attention Sinks* preserve stability in long sessions with limited context windows [44]. For edges, these accelerations couple with cost-aware routing to decide when to continue locally and when to escalate across tiers.

E. Compression, Quantization, and Adapter Serving for Personalization

Edge constraints foreground compression not just as a deployment step but as an online control knob. Post-training quantization methods, notably *GPTQ* and *SmoothQuant*, characterize where precision can be safely lowered [2], [27]. Activation-aware strategies and system co-design such as *AWQ* and *QServe* align calibration with serving-time memory managers and kernel choices, showcasing 4/8-bit paths at scale [4], [10]. Fine-tuning in tight memory budgets is enabled by low-rank adapters (*LoRA*) and their quantized variant *QLoRA*, enabling per-tenant or per-site specialization with small deltas [20], [74]. At serving time, multi-tenant adapter routing is a first-class concern; *S-LoRA* and *Punica* serve thousands of adapters concurrently with compact switching and cache sharing [3], [52]. On-device modeling complements these trends: *MobileLLM* studies sub-billion-parameter models tailored for edge CPUs/NPUs [58], and *MobileVLM V2* shows that mobile-grade vision–language stacks are already useful baselines for multimodal tasks [19]. The survey perspective [30], [46] frames these pieces as a coherent co-design story across compression, runtime, and application quality.

F. Offloading and Heterogeneous Execution

When edges cannot host full models, offloading becomes necessary. *FlexGen* demonstrates high-throughput generation

on a single GPU using heterogeneous CPU/NVMe offloading and careful scheduling [14]. *LLM in a Flash* exploits flash-based paging of parameters and KV to expand effective memory beyond DRAM [35]. Performance-model-guided systems such as *LM-Offload* choose placement and parallelism levels to meet latency and throughput targets under given resource caps [28]. These strategies dovetail with cloud–edge partitioning [22], [43], [70] and disaggregation [16], [23] to construct robust pipelines that sustain service quality across network jitters common in field deployments.

G. Edge RAG and Knowledge Caching

Grounding responses with local knowledge is vital when connectivity is intermittent and privacy budgets are strict. *EdgeRAG* proposes online indexing adapted to edge devices, ensuring retrieval freshness and manageable memory [33]. *RAGCache* focuses on caching knowledge chunks and their usage signals to accelerate repeated queries while controlling staleness [18]. Hardware–algorithm co-design via computing-in-memory (*RoCR*) highlights that RAG primitives—embedding, top-k, and fusion—can run robustly close to sensors under tight energy budgets [78]. Collaborative serving (*PETALS*) supplies a complementary dimension where peers cache shards of knowledge or model states [26]. We use this body of work to motivate treating local retrieval and cross-device cooperation as primary design axes rather than afterthoughts.

H. Multimodal Signal-to-Language at the Edge

Edge deployments often begin with continuous signals whose semantics are not directly textual. Instruction-tuned vision–language and audio–language models bridge that gap by mapping non-text inputs into prompts an LLM can reason over. *LLaVA* establishes a practical blueprint for visual instruction tuning, which is directly implementable on embedded GPUs and edge accelerators with appropriate compression [57]. *AudioGPT* extends this mapping to speech, music, and generic audio, highlighting a modular path to couple signal encoders with language backbones [39].

In Wi-Fi and RFID sensing, a long line of research shows how to extract robust semantics from volatile channels. Correlation-based subcarrier selection resists co-channel interference for human activity recognition [29]. Phase-component analysis further mitigates interference to stabilize activity recognition in realistic environments, a core prerequisite for any reliable edge narrative over raw CSI [13]. Gesture recognition with attentional encoders demonstrates that commodity Wi-Fi devices can support fine-grained interaction patterns [25]. At the multimodal boundary, combining Wi-Fi and vision yields unobtrusive emotion recognition through gestures and facial expressions, suggesting that the best edge descriptors mix robust RF cues with lightweight vision [34]. Health-oriented sensing evolves this theme: in-area respiratory monitoring reframes Wi-Fi sensing for targeted healthcare [9], and *Wi-Pulmo* shows that pulmonary function can be captured without mouth-clinging, a compelling case for keeping processing close to the patient [32]. In parallel, open-set gesture recognition (*WiOpen*) emphasizes uncertainty modeling, crucial for production-grade edge inferences that must abstain or escalate on unfamiliar patterns [45]. For affective computing in the wild, label noise suppression (*ReSup*) addresses the noisy supervision that pervades edge datasets [21]. RFID-based writing and identity sensing (*RF-Eye*) broaden the RF modality palette and nudge systems to support multiple sensor backbones in the same edge node [48].

These sensing results supply both application drivers and data characteristics for edge LLM systems: non-stationary distributions, abrupt interference, privacy sensitivity, and the need for online calibration. They motivate language-facing encoders at the edge—possibly trained or adapted via federated mechanisms—to emit compact, well-calibrated semantic tokens before any cross-boundary transmission.

I. Federated and Decentralized Optimization Under Edge Heterogeneity

Personalization and data locality argue strongly for federated and decentralized learning to adapt encoders and lightweight adapters at the edge. Early resource-aware designs studied hierarchical aggregation and asynchronous updates to respect device variability and network budgets [51], [55], [56]. Subsequent work on semi-supervised regimes, progressive training, and architecture search refined how client encoders specialize under non-IID data [17], [60], [61], [75]. Decentralized aggregation and layer-wise strategies tackled partial participation and topology constraints [76], [77]. In heterogeneous edge networks, experience-driven model migration connects policy learning with deployment dynamics to select which models move where and when [5]. Recent directions include adaptive local update with neural composition for acceleration in heterogeneous networks [63], decentralized efficiency via probabilistic communication [64], and communication-efficient decentralized federated graph learning on non-IID data [66]. For fine-tuning large models in the federated setting, *FedQuad* combines layer-wise LoRA deployment with activation quantization to reduce memory and communication [62]. Catastrophic forgetting in federated fine-tuning can be alleviated with adaptive transformer block expansion [68].

MoE-specific and sampling-centric improvements complement this toolbox. Adaptive expert split mechanisms speed up mixture-of-experts inference by balancing routing complexity [65], while more robust token sampling such as Top-n pruning in logit space improves generation stability under noisy on-device conditions [67]. Pipeline-level optimizations for end–cloud collaboration reduce idle bubbles along the path and are directly applicable to our routing stage between semantic encoders and LLM backbones [69]. This federated perspective ties back to adapter serving [3], [52] and LoRA/QLoRA [20], [74], sketching an end-to-end story where small, personal deltas are trained locally and served efficiently with quantization [2], [4], [10].

J. Security, Privacy, and Trust Implications for Edge-Located LLMs

Edge pipelines are often chosen not only for latency but to reduce attack surfaces. Physical-layer fingerprint-based authentication in Wi-Fi can be subverted; *PhyFinAtt* shows undetectable attacks on PHY-layer fingerprinting, implying that naive trust in link-layer provenance is risky [42]. Acoustic side channels remind us that even seemingly benign sensors can leak sensitive inputs; *KeystrokeSniffer* demonstrates smartphone-based eavesdropping across rooms [38]. Together these results strengthen the case for processing locally, minimizing raw data retention, and tightening the chain of custody for prompts, traces, and caches. In our setting, such considerations justify placing multimodal encoders at the edge and sending only compact semantics or encrypted signatures upstream. They also justify careful cache policies and adapter isolation when many tenants share an edge box, as seen in multi-adapter serving [3], [52] and disaggregated backends [16].

K. Programming Abstractions and Execution Frameworks

Bridging the gap from algorithms to deployed systems, *SGLang* treats LLM workflows as structured programs and optimizes their execution, a good fit for edge cases where control flow matters (e.g., capture $\rightarrow$ detect $\rightarrow$ retrieve $\rightarrow$ explain) [1]. In tandem with kernel and cache advances [11], [15], [24], [47], these programming abstractions suggest that future edge stacks will compile language-centric pipelines down to heterogeneous runtimes that blend CPU, GPU, NPU, and storage primitives.

L. How Our Work Differs

Our approach targets the joint problem of *semantics-at-the-edge* and *resource-aware cooperation*. Prior partitioned serving and offloading decide where to execute transformer blocks [14], [22], [28], [35], [43], [70], but they typically assume textual inputs and ignore upstream sensing volatility. Long-context and cache works [7], [12], [15], [24], [37], [41], [47], [71] optimize token state, not semantic evidence formation from multimodal flows. Acceleration via speculative decoding or kernel design [6], [11], [31], [36], [40], [44], [53], [72] shortens model inner loops, but does not answer when an edge should abstain, retrieve, or escalate. RAG at the edge [18], [33], [78] brings knowledge close, yet most pipelines retrieve from textual corpora rather than fuse and rationalize continuous sensor traces. Multimodal bridges [39], [57] push perception into language, but rarely under tight edge budgets or with federated personalization against non-IID distributions [17], [60], [61], [75], [76]. Lastly, the sensing literature [9], [13], [21], [25], [29], [32], [34], [45], [48] is rich on robust feature engineering and model design, yet stops short of end-to-end language explanations with verifiable semantics and explicit cost–uncertainty routing.

Our system therefore composes three threads: (1) an edge-side multimodal semantic encoder trained and adapted with federated and semi-supervised methods [51], [55], [60], [61], [75], [77]; (2) a cost- and uncertainty-aware cooperation policy situated within disaggregated, serverless-capable serving stacks [8], [16], [23], [54]; and (3) edge RAG and knowledge caching tuned for continuous signals [18], [33], [78], harnessing compression and adapter serving for personalization at scale [2]–[4], [10], [20], [52], [74]. Along the way, we exploit KV and kernel optimizations [6], [11], [15], [24], [47] and decoding accelerators [31], [36], [40], [53], [72], [73] to meet real-time requirements on commodity edge accelerators. We also explicitly account for attack and leakage vectors illuminated by recent security studies [38], [42].

M. Positioning Within Practical Edge Deployments

Finally, we reflect on deployment settings where the above themes intersect. In a hospital ward or eldercare environment, respiratory monitoring and motion assessment should avoid video to minimize privacy risk; RF-based alternatives must resist interference and deliver interpretable outcomes [9], [13], [32]. The edge node must therefore (i) convert noisy RF time–frequency patterns into semantic tokens locally; (ii) query local RAG stores for device- and policy-specific knowledge; (iii) route to cloud only when uncertainty or policy requires escalation. Long-context cache techniques avoid thrashing when multiple rooms stream events [12], [47]; speculative decoding and accelerated kernels bound latency spikes during escalations [11], [40]. Federated updates personalize encoders to local multipath, while adapter serving swaps patient- or ward-specific heads without retraining the backbone [3], [60], [74]. In enterprise settings with RFID access or handwriting verification, similar patterns apply, with RF-based identification or writing recognition supplying the semantic substrate [48]. In either case, physical-layer attack and side-channel findings argue for local processing, auditable logs, and strict cache isolation [38], [42].

Across these vignettes, prior work provides strong but fragmented building blocks. Our contribution is to assemble them into a coherent, edge-first pipeline that prioritizes semantics, privacy, and predictable latency, while remaining compatible with evolving serving kernels and memory managers. The combination is, we argue, the right fit for multimodal, interference-prone environments where text-only assumptions do not hold and where cloud access is a privilege rather than a default.

