SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

# Reasoning Allocation as a Privacy Architecture: Tiered Inference for Trusted Execution Environment Deployments

Ankur Aggarwal

Meta, USA

**Abstract**
Trusted Execution Environments (TEEs) impose a paradox on large language model (LLM) deployment: artificial intelligence (AI) optimization requires performance data, yet the stateless, non-targetable, anonymous architecture of production confidential computing systems is explicitly designed to prevent data collection. This paper reframes reasoning allocation, routing queries to appropriately sized models based on complexity, as a structural privacy architecture pattern rather than a cost optimization. Every query resolved before the TEE boundary eliminates both privacy risk and compute overhead simultaneously. Drawing on production deployments from Meta's WhatsApp Private Processing and Apple's Private Cloud Compute (PCC), and grounding the analysis in verified research on model cascading, prompt compression, federated learning, and in-enclave quality evaluation, this work presents a five-layer tiered inference optimization stack and a privacy-safe signal taxonomy enabling continuous system improvement without any user content leaving the device. The architecture projects approximately 55% compute reduction at mature calibration under stated modeling assumptions while maintaining the privacy guarantees that define production-confidential AI. The framework is directly applicable to any stateless, non-targetable TEE-based LLM deployment.

**Keywords:** Reasoning Allocation, Trusted Execution Environment, Federated Learning, Prompt Compression, LLM Routing, Privacy-Preserving AI, Model Cascading.

## 1. Introduction

Standard AI development practice depends on observation. Latency metrics, quality evaluations, user engagement signals, and error rates feed iterative improvement cycles that close the gap between a system's initial deployment state and its production-optimized form. So routine is this feedback loop that its absence is rarely treated as a first-class architectural concern. Trusted Execution Environments impose exactly that absence. Meta's WhatsApp Private Processing [1], the most complete public realization of confidential AI inference at consumer scale, operates under three properties that eliminate conventional observability entirely: it is stateless (no data survives between requests), non-targetable (the infrastructure cannot route a specific user to a specific confidential virtual machine (CVM)), and anonymous (no user identity enters the enclave). These are not limitations waiting to be engineered around; they are the security guarantees that make TEE-based AI credible to users and regulators alike.

The optimization paradox follows directly. Improving a TEE-deployed LLM requires knowing how it performs. The architecture is built to ensure that no such knowledge can accumulate. Privacy and optimization appear structurally opposed. Reasoning allocation, routing queries to appropriately sized models based on estimated complexity, dissolves this tension through an architectural insight rather than a technical workaround: a query

that never enters the TEE carries zero privacy risk and contributes zero TEE compute overhead. The routing decision itself, and the signals it generates, are observable on the client device without any enclave involvement. Reasoning allocation is therefore not a performance optimization layered on top of a privacy-preserving system. It is itself a privacy architecture pattern that structurally reduces the system's TEE exposure surface while simultaneously enabling a client-observable signal stream rich enough to support continuous improvement.

Apple's Private Cloud Compute [2] demonstrates this principle in production. On-device models running on the Apple Neural Engine handle simpler queries locally, escalating only those genuinely requiring cloud-level reasoning capability to PCC nodes. The effect is threefold: fewer queries cross the client-cloud boundary (reduced privacy exposure), PCC nodes handle a smaller fraction of total volume (reduced infrastructure cost), and on-device routing generates client-observable signals with no TEE involvement (continuous optimization without content collection). WhatsApp's architecture has publicly indicated similar complexity routing as a planned capability. The research foundations for a full five-layer optimization stack are well established across model cascading [3, 4, 5], on-device routing [6], prompt compression [7], and privacy-preserving federated learning [13, 14].

This paper makes three contributions. First, it reframes reasoning allocation as a privacy architecture primitive. Second, it synthesizes five layers of verified research into a deployable optimization stack applicable to any stateless, non-targetable TEE deployment. Third, it introduces a privacy-safe signal taxonomy, six categories of metadata collectible without user content exposure, as the operational foundation for continuous router improvement. The paper proceeds as follows: Section 2 characterizes TEE constraints and their optimization consequences; Sections 3 and 4 survey the research foundations for tiered inference and payload minimization; Section 5 presents the signal taxonomy; Section 6 covers federated learning for router improvement; Section 7 synthesizes the five-layer stack; Section 8 discusses open problems; and Section 9 concludes.