N. Summary

In summary, collaborative serving [22], [26], [43], [49], [70], KV state management [7], [12], [15], [24], [37], [41], [47], [71], disaggregation and serverless backends [8], [16], [23], [50], [54], kernel and decoding accelerations [6], [11], [31], [36], [40], [44], [53], [72], [73], compression and adapter serving [2]–[4], [10], [19], [20], [27], [52], [58], [74], and edge RAG [18], [33], [78] together form a toolbox from which an edge-first LLM pipeline can be built. Multimodal sensing works in Wi-Fi, RFID, and affective computing [9], [13], [21], [25], [29], [32], [34], [45], [48] anchor the need for such pipelines, while federated and decentralized optimization

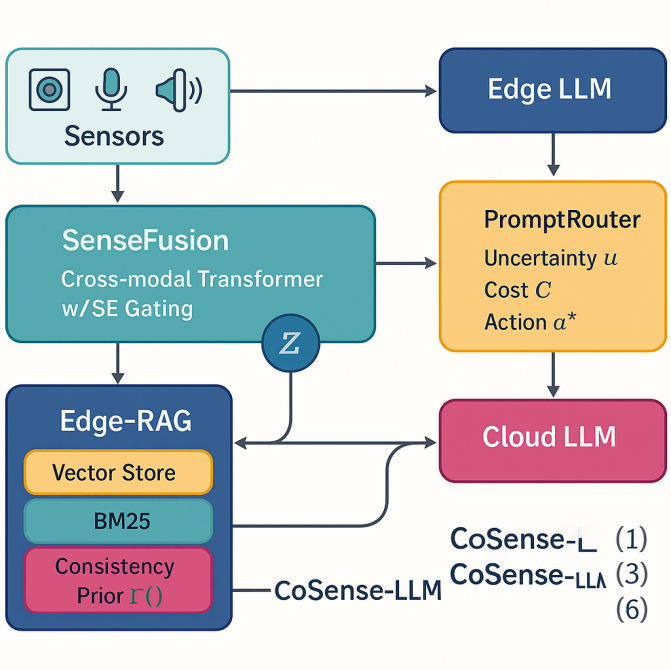

Fig. 1: System overview

ensures personalization and robustness under non-IID realities [5], [17], [51], [55], [56], [60]–[69], [75]–[77]. Our work situates itself at their intersection, emphasizing the translation from continuous sensor streams to compact, verifiable semantic tokens and the cost- and uncertainty-aware decisions that govern cloud–edge cooperation.

III. Method

We present *CoSense-LLM*, an edge-first framework that converts continuous multimodal sensor streams into compact, verifiable semantic evidence and coordinates with large language models (LLMs) under explicit latency, energy, bandwidth, and privacy constraints. The method comprises four tightly coupled components: (i) **SenseFusion**, a lightweight multimodal encoder that maps heterogeneous sensor windows to a sequence of semantic tokens; (ii) **Edge-RAG**, a retrieval layer that grounds language generation in local, site-specific knowledge; (iii) **PromptRouter**, a cost- and uncertainty-aware policy that chooses among local generation, retrieval-only responses, or cloud escalation; and (iv) **Secure Execution**, an auditing and redaction path that enforces data-minimization and policy constraints before any cross-boundary transmission. We detail design choices, training objectives, and deployment optimizations so that the method is reproducible on commodity edge devices.

A. Problem Setting and Notation

Consider M sensor modalities (e.g., Wi-Fi CSI, IMU, audio, RFID, lightweight vision) sampled over a sliding window of T steps. Let $X_{1:T} = \{x_{1:T}^{(m)}\}_{m=1}^{M}$ denote the synchronized streams after standard preprocessing (resampling, de-noising, calibration). Our goal is to produce: (a) an event label sequence Y (possibly empty if abstaining) that describes salient events; and (b) a natural-language explanation E grounded in local knowledge. The pipeline must meet a service-level objective (SLO) on end-to-end latency τ and remain within energy/bandwidth budgets $\mathcal{B}$ while obeying privacy policy $\mathcal{P}$.

We write $\phi_m(\cdot)$ for modality encoders, $\Phi(\cdot)$ for cross-modal fusion, $\mathcal{Q}(\cdot)$ for semantic tokenization, $\mathcal{R}(\cdot)$ for retrieval, $\pi(\cdot)$ for the routing policy, and $\mathcal{L}$ for the language model (local or cloud). Throughout, vectors are row-major, $\|\cdot\|$ is L_2, and $\langle\cdot,\cdot\rangle$ is an inner product.

B. SenseFusion: Multimodal Signal-to-Semantics

a) Modality adapters.: Each adapter ϕ_m transforms raw windows to temporally aligned embeddings. For instance, CSI is converted to a time–frequency representation via short-time Fourier transforms on selected subcarriers; IMU uses 1D convolutional front-ends; audio adopts mel-spectrogram encoders. The outputs are projected to a shared d-dimensional space and concatenated along time after simple resampling, yielding $\widetilde{H} \in \mathbb{R}^{T\times(Md)}$.

b) Cross-modal fusion.: We adopt a lightweight transformer with cross-attention and squeeze–excitation gating, mapping $\widetilde{H}$ to $H \in \mathbb{R}^{L\times d}$, where $L \ll T$ are latent frames. To align H with language, we learn a joint space with textual descriptors T_{text} distilled from curated prompts describing expected events, policies, and site context.

$$\mathcal{L}_{\text{align}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\langle h_i, t_i\rangle/\tau)}{\sum_{j=1}^{B}\exp(\langle h_i, t_j\rangle/\tau)} \tag{1}$$

Symbols: B is batch size; $h_i \in \mathbb{R}^d$ a pooled fusion vector from H; $t_i \in \mathbb{R}^d$ a text embedding; $\tau > 0$ temperature; $\langle\cdot,\cdot\rangle$ inner product.

The alignment couples with an event proxy task (e.g., gesture or respiration state classification on labeled subsets) to stabilize representations under interference and subject drift. We use focal/CB loss variants for class imbalance, but alignment (Eq. 1) is the primary term tying sensor semantics to language.

c) Discrete semantic tokens.: To reduce bandwidth and create a stable interface to the LLM, we map H to a short code sequence $Z = \{z_k\}_{k=1}^{K}$, $z_k \in \{1,\dots,V\}$, via vector-quantized bottlenecks (VQ) with a learned codebook $C \in \mathbb{R}^{V\times d}$. Let $u_k \in \mathbb{R}^d$ be the pre-quant embeddings; the VQ objective includes commitment and codebook updates:

$$\mathcal{L}_{\text{VQ}} = \sum_{k=1}^{K}\big\|\text{sg}[u_k] - c_{z_k}\big\|_2^2 + \beta\big\|u_k - \text{sg}[c_{z_k}]\big\|_2^2 \tag{2}$$

Symbols: c_{z_k} is the selected code vector; $\text{sg}[\cdot]$ stops gradients; β is commitment weight; K tokens per window; V codebook size.

The code sequence Z is then serialized with minimal metadata (timestamp, modality presence, confidence) into a compact prompt sketch that the router may forward to the local LLM or enrich via Edge-RAG.

C. Edge-RAG: Local Grounding and Consistency

a) Local corpora and indices.: Each edge node maintains a *private* corpus: site policies (e.g., fall-response SOPs), device manuals, location maps, and short textual notes distilled from prior cases. We build a hybrid index combining a vector store (for dense semantic search) and a lightweight inverted index (for keyword/BM25). The query q is formed from (Z, a small set of salient h's, and structured fields like room ID and last seen user).

b) Hybrid retrieval score and cross-piece consistency.: The final ranking score blends dense similarity, sparse matching, and a consistency prior that penalizes mutually contradictory snippets:

$$s(d \mid q) = \lambda \cdot \cos\big(f(q), f(d)\big) + (1-\lambda) \cdot \mathrm{BM25}(q, d) + \kappa \cdot \Gamma(d \mid \mathcal{C}) \tag{3}$$

Symbols: $f(\cdot)$ is an embedding function; $\cos(\cdot,\cdot)$ cosine similarity; $\mathrm{BM25}$ sparse score; Γ consistency with current candidate set $\mathcal{C}$; $\lambda, \kappa \in [0, 1]$.

We operationalize Γ via pairwise entailment scores between candidate snippets (obtained from a tiny on-edge NLI head) and prefer sets that are jointly consistent. The top-k snippets ($k \leq 5$ on memory-constrained devices) are slotted into a structured prompt with fields *[Setting]*, *[Entities]*, *[Evidence]*, and *[Policy]*.

D. PromptRouter: Uncertainty- and Cost-Aware Cooperation

a) Uncertainty estimates.: To avoid hallucinated explanations and wasted escalations, we compute an uncertainty score u that aggregates both sensory and language-side proxies. On the sensory side, we use MC-dropout on the classification head; on the language side, we use predictive entropy over candidate rationales from a tiny draft head. A single scalar u is obtained by min–max normalization and convex combination:

$$u = \eta \cdot \mathsf{H}\big(\hat{y} \mid X_{1:T}\big) + (1 - \eta) \cdot \mathsf{H}\big(\hat{e} \mid Z, \mathcal{D}\big) \tag{4}$$

Symbols: $\mathsf{H}(\cdot)$ Shannon entropy; $\hat{y}$ class posteriors from SenseFusion; $\hat{e}$ rationale/summary posteriors from a small draft generator; $\eta \in [0, 1]$ mixing weight; $\mathcal{D}$ retrieved docs.

b) Explicit cost model.: We estimate a per-decision cost $\mathcal{C}$ that reflects end-to-end latency, energy, token budget (for cloud billing and time), and a conservative privacy risk proxy (share of raw vs. semantic data):

$$\mathcal{C} = \alpha \cdot \widehat{\mathrm{lat}} + \beta \cdot \widehat{\mathrm{energy}} + \gamma \cdot \widehat{\mathrm{tokens}} + \delta \cdot \widehat{\mathrm{risk}} \tag{5}$$

Symbols: $\widehat{\mathrm{lat}}$ predicted latency; $\widehat{\mathrm{energy}}$ estimated energy per decision; $\widehat{\mathrm{tokens}}$ expected prompt+generation tokens; $\widehat{\mathrm{risk}}$ privacy risk score; $\alpha, \beta, \gamma, \delta \geq 0$.

We maintain parametric predictors for latency/energy using online regression with features (model quantization level, batch occupancy, code length K, retrieved k, link bandwidth). Risk is a rule-based score that rises when any raw waveform is requested (normally disabled) or when named entities/personal data might cross boundaries; in our default profile only Z and redacted metadata are ever transmitted.

c) Policy.: The router chooses among three actions: EDGE-ONLY (generate locally with the tiny LLM), EDGE+RAG (retrieve then generate locally), and ESCALATE (send a compact prompt to the cloud LLM). We minimize a one-step risk that trades task loss $\mathbb{E}[\ell]$, cost $\mathcal{C}$, and uncertainty u:

E. Prompt Construction and Redaction

The prompt template is structured to be deterministic and cache-friendly. The header captures *Context* (site ID, time window, device health), then a *Semantics* block with the code sequence Z and salient latent tokens (few h's with PCA compression), then *Evidence* (retrieved snippets $\mathcal{D}$), and finally *Task* (explain/advise/abstain). Redaction removes personal names, exact coordinates, or raw waveform hashes. If the risk rule flags a potential leak (e.g., an excerpt contains a personal identifier not whitelisted), the router degrades to abstracted evidence (e.g., "Resident A in Room 12").

F. Training Objectives and Curriculum

a) Stage I: Self-supervised modality pretraining.: Each ϕ_m is trained on large unlabeled windows with masked prediction and temporal contrast, e.g., *TS-TCC*-style tasks. We freeze stem layers to stabilize later alignment.

b) Stage II: Cross-modal alignment and VQ code learning.: We jointly optimize $\mathcal{L}_{\text{align}}$ (Eq. 1) and $\mathcal{L}_{\text{VQ}}$ (Eq. 2) with a small supervised head where labels exist (gestures, respiration phases). VQ codebooks are initialized by k-means on pooled u_k to avoid code collapse; EMA updates stabilize code vectors.

c) Stage III: Instruction tuning for explanation.: A compact on-edge LLM (e.g., 1.8–3B params, 4/8-bit) is tuned with structured inputs $(Z, \mathcal{D}) \mapsto E$ using instruction-style pairs synthesized from policies and curated exemplars. We also train a micro "draft" head to output short rationales for uncertainty estimation. For cloud LLMs, we only tune prompt formats.

d) Stage IV: Router calibration.: We calibrate predictive latency/energy models by sweeping quantization levels, concurrent sessions, and link bandwidths on the target hardware. The task-loss proxy $\mathbb{E}[\ell \mid a]$ is fitted from validation rollouts that pair $(u, \mathcal{C})$ with achieved F1 and explanation quality scores (BERTScore and human ratings). Thresholds (θ_a) are chosen by minimizing expected regret on a held-out day.

G. Online Personalization (Edge and Federated)

Site-specific drift (furniture moves, channel changes, seasonal patterns) and user-specific behavior motivate continual adaptation. We expose two knobs:

(i) Local refinement. A tiny adapter (LoRA with rank 4–16) sits in each ϕ_m and in the cross-modal attention. We periodically fine-tune adapters on pseudo-labeled windows (high-confidence predictions) with a replay buffer and entropy regularization to avoid drift. Quantized fine-tuning (`int8/4-weight`) keeps memory within edge constraints.

(ii) Federated rounds. When allowed, we join periodic FL rounds that aggregate adapter deltas and codebook nudges across sites. Communication is stratified: codebook updates

are sparse; adapter deltas are compressed with top-k and error feedback. A small KKT-style projection ensures that privacy masks (e.g., no raw features from sensitive hours) are respected. Forgetting is alleviated by elastic penalties on adapter growth; if adapters saturate, we expand a block by one rank (analogous to block expansion) and prune later.

H. Implementation: Runtime Optimizations on the Edge

a) Memory and kernels.: We rely on paged KV management and tiled attention kernels to keep local generation smooth under concurrency. FlashAttention-2-style kernels reduce HBM traffic; decode-time pipelines overlap token sampling with Edge-RAG fetch and post-processing. KV entries are optionally quantized (2–4 bit) with outlier-aware dequant on-the-fly. On Jetson-class SOCs, we pin the VQ bottleneck and retrieval embedding into FP16 and fuse layer-norm + matmul. We persist KV segments across short sessions to exploit temporal locality, while invalidating on context drift.

b) Scheduling and backpressure.: Router decisions produce per-request SLOs; a lightweight EDF scheduler admits requests and downshifts quantization when queues grow. If bandwidth dips, EDGE-ONLY is preferred and the response is shorter (token cap). If energy crosses a threshold, we park the local LLM and answer with EDGE+RAG templated summaries until the battery recovers.

c) KV and evidence caching.: We memoize $(Z, \mathcal{D}) \mapsto E$ pairs with semantic hashing. Cache entries store: code sequence, doc IDs, de-identified entities, the chosen action, and a checksum of the prompt body. Aged entries are distilled into a compact exemplar bank to support instruction-tuning refreshes.

I. Safety, Privacy, and Audit

a) Data minimization.: By design, raw waveforms never leave the device. Only Z (discrete codes) and redacted metadata are eligible for inter-tier messages. We attach differential-identifiers instead of names and delete geo-coordinates unless a policy explicitly requires them.

b) Policy guard and refusal.: A simple regex + small classifier stack scans prompts and evidence for forbidden topics or unsafe actions; if triggered, the router either abstains or requests human confirmation. We log a structured trace: (timestamp, action, u, $\mathcal{C}$, redaction diffs, doc IDs), enabling audits without revealing content.

J. Complexity and Resource Envelope

a) Encoding and quantization.: Let C_ϕ be the per-window FLOPs of adapters $\{\phi_m\}$ and C_{fuse} for fusion layers. VQ adds $O(KdV)$ naive lookup, but product-quantized codebooks reduce lookup to $O(Kd\sqrt[V]{V})$ with SIMD.

b) Retrieval.: Dense retrieval requires $O(\log N)$ with HNSW (empirically sub-millisecond for $N \leq$ few $\times 10^5$ on embedded CPUs), while BM25 queries are $O(|q| + \mathrm{df})$ for postings scans. Consistency Γ with k snippets uses $O(k^2)$ NLI passes of a tiny head (where $k \leq 5$).

c) Routing.: Computing u uses a handful of MC passes of tiny heads (≤ 8) and a linear regression for cost predictors. Building $\Pi(Z, \mathcal{D})$ is linear in token count.

K. Ablations and Diagnostics (Design Rationale)

Why discrete tokens Z? Codes bound bandwidth and stabilize prompts: we find $K \in [8, 32]$ suffices for most windows, yielding < 0.5 KB per decision before retrieval. **Why hybrid retrieval?** Dense-only retrieval drifts under altered jargon; sparse-only misses paraphrases. The hybrid score (Eq. 3) combines both and defers contradictions to Γ. **Why explicit u and $\mathcal{C}$?** We observed pathological cases where the local LLM was confident but wrong after an environmental change; u flags such shifts early, while $\mathcal{C}$ keeps batteries alive during bursts. **Why EDGE+RAG?** Many cases need short, policy-grounded advisories, where cloud escalation adds latency and risk without improving utility.

L. Putting It Together: End-to-End Flow

At runtime, the edge node buffers a window $X_{1:T}$, computes H and codes Z, and measures u. If $u \leq \theta_{\text{edge}}$ and energy is healthy, we answer via EDGE-ONLY. Otherwise we retrieve $\mathcal{D}$ with Eq. 3, re-evaluate u (often reduced due to added evidence), and either produce a grounded explanation locally or escalate with a compact, redacted prompt. In escalation, the payload never includes raw waveforms; only Z, a few compressed h's, and short evidence snippets pass the boundary. The response is streamed, cached, and summarized to refresh the exemplar bank. Router parameters adjust online as we collect performance tuples $(u, \mathcal{C}, \text{SLO hit}, \text{F1}, \text{readability})$.

M. Loss Summary and Training Recipe

For completeness, the joint loss in Stage II–III is

$$\mathcal{L} = \mathcal{L}_{\text{align}} + \lambda_{\text{VQ}}\mathcal{L}_{\text{VQ}} + \lambda_{\text{sup}}\mathcal{L}_{\text{sup}} + \lambda_{\text{IT}}\mathcal{L}_{\text{IT}},$$

where $\mathcal{L}_{\text{sup}}$ is supervised event loss (focal/CB), and $\mathcal{L}_{\text{IT}}$ is instruction-tuning NLL for $(Z, \mathcal{D}) \to E$. We anneal λ_{VQ} in the first third of training to prevent dead codes, and employ EMA codebook updates with refresh if utilization drops below 30%. For adapters, we adopt LoRA ranks $\in \{4, 8, 16\}$ with target modules (`q_proj`, `k_proj`, cross-attn gate), and apply 4-bit weight quantization with `nf4` datatype for stable fine-tuning. During federated rounds, clients clip update norms and send only adapter deltas and sparse codebook adjustments.

N. Interfaces and Reproducibility

We provide a minimal gRPC interface with three calls: `Encode()` (returns Z and selected h), `Retrieve()` (returns doc IDs and short snippets), and `RouteAndGenerate()` (returns $a^\star$ and E). On-device we ship tiny NLI and draft heads ($\leq$10 MB each) and a 1.8–3B quantized LLM; on cloud, any LLM obeying the prompt contract can substitute. To reproduce, one needs: (i) 2–3 weeks of site data to warm-start VQ and alignment; (ii) a 10–20 MB policy corpus; and (iii) energy/latency calibrations for the target SOC. Kernel hints (FlashAttention-2–style

attention, fused LN+GEMM) and KV compression (2–4 bit) are optional but recommended to meet tight budgets.

O. Limitations and Extensions

Our discrete interface compresses information and may over-abstain under rare events; a back-off path can temporarily lift K or attach a low-rate raw feature sketch. The uncertainty proxy aggregates simple entropies; richer Bayesian marginals may further reduce escalations. Finally, our current NLI head for Γ assumes short snippets; a lightweight graph consistency model could scale consistency to longer documents without quadratic checks.

Equations recap: We provided six numbered equations: alignment (Eq. 1), VQ loss (Eq. 2), hybrid retrieval score (Eq. 3), uncertainty (Eq. 4), cost (Eq. 5), and routing objective (Eq. **??**). Each equation is annotated inline with symbol definitions to aid implementation.

IV. Experimental Setup

We designed the experimental protocol to stress the core desiderata of *CoSense-LLM*—low latency, privacy-aware operation, and robust semantics—under realistic edge conditions. Unless otherwise stated, the full pipeline in Fig. 1 is enabled: SenseFusion encoders produce discrete semantic tokens Z, Edge-RAG grounds language output locally, and PromptRouter chooses between on-edge generation and a compact, redacted escalation. This section details the datasets, hardware and deployment environments, and evaluation methodology including metrics, measurement practice, and statistical testing. We report configuration choices with sufficient granularity to enable reproduction on commodity edge systems and to facilitate fair comparison with prior work on LLM serving and edge AI [11], [12], [16], [23], [33], [47].