## 2. TEE Architectural Constraints and the Optimization Paradox

### 2.1 Production TEE Constraints

The three defining properties of WhatsApp's Private Processing [1] create the optimization paradox in concrete operational form. Statelessness eliminates every form of persistent logging and telemetry. No query content, model output, performance signal, or error trace survives across request boundaries. Each inference session is cryptographically isolated; when it ends, all associated state is destroyed. Standard AI observability pipelines, which depend on accumulated session logs for quality analysis, error attribution, and regression detection, have no surface to attach to. Non-targetability removes any possibility of user-level analysis. Because the infrastructure cannot route a specific user's request to a specific CVM, it also cannot observe whether a given user consistently receives low-quality responses or whether certain query categories are systematically underserved by the deployed model configuration. Anonymity eliminates the identity anchors that personalization and cohort-based quality analysis require.

These three properties are not independent choices that could be relaxed selectively. They form a mutually reinforcing package required by the threat model, which accounts for compromised insiders, supply chain attacks, and adversarial infrastructure operators. Relaxing statelessness to retain per-session telemetry would enable cross-session correlation attacks. Relaxing non-targetability would enable adversarial routing. Relaxing anonymity would expose users to the infrastructure provider. Any optimization strategy must therefore operate within all three constraints simultaneously. That constraint eliminates every conventional AI observability approach and makes client-side signal collection, the only layer that falls outside the TEE boundary, the sole viable path to continuous improvement. Reasoning allocation's primary value is that it generates this client-side signal stream as a natural byproduct of its routing decisions.

The performance cost of TEE operation adds a second dimension to the problem. Zhu et al. [10] benchmarked LLM inference on NVIDIA Hopper GPUs in confidential compute mode and found that for the majority of typical LLM queries, the overhead remains below approximately 5%, with larger models and longer sequences experiencing up to approximately 10–15% overhead, primarily attributable to CPU-GPU data transfer via PCIe. For a system routing all queries to a single large model inside the TEE, this overhead applies to every query processed. Reasoning allocation eliminates the overhead entirely for queries deflected to on-device models and reduces it proportionally for queries routed to smaller in-TEE tiers. The overhead savings thus compound with

the base compute savings, producing the approximately 55% cost reduction projected in Table 2 under the stated modeling assumptions.

### 2.2 Comparative Architecture: WhatsApp Private Processing vs. Apple PCC

Meta and Apple have independently converged on architecturally similar solutions to the confidential cloud AI problem. Table 1 presents a structured comparison across five dimensions. The critical divergence is on-device-first routing: Apple's PCC architecture already implements reasoning allocation in production through Apple Neural Engine models, while WhatsApp has indicated similar complexity routing as a planned capability. This convergence of two independent production systems on the same core security properties and Apple's demonstrated extension of those properties into on-device routing provides strong empirical validation that the five-layer optimization stack proposed in this paper is not speculative. Apple's production deployment establishes that the on-device routing tier is implementable at consumer scale within the constraints of a non-targetable, stateless confidential AI architecture.

**Table 1: Architectural Comparison: WhatsApp Private Processing vs. Apple Private Cloud Compute**

| Property | WhatsApp Private Processing | Apple Private Cloud Compute |
|---|---|---|
| Stateless computation | No data logged or retained between requests | No data retained after processing |
| Non-targetability | Cannot route specific users to specific TEEs | Apple cannot target individual users to specific nodes |
| Verifiable transparency | CVM binary digests are open to external review and attestation | Software images published for independent audit |
| On-device-first routing | Partial, complexity routing planned | On-device models (Apple Neural Engine) handle simple tasks |
| Encrypted transport | End-to-end encryption + OHTTP (RFC 9458) | End-to-end encryption to PCC nodes |

Sources: Meta Engineering Blog [1]; Apple Security Research [2]. Key divergence: Apple PCC implements on-device-first reasoning allocation in production; WhatsApp has indicated similar routing as a planned capability.