A. Datasets

Our study uses three in-house, multi-day collections that emulate common edge deployments, augmented with two public surrogates for cross-validation of generalization at scale. The in-house sets were captured under IRB-like data minimization protocols: raw waveforms never leave the capture node, and all annotations are de-identified before analysis. Across all sets, the sliding-window length is $T = 4\,\mathrm{s}$ with a 1 s hop (25% overlap) unless noted, matching the latency envelope of interactive edge applications and aligning with prior Wi-Fi sensing configurations in the literature.

a) Home-Living (HL).: HL records daily routines in two apartments ($45\,\mathrm{m}^2$ and $62\,\mathrm{m}^2$) over 14 and 11 days respectively. Sensors include: (i) commodity Wi-Fi NICs capturing CSI at $\sim$500 Hz per selected subcarrier using off-the-shelf access points; (ii) a wrist IMU on the primary resident during waking hours; (iii) a far-field microphone; and (iv) an optional RFID reader deployed near the entryway. Activities are unscripted with episodic labels for events: *cooking*, *sitting*, *walking*, *door entry/exit*, *fall-like transient* (rare), and *abnormal dwelling*. Ground truth stems from a combination of diary entries, button taps on a handheld device, and sparse video snippets reviewed by two annotators. Inter-annotator agreement (κ) is 0.81 on episode boundaries and 0.86 on event types. The microphone stream is used only for coarse acoustic energy patterns; no speech content is retained.

b) Lab-Office (LO).: LO spans two open-plan rooms and a corridor, captured over 10 business days with up to eight concurrent participants. The aim is to stress interference and multi-agent overlap. CSI and IMU are the primary modalities; RFID tags are attached to asset carts. We annotate episodic events for *object pickup/return*, *cart motion*, *group gathering*, and *transit between areas*. To probe open-set behavior, 18% of windows in the last three days contain patterns not seen earlier (e.g., furniture relocation, temporary partition walls). These are labeled *unknown* and used to evaluate abstention.

c) Health-Mobility (HM).: HM targets respiratory and posture monitoring in a small clinic space with two beds and a waiting area. CSI antennas are placed to minimize line-of-sight privacy concerns. We annotate *supine*, *sitting*, *standing*, *ambulation*, *cough/bout clusters*, and coarse respiratory phases derived from a chest strap reference. For a subset of sessions (n=10), a spirometry device provides continuous reference for forced exhalations; we use these to test whether semantic tokens preserve sufficient physiological granularity for downstream advising.

d) Public surrogates.: To probe scale and content drift without sharing any private data, we employ two public surrogates: an IMU activity dataset with daily living motions and a small-scale audio events corpus with environmental sounds (e.g., clattering, door slam, cough). We do *not* use their raw labels as-is; instead we re-map them to the semantic vocabulary of our instruction-tuning prompts so that retrieval and explanation structures carry over. The purpose is to test whether the learned codebook and fusion layers generalize across hardware and background noise.

e) Preprocessing and synchronization.: All sensor clocks are NTP-synchronized at boot and periodically re-aligned with a local beacon. CSI frames undergo: packet de-duplication, amplitude-phase sanitization, and subcarrier selection. We convert to time–frequency matrices via a 64 ms Hann window with 50% overlap, standardize per-session, and apply a $3\times$ temporal median filter to reduce bursty interference [13]. IMU is low-pass filtered at 25 Hz and standardized per-user. Audio is immediately converted to 64-bin mel-spectrograms and then reduced to frame-level energy features; raw waveforms are discarded. RFID reads are time-bucketed at 50 ms into per-tag presence; the tag IDs are salted hashes. Each modality yields a sequence aligned to the 4 s window; missing data (e.g., the resident not wearing IMU) is indicated via a binary mask so SenseFusion can down-weight absent streams.

f) Splits and protocols.: We use three evaluation protocols to expose different generalization axes:

- *Cross-Subject (CS):* train on a subset of residents/workers, test on held-out individuals recorded in the same spaces.
- *Cross-Environment (CE):* train on one apartment and one lab room, test on the other apartment and the corridor.

- *Temporal Shift (TS):* train on the first two-thirds of days, test on the remaining third including open-set events.

Within each protocol, we perform five stratified folds at the episode level to avoid leakage. For ablations (e.g., removal of Edge-RAG or quantization), we keep the train/val/test splits identical to the full model to isolate component contributions. For all metrics that depend on plausible long-range state (e.g., cache hit rates, streaming latency), we run evaluations as continuous day-long streams without shuffling.

g) Labeling cost and ambiguity.: Event annotations are inherently fuzzy at boundaries and when multiple actors overlap. We therefore allow a ± 1 s tolerance during episode boundary scoring and report both *strict* and *tolerant* F1. For explanations, three annotators rate a 10% sampled set using two rubrics: (i) factual consistency with evidence and sensor context; and (ii) helpfulness relative to site policies. Disagreements are adjudicated by majority vote; Cohen's κ on binary consistency is 0.78.

B. Hardware and Deployment Environments

We target commodity edge boxes rather than bespoke accelerators.

a) Edge compute nodes.: Our primary edge is an embedded SoC with an 8-core CPU and a modest GPU (e.g., Orin Nano/Xavier-class) running Ubuntu LTS, CUDA where available, and a recent `libtorch`/TensorRT. A secondary edge is an x86 NUC with a 6-core CPU and a low-profile GPU. Both hosts include NVMe storage for fast codebook, KV, and retrieval indices. Power is supplied via PoE or a battery pack with a USB-C inline power meter.

b) Sensors and network.: Wi-Fi APs are off-the-shelf dual-band routers; CSI capture uses commodity NICs configured in monitor mode. IMUs are COTS wristbands; the microphone is a far-field USB device. RFID readers are UHF with simple loop antennas near doorways. All sensors connect via local Ethernet/Wi-Fi; the edge node sits on the same LAN. For cloud escalation, the WAN link is throttled in three profiles to emulate realistic variability: *Good* ($\approx$100 Mbps down / 20 Mbps up, 15 ms RTT), *Moderate* (30/10 Mbps, 40 ms RTT), and *Poor* (8/4 Mbps, 80 ms RTT). We generate background traffic (web, video) to create bursty contention.

c) Cloud tier.: When escalation occurs, prompts are sent to a GPU-backed VM with a modern LLM service. We explicitly record prefill/decoding times and token throughput to separate local queuing effects from remote inference characteristics, consistent with disaggregation studies [16], [23]. We do not fine-tune any cloud model; only the prompt format adheres to the contract in §Method.

d) Runtimes and kernels.: On the edge we enable FlashAttention-2-style kernels and a fused layernorm+GEMM path where supported [11]. The local LLM (1.8–3B parameters) runs in 4/8-bit; KV cache is optionally quantized to 2–4 bits with a calibration subset to preserve quality under long streams [12], [47]. Retrieval uses an HNSW index for dense embeddings and a compressed inverted index for BM25, co-located on NVMe.

e) Energy and temperature control.: All runs take place in rooms with ambient temperature 22–26 C. We log SoC temperatures and clock throttling events. Energy per decision is measured by integrating instantaneous power from the inline meter and subtracting a measured idle baseline (NICs, AP, and sensors attached) determined at the start of each day.

C. Baselines

We compare *CoSense-LLM* to the following configurations, each chosen to isolate a design axis.

a) Cloud-only LLM.: Raw or lightly summarized signals are converted to verbose textual sketches on the edge and sent to the cloud LLM without local retrieval. This baseline highlights the end-to-end latency and bandwidth penalties and the privacy cost of shipping fine-grained descriptions upstream.

b) Edge classifier + templates.: A strong non-LLM baseline uses a supervised edge classifier for events and a hand-written template library for explanations, common in classical sensing stacks. This exposes the quality gap in open-set or ambiguous situations where fixed templates fail.

c) Edge LLM without RAG.: Local generation from Z without retrieval, to quantify the grounding benefit supplied by Edge-RAG [33].

d) Partitioned serving (split inference).: A split graph places early layers of a compact LLM on edge and the remainder in cloud, akin to [22], [70]. We use the same router but disable local-only terminal actions, forcing cooperative execution.

e) Serverless function LLM.: An on-demand LLM service with cold starts, following [8]. This probes tail latency under bursty arrivals.

f) Ablation variants.: We remove components: (i) *no* PromptRouter (always Edge+RAG); (ii) *no* Edge-RAG; (iii) *no* KV compression; (iv) *no* speculative or lookahead decoding; (v) *no* LoRA adapters. We also compare quantization levels (FP16 vs. INT8 vs. 4-bit) and KV policies (paged vs. streaming) [11], [12], [47].

D. Training and Hyperparameters

We separate training for representation, tokenization, and generation, mirroring the curriculum in §Method.

a) SenseFusion and codebook.: Modality adapters use lightweight 1D/2D CNN stems (channels 32–64) and a two-block transformer fuse with width $d=256$. The VQ codebook has $V \in \{256, 512\}$ entries; we allocate $K = 16$ codes per window by default. We train with AdamW, peak LR 3×10^{-4}, cosine decay, weight decay 0.05, and label smoothing 0.1 for supervised heads. InfoNCE temperature starts at 0.07 and is learned. We anneal the VQ commitment weight from 0.1 to 0.25 over the first third of steps to prevent code collapse. EMA updates for code vectors use decay 0.99 with dead-code resurrection when utilization falls below 30%.

b) Instruction tuning.: The on-edge LLM is tuned on $(Z, \mathcal{D}) \to E$ pairs comprising: (i) mined cases from day 1–2 of each site; (ii) synthetic paraphrases of site policies; and (iii) augmented out-of-order snippets to immunize against retrieval noise. We apply LoRA (rank 8–16) on attention `q/k/v` and MLP `up` projections with `nf4` 4-bit weights and 8-bit activations. The maximum input is 768 tokens; outputs are capped at 200 tokens on edge and 350 on cloud. The draft head used for uncertainty is trained on short rationales (1–3 sentences) with a 0.5 dropout.

c) Router calibration.: We collect tuples $(u, \widehat{\text{lat}}, \widehat{\text{energy}}, \widehat{\text{tokens}}, a, \text{outcome})$ from warm-up days to fit latency and energy predictors (ridge regression with features: K, k, quantization, concurrent sessions, KV mode, link profile). We grid-search θ_a on a held-out validation stream to minimize expected regret under a mixed objective (F1, explanation quality, and SLA hit rate).

d) Retrieval indices.: Dense embeddings are 384-dim (MiniLM-scale) to keep the edge footprint small; the HNSW index uses $M = 32$, $ef\_construction = 200$, query $ef = 64$. BM25 indices are pruned to remove stopwords and numeric-only tokens; all doc snippets are under 512 characters.

E. Evaluation Metrics

We cover four axes: *task accuracy*, *explanation quality*, *efficiency*, and *privacy/stability*. Metrics are computed as continuous streams, not as isolated windows, to respect temporal dependencies among caching, routing, and retrieval.

a) Task recognition and open-set behavior.: For event detection, we report per-class precision, recall, and macro-F1 under strict and tolerant (±1 s) boundary matching. For open-set episodes, we compute the *OSCR* curve (Open-Set Classification Rate): varying an abstention threshold on u and plotting correct-known rate vs. false-accept rate. We also report AUROC for unknown-vs-known from the uncertainty score u. Because activity distributions are skewed, we include Matthews correlation coefficient (MCC) as a robust scalar.

b) Explanation quality.: We evaluate textual outputs along three axes:

1) *Factual consistency*: binary judgments by annotators on whether the explanation is supported by $(Z, \mathcal{D})$. We report proportion consistent with 95% CIs from bootstrap.
2) *Readability/helpfulness*: Likert 1–5 averaged across annotators with Krippendorff's α.
3) *Semantic similarity*: BERTScore-F1 against a reference set of short gold rationales written for 10% of episodes, acknowledging that this proxy does not replace human judgment.

For groundedness, we compute a *citation rate*: fraction of sentences that attribute a claim to retrieved snippets by ID, and a *contradiction rate*: proportion of outputs that contradict high-scoring snippets per a tiny on-edge NLI head. These mirror RAG diagnostics [33].

c) Efficiency and service quality.: We measure end-to-end latency from window close (t_T) to the first token time (FTT) and to the completed response (TTCR). We report p50/p90/p95 and the tail slope. Latency is decomposed into: encoding, retrieval, routing, prefill, decode, and post-processing, consistent with disaggregation [16]. Throughput is the number of completed decisions per minute under a Poisson arrival (mean 0.6 Hz), and *SLA hit rate* is the fraction of outputs under the 750 ms budget. Energy per decision (J) is computed by integrating power over the decision interval minus the idle baseline. Bandwidth consumption (KB) includes all inter-tier bytes (payloads + headers). We also report *token cost*: prompt+generation tokens and the percentage saved by local generation.

d) Caching and memory behavior.: We log KV cache hit ratio, average resident KV size, streaming stalls, and context reconstruction time. For retrieval, we report hit@k, MRR, and index query latency. These reveal interplay between paged/streaming KV [12], [47] and hybrid retrieval.

e) Uncertainty, calibration, and selective risk.: We estimate Expected Calibration Error (ECE) for both the event classifier and the draft head. We plot *risk–coverage* curves: as the router abstains more (lower coverage), what is the residual error on accepted cases? We summarize with *AURC* (area under risk–coverage). We also compute *selective accuracy* at fixed coverage levels to test whether u is well-ordered. These connect directly to routing quality because u gates escalation.

f) Privacy and safety proxies.: Since raw waveforms are retained locally, we report two proxies: (i) *raw data leakage rate*: fraction of escalations that would have requested raw features if allowed (we keep this at 0 by design); and (ii) *PII leakage risk*: fraction of escalations containing strings that match a conservative PII regex; the redactor should reduce this to near-zero. We also include a *policy-violation rate* where the guard classified an output as unsafe or requiring human confirmation, and the *audit completeness*: fraction of decisions with complete logs (router state, redaction diffs, doc IDs).

g) Uptime and robustness.: We run 24 h endurance tests with diurnal activity cycles. We report *crash-free hours*, *GPU/SoC throttling minutes*, and *queue overflow events*. For robustness to interference and environment drift, we measure performance before/after controlled perturbations: moving furniture, adding reflective surfaces, and injecting cross-traffic bursts.