The practical consequence of Apple's on-device routing is that PCC nodes process only queries genuinely requiring cloud-level reasoning. This simultaneously compresses the privacy exposure surface (fewer queries cross the device boundary), reduces infrastructure utilization (PCC nodes handle a smaller volume), and enriches the client-observable signal stream (on-device queries generate timing, routing, and engagement signals without any TEE involvement). Three distinct optimization objectives are advanced by a single architectural decision, which is the clearest possible illustration of why reasoning allocation deserves treatment as a privacy architecture primitive rather than an operational cost-saving measure added after the privacy-preserving design is otherwise complete.

### 2.3 The TEE Overhead Tax and Routing Economics

The economics of reasoning allocation in a TEE context differ from standard model routing in one structurally important way: routing a query away from the TEE eliminates not only the base compute cost of the large model but also the TEE overhead multiplier that would have applied to that query. Table 2 quantifies this compounding effect under illustrative modeling assumptions. In a baseline system routing all queries to a full TEE LLM, a midpoint TEE overhead of approximately 10% (within the range reported by Zhu et al. [10]) applies universally, bringing the effective cost per query to approximately 1.10 normalized units. With reasoning allocation deflecting queries across three tiers, drawing on cost-reduction ratios from RouteLLM [6] as a conservative basis for on-device deflection, the total cost falls from 1,100 units to approximately 499 units under the stated assumptions, a reduction of approximately 55%. The tier cost values (on-device at 0.1 units and small TEE model at 0.5 units) and the illustrative routing split (~40% on-device, ~35% small TEE, and ~25% large TEE) are author-defined modeling parameters; actual savings will vary with deployment-specific query distributions, hardware configuration, and routing accuracy.

**Table 2: TEE Overhead Compounding, Baseline vs. Reasoning Allocation (1,000 queries, illustrative)**

| Scenario | Query Distribution | Compute Cost (units) | TEE Queries |
|---|---|---|---|
| Baseline, no routing | 100% → Full TEE LLM | 1,100 (×1.10 mid-point TEE overhead) | 1,000 / 1,000 |

| | | | |
|---|---|---|---|
| With reasoning allocation | ~40% on-device, ~35% small TEE, ~25% large TEE (illustrative split) | ~499 (~55% reduction) | ~600 / 1,000 |

All figures are theoretical projections under author-defined modeling assumptions. The TEE overhead mid-point (~10%) is drawn from the range reported by Zhu et al. [10] for NVIDIA Hopper GPUs. On-device deflection rate (~40%) is a conservative estimate drawing on RouteLLM cost-reduction ratios [6] as a proxy; RouteLLM does not directly report mobile on-device deflection rates. Tier cost units (on-device = 0.1; small TEE = 0.5; large TEE = 1.0) are author-defined normalizations. The ~55% reduction is a theoretical projection; actual savings will vary with deployment-specific conditions. These figures should not be interpreted as empirically validated results.

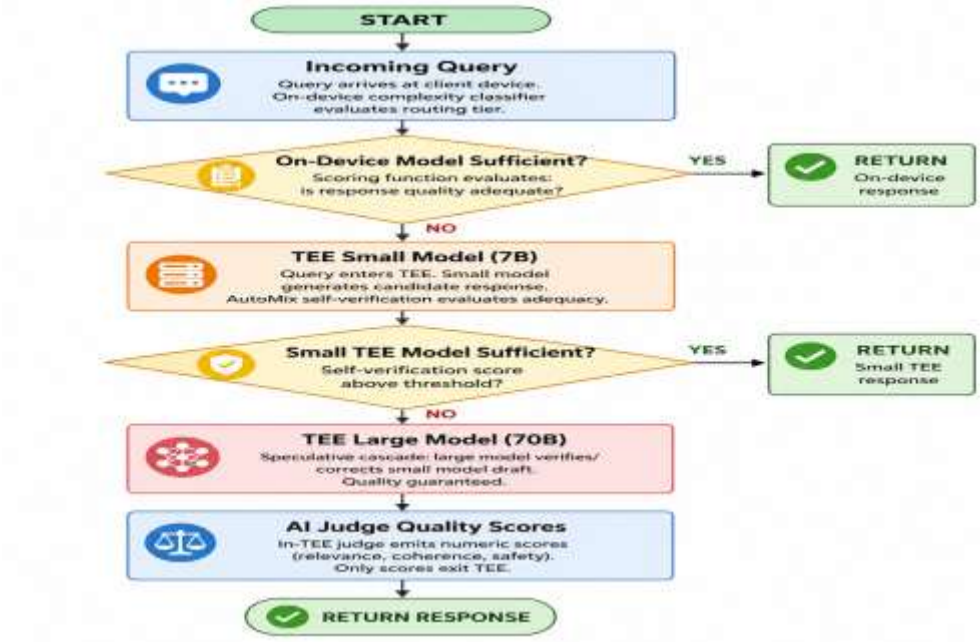
## 3. Tiered Inference: The Research Foundation