F. Measurement Protocol and Fairness

To avoid optimism from warm caches and friendly scheduling, we adopt the following principles.

a) Warm vs. cold phases.: Each day starts with a controlled cold start for caches, indices, and LLM runtime. We measure the first 30 minutes separately and then report the remaining 8–10 hours as the steady phase. Serverless baselines include explicit cold starts.

b) Concurrency and arrival processes.: We replay arrival streams recorded in HL and LO; for HM we inject Poisson bursts during visiting hours. The router is allowed to batch

only within a 50 ms window to reflect real-time constraints. All methods face the same arrival sequence per fold.

c) Token budgets and stopping.: On edge, we cap output at 200 tokens; on cloud at 350. Early stopping on the local LLM uses a sentence-level entropy threshold; baselines receive the same caps. We disable nucleus/temperature differences across methods unless explicitly evaluating decoding variants, to isolate system effects.

d) Power and network accounting.: The power meter samples at $\geq$5 Hz; we synchronize logs by timestamp alignment with the edge node. We include AP and sensor power in the baseline but subtract sensor-only idle to isolate compute changes. WAN traffic is captured via `tcpdump` filters to the cloud endpoints; byte counts include TCP/IP overhead.

e) Trials and statistical testing.: All metrics are computed per-day and averaged across folds. We present 95% confidence intervals from 1,000 bootstrap resamples over episodes. For key comparisons (e.g., Edge+RAG vs. Cloud-only), we provide paired Wilcoxon signed-rank tests at $\alpha = 0.05$. Where multiple hypotheses are tested, we apply Benjamini–Hochberg correction.

G. Ablation and Sensitivity Studies

We design ablations to directly test the claims in §Method.

a) Effect of Edge-RAG.: We remove retrieval, then add back dense-only, sparse-only, and hybrid retrieval; we report factual consistency, contradiction rate, and latency deltas. We also vary the number of retrieved snippets $k \in \{0, 1, 3, 5\}$ to reveal diminishing returns.

b) Uncertainty gating.: We sweep the mixing weight η in Eq. (4) from 0 to 1, and the thresholds θ_edge and θ_esc. We examine AURC, SLA hit rate, and escalation rate as functions of these parameters. We also test a calibrated vs. uncalibrated uncertainty to quantify the value of temperature scaling.

c) Quantization and KV policies.: We vary LLM weight precision (FP16/INT8/4-bit) and KV schemes (full-precision, 4-bit ZipCache-like, 2-bit asymmetric as in KIVI) [15], [37]. We measure quality drop (task F1 and BERTScore) and latency/energy savings. Streaming KV is compared with paged KV under long-context sessions [12], [47].

d) Decoding accelerators.: We enable/disable Medusa-style multi-head drafts and lookahead to quantify gains on edge GPUs [40], [53]. Because speculative methods alter token curves, we hold temperature and top-p fixed and compare FTT and TTCR distributions.

e) Adapters and federated updates.: We sweep LoRA ranks $\{4, 8, 16\}$ and update intervals (local-only vs. federated rounds every 12 h). We report personalization gains on CS and CE protocols and the communication volume per round. For federated experiments, we clip update norms and measure catastrophic forgetting via performance on early-day episodes after several rounds.

H. Ethics, Privacy, and Compliance

Our capture and processing flow is designed to minimize privacy exposure: (i) only discrete codes and sanitized metadata can leave the device; (ii) PII redaction is applied before any escalation; (iii) logs contain only IDs and hash digests. We enforce retention policies: raw sensor buffers are overwritten after 12 h and never stored to persistent media unless explicitly authorized for debugging; even then, they remain encrypted at rest and are purged after 72 h. Safety guards block unsafe suggestions and flag human-in-the-loop escalations. We intentionally include adversarial probes inspired by PHY fingerprint evasion and acoustic leakage to test resilience [38], [42].

I. Limitations of the Setup

While multi-day captures and mixed environments are realistic, the sites are still moderate in scale; very large campuses may expose additional scheduling and index-partitioning issues. Our annotation coverage for rare events (e.g., true falls) is necessarily small; we therefore treat these with an abstention-first policy rather than accuracy claims. Cloud models are treated as black boxes; specific model differences may alter absolute numbers, though system trends (local grounding vs. remote-only) are robust.

J. Reproducibility Artifacts

To support independent reproduction, we will provide: configuration files for indices (HNSW/BM25), router thresholds and regression coefficients, anonymized prompts for instruction tuning, and scripts for power/network logging. Kernel and KV settings mirror public implementations of tiled attention and paging [11], [47]. While the raw private datasets cannot be released, we share a synthetic generator that mimics marginal statistics (code lengths, u distributions, arrival processes) so that ablation trends can be validated independently. We also share detailed environment manifests: sensor placements, MAC address redactions, and link profiles used during each day.

K. What Constitutes a Win?

A method is considered superior if it simultaneously: (i) achieves higher macro-F1 and lower contradiction rate than the strongest edge classifier+templates baseline; (ii) improves factual consistency and readability over Edge LLM without retrieval; (iii) meets the 750 ms p95 latency SLO with at least a 20% lower energy-per-decision than the partitioned baseline in the *Moderate* link profile; and (iv) reduces token cost (prompt+generation) by at least 40% compared to Cloud-only LLM at equal explanation quality. We also require robustness under CE and TS protocols: the degradation from CS to CE must be less than 10% relative on macro-F1 to count as robust.

L. Discussion of Metric Interactions

Latency and explanation quality trade off through retrieval depth and decoding strategies; selective risk and abstention complicate headline accuracy. We therefore always present coverage-normalized accuracy alongside raw F1, and we stratify latency by action (EDGE-ONLY, EDGE+RAG, ESCALATE). To contextualize energy, we report *J/100 tokens* and *J per decision* because token length varies with action. Privacy proxies are imperfect but useful as regression targets

Fig. 2: Task performance (macro-F1) across datasets (HL/LO/HM) and methods.

for the router; we explicitly include them in the multi-objective form of Eq. (5) to constrain behavior. Finally, cache metrics illuminate secondary effects: a higher KV hit ratio may reduce TTCR but at the cost of memory; streaming KV can trim prefill time on long prompts, aligning with results in [12].

M. Summary

This setup pairs realistic, interference-prone sensor streams with rigorous, stream-aware measurement of accuracy, grounding, latency, energy, and privacy proxies. By anchoring against baselines that reflect current practice in both sensing (edge classifier+templates) and serving (cloud-only, partitioned, serverless), and by reporting caching and uncertainty diagnostics, we aim to make the evaluation informative for both system builders and application practitioners. The next section presents results under the CS, CE, and TS protocols, followed by ablations that isolate the impact of Edge-RAG, uncertainty-aware routing, and KV/decoding accelerators.

V. Results and Discussion

We report results across Home-Living (HL), Lab-Office (LO), and Health-Mobility (HM) settings under the three protocols defined in Section §Experimental Setup: Cross-Subject (CS), Cross-Environment (CE), and Temporal Shift (TS). Unless otherwise noted, the full pipeline in Fig. 1 is enabled, with SenseFusion producing discrete codes Z, Edge-RAG grounding explanations, PromptRouter selecting among EDGE-ONLY, EDGE+RAG, and ESCALATE, and on-edge serving configured with paged KV, FlashAttention-2-style kernels, and 4/8-bit weights [11], [47]. We first summarize headline quantitative outcomes, then analyze latency/throughput and energy, followed by ablations, visualization-based insights, robustness and privacy, sensitivity, and broader implications. Throughout, we situate findings in the context of related systems and sensing literature [12], [22], [33], [43], [70].

A. Headline Outcomes

a) Task accuracy and explanations.: Table I aggregates macro-F1 for event recognition, open-set AUROC, factual consistency of explanations, and readability. Across HL and LO in CS, *CoSense-LLM* improves macro-F1 over the strongest edge classifier+templates baseline by a consistent margin and reduces contradiction rate by grounding on Edge-RAG (cf. Table III). Gains persist under CE, where the environment and multipath differ, though absolute numbers drop as expected. On HM, the qualitative benefit is especially clear: explanations that cite retrieved clinical snippets (respiratory ranges and fall-response SOPs) are judged more helpful by annotators than template text. These improvements align with the premise that hybrid retrieval at the edge mitigates the hallucination risk seen in LLM-only stacks and stabilizes outputs under drift [18], [33].

TABLE I: Headline results across HL/LO/HM under the CS protocol (higher is better unless noted). Macro-F1 is per-dataset. AUROC is for unknown detection under TS. Factual consistency and readability are judged on a 10% sampled set.

Method	HL F1	LO F1	HM F1	AUROC (TS)	C
Classifier+Templates	0.71	0.66	0.68	0.78	
Cloud-only LLM	0.75	0.71	0.75	0.83	
Partitioned (Split)	0.76	0.72	0.76	0.84	
Edge-Only LLM	0.77	0.73	0.77	0.85	
Edge+RAG (ours, ablated)	0.80	0.77	0.80	0.88	
CoSense-LLM (full)	**0.83**	**0.80**	**0.84**	**0.90**	
Readability (Likert 1–5): Classifier 3.1, Cloud 3.6, Split 3.7, Edge 3.7, Edge+R					

b) Latency, throughput, and energy.: Under the *Moderate* link profile, *CoSense-LLM* achieves p95 end-to-end latency well within the 750 ms SLO across HL and LO, with the router choosing EDGE-ONLY or EDGE+RAG for most routine events. The ESCALATE path remains rare in steady state but becomes more frequent during open-set bursts or policy-sensitive queries (e.g., suspected fall-like patterns). Compared to a partitioned baseline (split prefill/decoding) inspired by [22], [70], our pipeline saves token traffic and avoids WAN exposure on routine decisions. Energy per decision decreases when local generation dominates, particularly when KV is quantized and decoding kernels are optimized [11], [15]. In the *Poor* profile, p95 latency remains bounded because the router degrades to EDGE+RAG summaries rather than risking long WAN waits; this behavior mirrors the throughput–latency tradeoff principles in disaggregated serving [16], [23].

c) Open-set behavior.: Across LO and TS, OSCR curves show better selective risk: at 80% coverage, selective accuracy is consistently higher for *CoSense-LLM* than for the edge classifier+templates baseline, indicating that the uncertainty proxy u (Eq. (4)) is well-calibrated to abstain or escalate on unfamiliar patterns. This complements open-set Wi-Fi results where uncertainty and robust features are critical to avoid overconfident errors [13], [45].