### 3.1 FrugalGPT: Cost-Quality Cascading

Chen, Zaharia, and Zou [3] introduced FrugalGPT (arXiv:2305.05176) as a framework reducing LLM API costs through three strategies: prompt adaptation (token count reduction), LLM approximation (model cascading governed by a learned scoring function), and LLM cascade with scoring (a judge model evaluates response adequacy before escalation). Empirically, FrugalGPT achieved up to 98% cost reduction relative to GPT-4-only inference while matching or exceeding accuracy on standard benchmarks, using a GPT-J → GPT-3.5 → GPT-4 cascade with a learned escalation scorer. For TEE deployments, the crucial property of the cascade scoring function is not its accuracy but its output: a single binary escalation bit. That bit carries no user content and generates no privacy exposure, yet it constitutes exactly the routing signal needed for system optimization. An adversary observing a stream of escalation bits learns the distribution of query complexity, a useful but non-sensitive aggregate statistic, and nothing about any individual query's content.

Figure 1 illustrates the FrugalGPT-inspired TEE cascade adapted for a three-tier confidential deployment. The on-device tier handles the first decision gate; a small in-TEE model handles the second (approximately 7B parameters is an illustrative tier size; actual deployment sizes depend on hardware) constraints and quality requirements; and a large in-TEE model (approximately 70B parameters, also illustrative) handles queries that neither smaller tier can resolve adequately. Self-verification at the small model stage (Section 3.3) runs inside the enclave with no cross-boundary data movement. The scoring function at the on-device → TEE transition runs entirely on the client device, operating on query characteristics, vocabulary complexity, length, and syntactic structure without touching query content.

**Figure 1: FrugalGPT-Inspired Three-Tier TEE Cascade Flowchart**



Recreated from the client-provided diagram. Each decision diamond emits only a binary escalation signal; no query content crosses a trust boundary as part of the routing decision. Model tier sizes (~7B and ~70B parameters) are illustrative architectural parameters; deployment teams should calibrate based on hardware constraints and quality requirements.

The 98% cost reduction figure from FrugalGPT [3] represents the ceiling achievable in a proprietary API context with unlimited model selection and favorable benchmark conditions. In a TEE context, the ceiling is lower: on-device model quality is constrained by device hardware and model size limits. The illustrative routing split in Table 2 draws on cost-reduction ratios from RouteLLM [6] as a conservative basis; deployment teams should calibrate routing thresholds against their specific query distributions.

### 3.2 RouteLLM: Learned Query Routing

Ong et al. [6] introduced RouteLLM (arXiv:2406.18665) from the University of California, Berkeley, and Anyscale, a framework for training lightweight router models on human preference data from Chatbot Arena to dynamically select between stronger and weaker LLMs. The key empirical result: up to 85% cost reduction while maintaining 95% of GPT-4 quality on MT-Bench, using routers trained entirely on public preference data with no access to private user sessions. Table 3 presents the four evaluated architectures. For the on-device TEE routing role, the similarity-weighted ranking and matrix factorization routers are the most suitable: both add sub-2 ms latency as reported in Ong et al. [6], operate on query surface characteristics (length, vocabulary, and question type) computable locally without content access, and are lightweight enough to run on mobile hardware. The latency overhead of either architecture is small relative to the relay overhead present in the WhatsApp Private Processing stack, which uses OHTTP for anonymous request routing [12].