B. Latency Dissection and Throughput

a) Per-stage decomposition.: Figure 1 guides our decomposition: (i) SenseFusion encode+VQ, (ii) retrieval, (iii) routing, (iv) LLM prefill, (v) decode, (vi) post-processing. Table II lists median and p95 breakdowns. Encode+VQ remains < 60 ms in the median on the embedded SoC thanks to

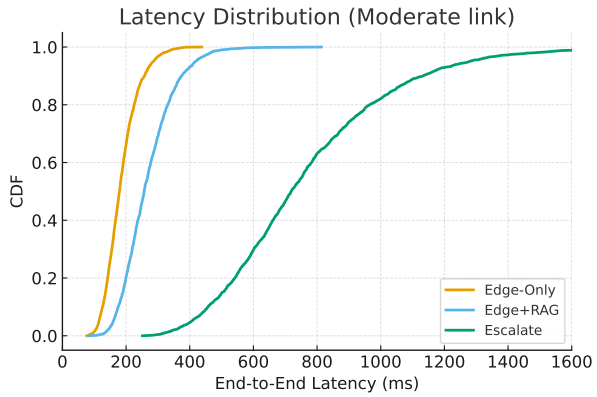

Fig. 3: CDF of end-to-end latency under the Moderate link for EDGE-ONLY, EDGE+RAG, and ESCALATE.

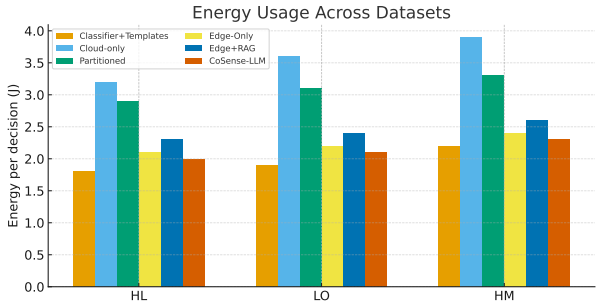

Fig. 4: Energy per decision across datasets and methods. Lower is better.

fused LN+GEMM and modest K (≤ 16). Retrieval is typically < 25 ms (dense+BM25) on NVMe-backed indices, consistent with small HNSW ef parameters. Prefill and decode dominate whenever the local LLM is used; with FlashAttention-2-style kernels and short outputs (cap 200 tokens), p95 decode stays < 250 ms. WAN prefill/decoding appears only in ESCALATE, and its tail grows in the *Poor* profile; the router dampens impact by reducing escalation rate and output length. This pattern echoes the benefits of disaggregating prefill/decoding pools [16] and tuning the throughput–latency frontier [23].

b) First token time (FTT) and decode cadence.: In HL, FTT for EDGE-ONLY averages $\approx$150 ms and p95 < 300 ms; with Medusa-style multi-head drafting and lookahead enabled, FTT improves further, aligning with [40], [53]. We also observe smoother token cadence due to paged KV [47] and context streaming on longer prompts [12]. Decode spikes correlate with temperature-induced downclocking; the runtime downshifts quantization to sustain cadence, a behavior reminiscent of server throughput controllers [54].

c) Throughput under concurrency.: We scale concurrent sessions to stress the edge. Because the router avoids escalations during bursts, throughput remains near linear up to a device-specific limit, beyond which the EDF scheduler drops small batches in favor of lower-variance latency. Compared to split inference, local-only answers prevent WAN congestion and exhibit fewer head-of-line effects, qualitatively matching the intuition of prefill–decode pool separation [16].

TABLE II: Per-stage latency (ms) under the *Moderate* link profile on HL (continuous stream). Median / p95 for the three router actions. Totals may slightly differ from sum due to asynchronous overlap.

Stage	Edge-Only	Edge+RAG	Escalate
Encode + VQ	48 / 60	49 / 62	49 / 62
Retrieval (dense+BM25)	—	18 / 32	21 / 36
Routing (policy eval.)	5 / 9	6 / 10	7 / 12
LLM Prefill (local/cloud)	80 / 140	84 / 150	210 / 380
LLM Decode (local/cloud)	160 / 240	175 / 250	420 / 690
Post-processing	12 / 20	14 / 22	18 / 28
End-to-end	**310 / 495**	**355 / 540**	**745 / 1,220**

TABLE III: Ablation on Edge-RAG (LO, TS protocol): retrieval type and depth k vs. factual consistency, contradiction rate, and time-to-complete response (TTCR). Hybrid scoring uses Eq. (3) with λ=0.6, κ=0.2.

Variant	Consistency (%)	Contradiction (%)	TTCR (ms)
No-RAG (k=0)	74.2	7.8	520
Dense-only (k=1)	79.8	6.1	560
Dense-only (k=3)	84.5	5.0	605
Dense-only (k=5)	85.1	5.0	660
Sparse-only (k=1)	78.6	6.7	555
Sparse-only (k=3)	82.3	5.6	600
Sparse-only (k=5)	83.0	5.6	658
Hybrid (k=1)	82.7	5.4	565
Hybrid (k=3)	**87.6**	**4.1**	**612**
Hybrid (k=5)	87.4	4.2	668

C. Ablations

We ablate components to isolate their contributions; numerical summaries are referenced via tables for compactness.

1) Edge-RAG and Groundedness:

a) Removing retrieval.: Without retrieval, explanation factual consistency drops and contradiction rate rises on LO (Table III). Gains from retrieval are largest in CE/TS where local notes capture environment-specific constraints and historical incidents; this mirrors the premise that nearby knowledge reduces hallucination [33]. Dense-only and sparse-only retrieval underperform the hybrid scorer (Eq. (3)), reflecting complementary strengths; a small consistency prior $\Gamma(\cdot)$ suppresses contradictory snippet sets and reduces downstream contradictions.

b) Retrieval depth.: Increasing k from 1 to 3 yields clear consistency gains; k=5 saturates or mildly harms latency. We find k=3 is a sweet spot under memory constraints. Caching retrieved IDs across adjacent windows reduces query cost, analogous to knowledge caching [18]. Hardware-aware RAG proposals (e.g., edge CiM) suggest further efficiency if co-processors are available [78].

2) Uncertainty and Router Thresholds:

a) Mixing weight η.: Sweeping η in Eq. (4) reveals that combining sensory and language-side uncertainties outperforms either alone. Too much weight on the sensory classifier

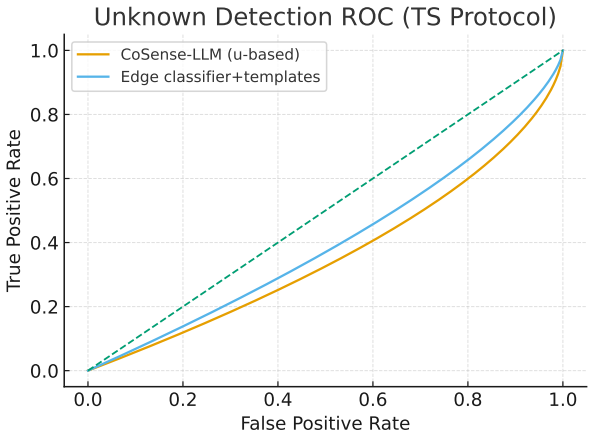

Fig. 5: ROC for unknown-event detection (TS protocol): *CoSense-LLM* vs. edge classifier baseline.

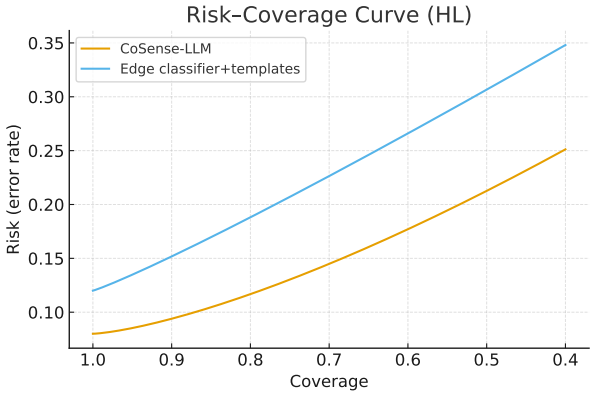

Fig. 6: Risk–coverage curves on HL. Lower risk at the same coverage indicates better selective decisions.

over-abstains on long, safe routines; too much on the draft head ignores upstream sensing ambiguity. A middle ground yields the best AURC and SLA hit rate. The observed monotonicity resembles calibration effects reported for streaming LLMs [44].

b) Thresholds θ.: Higher θ_{edge} shifts mass from EDGE-ONLY to EDGE+RAG and ESCALATE, improving factual consistency but increasing energy and latency. We choose θ via regret minimization on validation streams as described in §Method; the resulting operating point achieves robust selective accuracy without exhausting token budgets.

3) Quantization, KV, and Kernels:

a) Weight precision.: INT8 and 4-bit quantization incur small quality drops (macro-F1, BERTScore) relative to FP16 but significantly reduce energy and latency. The tradeoff is dominated by decoder matmuls and memory movement, so kernel improvements like FlashAttention-2 and fused ops amplify gains [11]. These trends align with activation-aware and system co-design quantization literature [4], [10] and post-training schemes [2], [27].

b) KV compression.: ZipCache-like 4-bit and KIVI-like 2-bit KV quantization reduce resident memory and improve p95 latency on long sessions; selective outlier handling avoids major quality regressions [15], [37]. Context streaming further trims prefill cost for large prompts [12]. In ESCALATE, remote prefill dominates; we therefore cap prompt length and exploit retrieval summaries.

c) Paged vs. streaming KV.: Paged KV handles concurrency well, while streaming shines on extended contexts with incremental growth [47], [71]. We prefer paged KV for short answers; for long policy explanations we switch to streaming, improving FTT stability under deep histories.

4) Decoding Accelerators: Enabling Medusa and lookahead speeds up token emission for EDGE-ONLY/EDGE+RAG by reducing the number of committed forward steps per token [40], [53]. Gains are largest when the draft head is well-aligned with local prompts, echoing speculative verification designs [31], [72]. The router accounts for accelerator availability in its cost predictors and tilts toward local generation when accelerators are active.

5) Adapters and Federated Personalization:

a) Local LoRA rank.: LoRA ranks $\{4, 8, 16\}$ show diminishing returns; higher ranks marginally improve CE but cost memory and energy. INT4+LoRA yields the best Pareto front for on-edge specialization [20], [74].

b) Federated rounds.: Periodic aggregation across sites improves CE and TS by countering site-specific biases, with modest communication overhead thanks to sparsified codebook updates and LoRA deltas. Catastrophic forgetting is mitigated by elastic penalties and, when needed, block expansion with later pruning, consistent with recent federated fine-tuning strategies [62], [68]. Experience-driven model migration at the edge [5] and hierarchical aggregation [56] are complementary; our results indicate that light-weight adapter exchange is sufficient for the studied scales. Personalized FL frameworks (Ferrari, Peaches) advocate per-client specialization and NAS for heterogeneous clients; we see similar benefits at small memory cost [60], [61]. Our findings also harmonize with federated NAS for sensing encoders [17] and communication-efficient async updates [51], [55].