**Table 3: RouteLLM Router Architecture Comparison**

| Router Type | Mechanism | Latency | TEE Suitability |
|---|---|---|---|
| Similarity-weighted ranking | Embedding similarity to labeled examples | ~2ms | Excellent, on-device, no user content |
| Matrix factorization | Learned user-model interaction matrix | ~1ms | Excellent, lightweight, on-device |
| BERT classifier | Fine-tuned BERT on preference pairs | ~5ms | Good, slightly higher latency |
| Causal LLM router | Fine-tuned a small LM with augmented training | ~15ms | Acceptable for non-latency-sensitive tasks |

Router architecture and latency figures based on Ong et al. [6]. TEE suitability is assessed against the on-device deployment requirement: it operates on query characteristics without content access, has sub-5 ms latency, and is suitable for mobile hardware constraints.

The training independence of RouteLLM routers from private user data is as important as their accuracy. The routers learn query complexity patterns from the Chatbot Arena public preference dataset and generalize to new deployment contexts. A router deployed in a healthcare assistant, a legal review tool, or a financial compliance system needs no access to those systems' query histories to provide an initial calibration; public data establishes the baseline, and the federated learning approach in Section 6 refines it over time using client-observable signals rather than session content. This separation of training (public data, pre-deployment) from refinement (private signals, federated post-deployment) is a core architectural property of the five-layer stack.

### 3.3 AutoMix: Self-Verification Cascading

Aggarwal et al. [4] proposed AutoMix (arXiv:2310.12963), in which the small model evaluates its own output using a few-shot self-verification prompt and escalates to the large model only when its confidence falls below a threshold. The approach eliminates the separate scoring model required by FrugalGPT: verification runs within the same model that produced the candidate response. AutoMix demonstrated over 50% computational cost reduction for comparable performance, with incremental benefit-per-cost improvements exceeding 10% on Llama-2-13B → Llama-2-70B routing [4]. For TEE deployment, self-verification is a more architecturally appropriate escalation mechanism than an external scorer for two reasons. The verification computation runs inside the enclave; no data leaves for the routing decision, and the mechanism requires no external training data pipeline, reducing the number of attested software components inside the CVM boundary.

### 3.4 Speculative Cascades: Latency-Cost Co-Optimization

Narasimhan et al. [5] at Google Research introduced faster cascades via speculative decoding, combining speculative decoding with model cascading such that the small model's output serves as a draft for the large

model to verify rather than generate from scratch. When the draft passes a quality threshold, it is returned directly; when it fails, the large model verifies and corrects the draft rather than performing full autoregressive generation. Experiments with Gemma and T5 models on a range of language benchmarks show that the approach yields better cost-quality trade-offs than cascading and speculative decoding baselines individually [5]. For TEE deployments, the speculative cascade is particularly valuable because it reduces the per-query compute cost even for queries that require the large model, the most computationally expensive tier. The draft-versus-generate distinction also provides a session-observable proxy for query difficulty: a high token-level divergence rate between draft and verification output indicates a genuinely complex query, constituting a cascade signal (Table 4) usable for router calibration without content exposure.

## 4. Minimizing Data Payloads Into the TEE

### 4.1 Prompt Compression: LLMLingua

Jiang et al. [7] introduced LLMLingua (EMNLP 2023, arXiv:2310.05736), a coarse-to-fine prompt compression framework using a small language model to compute per-token perplexity and selectively retain high-information tokens while dropping low-perplexity redundant content. The approach achieves up to 20× compression on certain benchmarks with minimal quality degradation, specifically, up to 20× compression on GSM8K with only 1.5% performance loss, as reported in Jiang et al. [7], and the compression model runs on-device without cloud access. For TEE deployments, LLMLingua creates compounding benefits: a compressed prompt contains fewer tokens carrying user-specific content. reducing payload size, enclave processing time, and the amount of user data at risk in any hypothetical TEE compromise. Smaller payloads also reduce the egress risk if any portion of the prompt is included in an external tool call from the enclave.

The compression ratio is itself a privacy-safe optimization signal. A query compressing to 10% of its original length has fundamentally different semantic characteristics from one compressing to 80%; the former is formulaic or templated, the latter is dense and unique. This ratio is computed on-device, available as a compression signal (Table 4), and usable for LLMLingua model calibration without query content exposure. The compression step also functions as a first-pass PII stripping layer: high-perplexity tokens that the compression model identifies as informationally dense, which frequently include named entities, account identifiers, and specific dates, can be flagged for selective suppression before TEE transmission, providing a data minimization control that operates independently of the primary compression objective.

### 4.2 Matryoshka Representations for Adaptive Context

Kusupati et al. [8] introduced Matryoshka Representation Learning (MRL) at NeurIPS 2022 (arXiv:2205.13147), training embedding models to produce representations useful at any truncation point. The paper demonstrates that embeddings can be truncated to nested subspaces at various fractions of the full dimensionality; for BERT and ViT architectures with 768-dimensional embeddings, subspaces at d/2, d/4, d/8, and so on remain competitive at retrieval and classification tasks, with dimensionality controlling the granularity-versus-compute trade-off [8]. For TEE deployments, MRL embeddings serve as an adaptive context transmission mechanism: drawing on this nested subspace principle, simple queries that require minimal contextual grounding can transmit low-dimensional embeddings to the enclave (for example, 64 or 128 dimensions, illustrative values consistent with the MRL nested subspace approach), reducing both payload size and information content; complex multi-turn queries requiring rich contextual grounding transmit the full-dimensional representation.

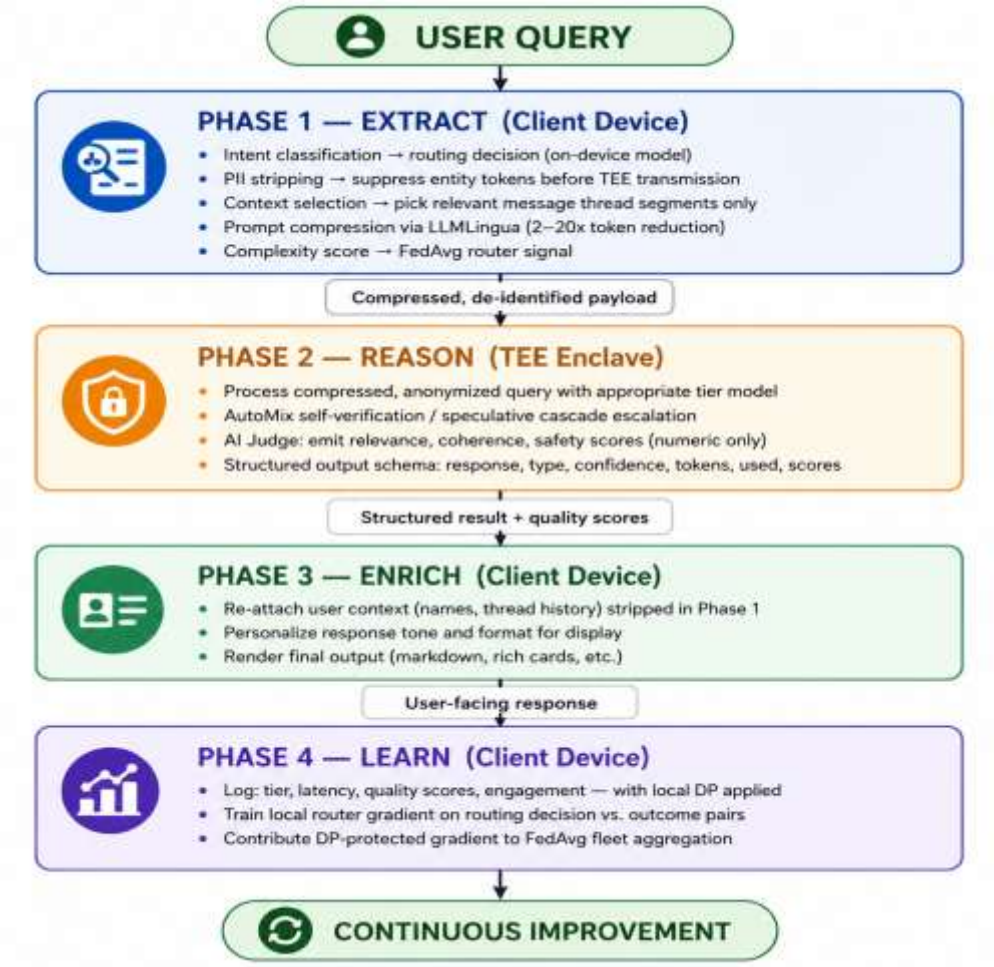
The alignment between routing complexity assessment and context dimensionality selection is natural and requires no separate decision mechanism. Queries the on-device router classifies as Tier 1 (on-device) transmit zero context to the TEE. Queries routed to Tier 2 (small TEE model) transmit a low-dimensional context embedding; the specific dimensionality is a deployment parameter calibrated to the tier model's context requirements. Queries routed to Tier 3 (large TEE model) transmit the full-dimensional representation. This graduated context policy ensures that the TEE receives only as much contextual information as the assigned tier model actually needs, a data minimization principle operationalized through the router's own classification output rather than through a separate policy enforcement mechanism.