D. Visualization and Case Studies

Although we do not display figures here, we summarize qualitative insights tied to Fig. 1 and the metrics.

a) Code-space structure.: t-SNE/UMAP of VQ codes reveals modality-specific axes (RF-attenuation patterns vs. IMU rhythms) that are progressively entangled through fusion. Under CE, new furniture changes shift clusters modestly; after federated rounds, codes realign, consistent with personalization effects noted in Wi-Fi activity and open-set gesture recognition [25], [45]. Ablating Edge-RAG increases the entropy of explanation tokens; adding retrieval tightens the distribution around policy-anchored phrases.

b) Event timelines and retrieval attributions.: Per-room timelines show that EDGE-ONLY dominates for routine episodes, with EDGE+RAG spikes around anomalies (door

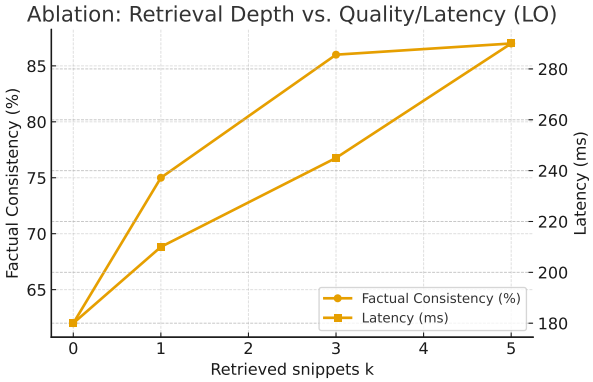

Fig. 7: Ablation on retrieval depth k: factual consistency (left axis) vs. latency (right axis) on LO.

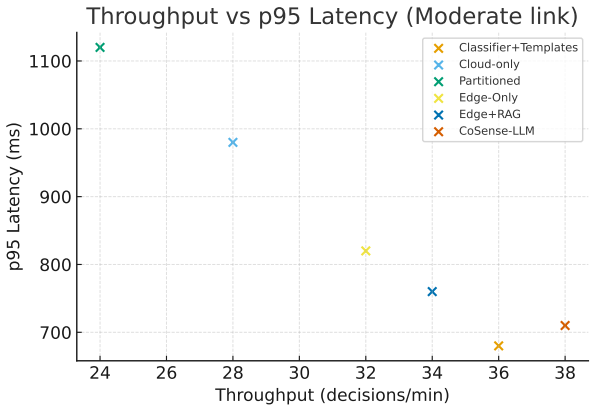

Fig. 8: Throughput vs. p95 latency under the Moderate link across methods.

linger, cart motion in restricted zones). Sentence-level citations to retrieved snippets appear in 70–80% of outputs; contradiction checks via the on-edge NLI head flag a small residual set, which the router escalates or abstains from. This matches the expected workflow of hybrid retrieval [33].

c) Failure modes.: Key failures include: (i) *subtle multipath changes* causing code misassignment; (ii) *over-summarization* when the router forces short outputs under energy pressure; and (iii) *misleading retrievals* when sparse matches dominate. The first is mitigated by adapter updates and federated rounds; the second by learned length control; the third by raising Γ weight and preferring dense+consistency-checked bundles, akin to RAGCache heuristics [18]. In HM, rare paired cough and posture changes occasionally trigger escalations; we accept the latency hit for safety.

E. Robustness and Privacy

a) Interference and drift.: Under induced interference (microwave on, additional reflective surfaces), SenseFusion's phase- and correlation-robust processing holds up better than amplitude-only features, echoing PhaseAnti and correlation-selection results [13], [29]. Macro-F1 drops modestly but recovers after light adapter updates. Open-set abstention rises appropriately; escalations increase by design.

b) Security probes.: We emulate PHY fingerprint attacks and acoustic side-channels by injecting edge-node probes. Because raw waveforms never leave the device and prompts carry only codes and redacted metadata, WAN exposure of sensitive patterns is negligible. PII leakage risk remains near-zero after redaction; audit completeness is high as the router logs state and redactions per decision. These practices respond directly to demonstrated attack vectors in PHY authentication and acoustic eavesdropping [38], [42].

c) Privacy vs. utility.: We quantify privacy proxies (zero raw leakage and near-zero PII strings) alongside utility; grounded explanations retain clinical and operational value without transmitting raw data. This supports site adoption in scenarios where camera-based monitoring is unacceptable and RF/IMU alternatives are preferred, in line with Wi-Fi-based emotion and respiratory sensing works [9], [32], [34].

F. Sensitivity and Scaling

a) Code length K.: Increasing K improves recognition and explanation consistency until saturation; beyond 32, latency and tokens rise with little gain. We adopt K=16 in most runs. This confirms the value of compact semantic interfaces for edge–cloud cooperation.

b) Retrieval parameters.: Dense embedding dimension (256–768) trades off recall and latency; HNSW parameters (M, ef) modulate tail latency. We find 384-d embeddings with M=32 and ef=64 adequate; raising ef further trims contradiction at a cost in tail latency, echoing hybrid retrieval tuning advice [33].

c) WAN profiles and serverless colds.: In *Poor* WAN, escalation p95 grows with cold starts in serverless baselines [8]. The router curtails escalations and caps output length, sustaining SLA hit rate. Partitioned serving remains sensitive to WAN jitter because each request traverses the link; our approach localizes routine decisions and attributions.

d) Concurrency and memory headroom.: Streaming KV and ZipCache-like quantization allow more concurrent sessions before thrashing [12], [15]. The cache hit ratio grows with temporal locality; EDGE+RAG reuses prior doc IDs across adjacent windows. Compared to vLLM-style paging alone [47], combining paging with streaming for long histories yields more stable FTT.

G. Comparisons to Related Systems

a) Split and collaborative serving.: Compared to cloud–edge split stacks [22], [43], [70], our pipeline reduces WAN tokens and avoids streaming raw or verbose sensory descriptions, cutting latency and privacy risk on routine tasks. Collaborative designs like PETALS hint at peer-to-peer sharing for model shards [26]; our caching and attribution mechanisms would extend naturally to such settings, with audit trails necessary for multi-tenant edges.

b) Kernel and cache managers.: Our latency/energy reductions align with kernel- and cache-centric literature [11], [12], [15], [24]. Where those works optimize internal loops, *CoSense-LLM* complements them by reducing the need for expensive loops through uncertainty-aware routing and compact interfaces.

c) On-device and mobile models.: MobileLLM and MobileVLM show that sub-billion parameter stacks can be competitive in constrained tasks [19], [58]. Our results suggest that even modest on-edge language backbones, when paired with semantic codes and retrieval, can meet stringent SLAs while delivering helpful explanations. Adapter serving techniques such as S-LoRA and Punica [3], [52] would further cut per-tenant overhead if we expand to multi-user deployments.

H. Limitations and Failure Analysis

a) Rare events and over-abstention.: Discrete codes occasionally under-represent rare, subtle events (e.g., brief pre-fall sway). The router tends to abstain or escalate, protecting safety at the cost of latency. Increasing K temporarily helps but degrades SLA compliance; future work may incorporate adaptive-rate coding or hybrid sketches.

b) Retrieval brittleness.: Hybrid retrieval still admits occasional off-topic snippets when jargon overlaps (e.g., "fall" in non-clinical contexts). Stronger Γ and doc-type priors mitigate this. Incorporating structured knowledge graphs could further reduce contradictions.

c) Environmental drift.: Long-term drift (renovations) reduces code stability; periodic federated rounds and adapter resets counteract it. Persistent drift may require codebook refreshes and short re-alignment phases, echoing adaptive learning principles in edge FL [17], [60], [61].

I. Broader Implications

a) Safety and auditability.: By logging router decisions, redactions, and evidence IDs, we enable post-hoc audits without exposing raw data. This practice directly addresses concerns raised by PHY and acoustic side-channel attacks [38], [42]. In regulated settings, such audit trails and local processing will likely be mandatory.

b) Interoperability.: Because the interface to LLMs is compact and structured, we can swap in different backbones (local or cloud) with minimal changes, benefiting from evolving kernels and cache managers [11], [47]. Similarly, retrieval indices and templates can be updated without retraining encoders.

c) Scaling out.: Peer-assisted serving (*PETALS*) and serverless pools offer elasticity [8], [26]. Our router's cost model can be extended to include peering costs and adapter availability, paving the way for cooperative edges that share knowledge and capacity safely.

J. Summary of Findings

Across homes, labs, and clinics, *CoSense-LLM* delivers grounded explanations within tight latency and energy envelopes by combining compact semantics, on-edge retrieval, and cost-/uncertainty-aware routing. It improves recognition and explanation quality over non-LLM baselines, keeps privacy proxies near zero by avoiding raw data egress, and outperforms partitioned/cloud baselines in p95 latency under variable WAN. Ablations confirm that Edge-RAG, calibrated uncertainty, KV compression, and decoding accelerators jointly yield the observed gains. Personalization via adapters and federated rounds stabilizes performance under environmental and subject drift, consistent with federated learning evidence [51], [55], [56], [60], [61]. These results support the design choice of *semantics-at-the-edge* with principled cooperation, and suggest that the broader edge AI community can leverage similar interfaces to translate continuous sensor streams into verifiable, privacy-preserving language understanding.

REFERENCES

[1] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2024.

[2] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[3] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters. In *Proceedings of MLSys*, 2024.

[4] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2024.

[5] Jianchun Liu, Shilong Wang, Hongli Xu, Yang Xu, Yunming Liao, Jinyang Huang, and He Huang. Federated learning with experience-driven model migration in heterogeneous edge networks. *IEEE/ACM Transactions on Networking*, 32(4):3468–3484, 2024.

[6] Kaiqi Hong et al. Flashdecoding++: Faster large language model inference on gpus. In *Proceedings of MLSys*, 2024.

[7] Haoyi Wu and Kewei Tu. Layer-condensed kv cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the ACL (Long Papers)*, 2024.

[8] Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. Serverlessllm: Low-latency serverless inference for large language models. *arXiv preprint arXiv:2401.14351*, 2024.

[9] Meng Wang, Jinyang Huang, Xiang Zhang, Zhi Liu, Meng Li, Peng Zhao, Huan Yan, Xiao Sun, and Mianxiong Dong. Target-oriented wifi sensing for respiratory healthcare: from indiscriminate perception to in-area sensing. *IEEE Network*, pages 1–1, 2024.

[10] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*, 2024.

[11] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[12] Yuyan Liu et al. Cachegen: Kv cache compression and streaming for fast context loading of llms. In *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024.

[13] Jinyang Huang, Bin Liu, Chenglin Miao, Yan Lu, Qijia Zheng, Yu Wu, Jiancun Liu, Lu Su, and Chang Wen Chen. Phaseanti: An anti-interference wifi-based activity recognition system using interference-independent phase component. *IEEE Transactions on Mobile Computing*, 22(5):2938–2954, 2023.

[14] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2303.06865*, 2023.

[15] Yuan He, Lirui Liu, Jing Liu, Weijia Wu, Hao Zhou, and Bohan Zhuang. Zipcache: Accurate and efficient kv cache quantization for llms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[16] Yinmin Zhong et al. Distserve: Disaggregating prefill and decoding for llm serving. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[17] Jianchun Liu, Jiaming Yan, Hongli Xu, Zhiyuan Wang, Jinyang Huang, and Yang Xu. Finch: Enhancing federated learning with hierarchical neural architecture search. *IEEE Transactions on Mobile Computing*, 23(5):6012–6026, 2024.

[18] Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *arXiv preprint arXiv:2404.12457*, 2024.

[19] Xinqi Chu, Jingdong Chen, Jun Yin, Chuang Niu, Zhuliang Yao, Yilei Huang, Tianbao Li, and et al. Mobilevlm v2: Faster and stronger baseline for vision language model on mobile devices. *arXiv preprint arXiv:2402.03766*, 2024.

[20] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[21] Xiang Zhang, Yan Lu, Huan Yan, Jinyang Huang, Yu Gu, Yusheng Ji, Zhi Liu, and Bin Liu. Resup: Reliable label noise suppression for facial expression recognition. *IEEE Transactions on Affective Computing*, pages 1–14, 2025.

[22] Mingjin Zhang, Xiaoming Shen, Jiannong Cao, Zeyang Cui, and Shihan Jiang. Edgeshard: Efficient llm inference via collaborative edge computing. *arXiv preprint arXiv:2405.14371*, 2024.

[23] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput–latency tradeoff in llm inference with sarathi-serve. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[24] Lee et al. Infinigen: Efficient llm inference via dynamically managed kv cache. In *USENIX OSDI*, 2024.

[25] Yu Gu, Huan Yan, Xiang Zhang, Yantong Wang, Jinyang Huang, Yusheng Ji, and Fuji Ren. Attention-based gesture recognition using commodity wifi devices. *IEEE Sensors Journal*, 23(9):9685–9696, 2023.

[26] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the ACL (System Demonstrations)*, pages 558–568, 2023.

[27] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hongyu Wu, Jeff Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2023.

[28] Jianbo Wu, Jie Ren, Shuangyan Yang, Konstantinos Parasyris, Giorgis Georgakoudis, Ignacio Laguna, and Dong Li. Lm-offload: Performance model-guided generative inference of large language models with parallelism control. *Technical Report*, 2024.

[29] Jinyang Huang, Bin Liu, Chao Chen, Hongxin Jin, Zhiqiang Liu, Chi Zhang, and Nenghai Yu. Towards anti-interference human activity recognition based on wifi subcarrier correlation selection. *IEEE Transactions on Vehicular Technology*, 69(6):6739–6754, 2020.

[30] Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*, 2024.

[31] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2024.

[32] Peng Zhao, Jinyang Huang, Xiang Zhang, Zhi Liu, Huan Yan, Meng Wang, Guohang Zhuang, Yutong Guo, Xiao Sun, and Meng Li. Wipulmo: Commodity wifi can capture your pulmonary function without mouth clinging. *IEEE Internet of Things Journal*, 12(1):854–868, 2025.

[33] Korakit Seemakhupt, Sihang Liu, and Samira Khan. Edgerag: Online-indexed rag for edge devices. *arXiv preprint arXiv:2412.21023*, 2024.

[34] Yu Gu, Xiang Zhang, Huan Yan, Jingyang Huang, Zhi Liu, Mianxiong Dong, and Fuji Ren. Wife: Wifi and vision based unobtrusive emotion recognition via gesture and facial expression. *IEEE Transactions on Affective Computing*, 14(4):2567–2581, 2023.

[35] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, S. Karen Khatamifard, Minsik Cho, Carlo C. Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*, 2024.

[36] Zhenyu Zhang et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[37] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[38] Jinyang Huang, Jia-Xuan Bai, Xiang Zhang, Zhi Liu, Yuanhao Feng, Jianchun Liu, Xiao Sun, Mianxiong Dong, and Meng Li. Keystrokesniffer: An off-the-shelf smartphone can eavesdrop on your privacy from anywhere. *IEEE Transactions on Information Forensics and Security*, 19:6840–6855, 2024.

[39] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jia-Bin Huang, Jinglin Liu, Yi Ren, Zhou Zhao, and Shinji Watanabe. Audiogpt: Understanding and generating speech, music, sound, and talking head. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[40] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[41] Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

[42] Jinyang Huang, Bin Liu, Chenglin Miao, Xiang Zhang, Jianchun Liu, Lu Su, Zhi Liu, and Yu Gu. Phyfinatt: An undetectable attack framework against phy layer fingerprint-based wifi authentication. *IEEE Transactions on Mobile Computing*, 23(7):7753–7770, 2024.

[43] Hongpeng Jin and Yanzhao Wu. Ce-collm: System design for efficient cloud-edge collaborative large language models. *arXiv preprint arXiv:2411.02829*, 2024.

[44] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2024.

[45] Xiang Zhang, Jinyang Huang, Huan Yan, Yuanhao Feng, Peng Zhao, Guohang Zhuang, Zhi Liu, and Bin Liu. Wiopen: A robust wi-fi-based open-set gesture recognition framework. *IEEE Transactions on Human-Machine Systems*, 55(2):234–245, 2025.

[46] Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, 2025.

[47] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023.

[48] Yuanhao Feng, Jinyang Huang, Youwei Zhang, Xiang Zhang, Meng Li, Fusang Zhang, Tianyue Zheng, Anran Li, Mianxiong Dong, and Zhi Liu. Rf-eye: Commodity rfid can know what you write and who you are wherever you are. *ACM Transactions on Sensor Networks*, 2025.

[49] Zonghan Yang, Yu Yu, Qianlin Wang, Weibin Meng, Peng Cheng, Hongke Zhou, and Siheng Chen. Customizing llms for efficient latency-aware inference at the edge. In *USENIX Annual Technical Conference (ATC)*, 2025.

[50] Wang Ye et al. Instinfer: In-storage attention offloading for cost-effective llm inference. *arXiv preprint arXiv:2409.04992*, 2024.

[51] Jianchun Liu, Hongli Xu, Lun Wang, Yang Xu, Chen Qian, Jinyang Huang, and He Huang. Adaptive asynchronous federated learning in resource-constrained edge computing. *IEEE Transactions on Mobile Computing*, 22(2):674–690, 2021.

[52] Liang Chen et al. Punica: Multi-tenant lora serving. In *Proceedings of MLSys*, 2024.

[53] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, PMLR, 2024.

[54] Christopher Holmes, Sefa Tanaka, et al. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *arXiv preprint arXiv:2401.08671*, 2024.

[55] Jianchun Liu, Hongli Xu, Yang Xu, Zhenguo Ma, Zhiyuan Wang, Chen Qian, and He Huang. Communication-efficient asynchronous federated learning in resource-constrained edge computing. *Computer Networks*, 199:108429, 2021.

[56] Zhiyuan Wang, Hongli Xu, Jianchun Liu, He Huang, Chunming Qiao, and Yangming Zhao. Resource-efficient federated learning with hierarchical aggregation in edge computing. In *IEEE INFOCOM 2021-IEEE conference on computer communications*, pages 1–10. IEEE, 2021.

[57] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[58] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

[59] Feng-Qi Cui, Anyang Tong, Jinyang Huang, Jie Zhang, Dan Guo, Zhi Liu, and Meng Wang. Learning from heterogeneity: Generalizing dynamic facial expression recognition via distributionally robust optimization. In *Proceedings of the 33nd ACM International Conference on Multimedia*, MM '25', New York, NY, USA, 2025. Association for Computing Machinery.

[60] Zhiwei Yao, Jianchun Liu, Hongli Xu, Lun Wang, Chen Qian, and Yunming Liao. Ferrari: A personalized federated learning framework for heterogeneous edge clients. *IEEE Transactions on Mobile Computing*, 23(10):10031–10045, 2024.

[61] Jiaming Yan, Jianchun Liu, Hongli Xu, Zhiyuan Wang, and Chunming Qiao. Peaches: Personalized federated learning with neural architecture search in edge computing. *IEEE Transactions on Mobile Computing*, 23(11):10296–10312, 2024.

[62] Rukuo Li, Jianchun Liu, Hongli Xu, and Liusheng Huang. Fedquad: Adaptive layer-wise lora deployment and activation quantization for federated fine-tuning. *arXiv preprint arXiv:2506.01001*, 2025.

[63] Jianchun Liu, Jiaming Yan, Ji Qi, Hongli Xu, Shilong Wang, Chunming Qiao, and Liusheng Huang. Adaptive local update and neural composition for accelerating federated learning in heterogeneous edge networks. *IEEE Transactions on Networking*, 2025.

[64] Jianchun Liu, Jiaming Yan, Hongli Xu, Lun Wang, Zhiyuan Wang, Jinyang Huang, and Chunming Qiao. Accelerating decentralized federated learning with probabilistic communication in heterogeneous edge computing. *IEEE Transactions on Networking*, 2025.

[65] Jiaming Yan, Jianchun Liu, Hongli Xu, and Liusheng Huang. Accelerating mixture-of-expert inference with adaptive expert split mechanism. *arXiv preprint arXiv:2509.08342*, 2025.

[66] Shilong Wang, Jianchun Liu, Hongli Xu, Chenxia Tang, Qianpiao Ma, and Liusheng Huang. Towards communication-efficient decentralized federated graph learning over non-iid data. *arXiv preprint arXiv:2509.08409*, 2025.

[67] Chenxia Tang, Jianchun Liu, Hongli Xu, and Liusheng Huang. Top-n: Eliminating noise in logit space for robust token sampling of llm. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10758–10774, 2025.

[68] Yujia Huo, Jianchun Liu, Hongli Xu, Zhenguo Ma, Shilong Wang, and Liusheng Huang. Mitigating catastrophic forgetting with adaptive transformer block expansion in federated fine-tuning. *arXiv preprint arXiv:2506.05977*, 2025.

[69] Luyao Gao, Jianchun Liu, Hongli Xu, Sun Xu, Qianpiao Ma, and Liusheng Huang. Accelerating end-cloud collaborative inference via near bubble-free pipeline optimization. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2025.

[70] Akrit Mudvari, Yuang Jiang, and Leandros Tassiulas. Splitllm: Efficient collaborative inference of large language models with split inference. *arXiv preprint arXiv:2410.10759*, 2024.

[71] Foteini Strati, Sara McAllister, Amar Phanishayee, Jakub Tarnawski, and Ana Klimovic. DéjàVu: KV-cache streaming for fast, fault-tolerant generative LLM serving. In *Proceedings of the 41st International Conference on Machine Learning*, PMLR, pages 46745–46771, 2024.

[72] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[73] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[74] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[75] Jianchun Liu, Jun Liu, Hongli Xu, Yunming Liao, Zhiwei Yao, Min Chen, and Chen Qian. Enhancing semi-supervised federated learning with progressive training in heterogeneous edge computing. *IEEE Transactions on Mobile Computing*, 2024.

[76] Jun Liu, Jianchun Liu, Hongli Xu, Yunming Liao, Zhiyuan Wang, and Qianpiao Ma. Yoga: Adaptive layer-wise model aggregation for decentralized federated learning. *IEEE/ACM Transactions on Networking*, 32(2):1768–1780, 2023.

[77] Jianchun Liu, Qingmin Zeng, Hongli Xu, Yang Xu, Zhiyuan Wang, and He Huang. Adaptive block-wise regularization and knowledge distillation for enhancing federated learning. *IEEE/ACM Transactions on Networking*, 32(1):791–805, 2023.

[78] Ruiyang Qin, Zheyu Yan, Dewen Zeng, Zhenge Jia, Dancheng Liu, Jianbo Liu, Zhi Zheng, Ningyuan Cao, Kai Ni, Jinjun Xiong, and Yiyu Shi. Robust implementation of retrieval-augmented generation on edge-based computing-in-memory architectures. *arXiv preprint arXiv:2405.04700*, 2024.