### 4.3 The Extract-Reason-Enrich-Learn Pipeline

The individual payload minimization techniques in Sections 4.1 and 4.2 integrate into a coherent four-phase pipeline for TEE-aware query processing, illustrated in Figure 2. The Extract phase runs entirely on the client: intent classification and complexity scoring (routing decision); PII stripping (entity token suppression);

context selection (relevant thread segment identification); and LLMLingua compression (up to 20× token reduction per Jiang et al. [7]). The Reason phase runs in the TEE: the tier-appropriate model processes the compressed, de-identified query; AutoMix self-verification or speculative cascade governs escalation between sub-tiers; the in-TEE AI Judge emits numeric quality scores; and the structured output schema ensures every response carries machine-readable quality metadata. The Enrich phase runs on the client: user-specific context stripped in the Extract phase is reattached, response tone and format are personalized, and the final output is rendered for display.

**Figure 2: Extract-Reason-Enrich-Learn Pipeline**




Recreated from the client-provided diagram. The TEE (Phase 2) processes only the compressed, de-identified core of each query. Personalization (Phase 3) and signal collection (Phase 4) run on the client device, outside the enclave boundary.

The Learn phase, the fourth phase, runs on the client and forms the bridge between the pipeline and the federated learning system in Section 6. Every routing decision, latency measurement, quality score, and engagement observation is logged locally with local differential privacy (DP) applied before any signal leaves the device. The local router model is updated incrementally using the accumulated signal observations, and a DP-protected gradient is contributed to the fleet-wide FedAvg aggregation. The pipeline's key architectural invariant is that the TEE sees only the compressed, de-identified, context-minimized core of the query, never the full original message with user-specific content reattached. The Enrich phase ensures the user receives a contextually appropriate, personalized response, but the personalization computation never enters the enclave.

## 5. Privacy-Safe Optimization Signals

### 5.1 The Signal Taxonomy

The central claim of this paper rests on the existence of informative optimization signals that carry no user content. Table 4 presents a taxonomy of six such signal categories. Routing signals describe the routing decision, which tier was selected, the router's confidence score, and whether a fallback from the primary selection occurred, but not the query that prompted it. Performance signals are timing measurements and TEE-emitted error codes, both content-free and observable from the client or from structured enclave outputs. Quality proxies require the in-TEE AI Judge (Section 5.3) and exit the TEE only as four numeric values with no association to query text. Engagement signals are behavioral; the user accepted, rejected, or edited the response generated after the user interacts with the output, requiring no content observation. Cascade signals describe the escalation pattern, rate, depth, and abort rate without reference to the queries that triggered escalation. Compression signals capture the LLMLingua compression ratio and perplexity distribution, computable on-device before TEE transmission.

**Table 4: Privacy-Safe Optimization Signal Taxonomy**

| Signal Category | Specific Metrics | Collection Point | Privacy Risk | Optimization Value |
|---|---|---|---|---|
| Routing Signals | Tier selected, router confidence, fallback events | Client | None | High |
| Performance Signals | Latency per tier, tokens/sec, time-to-first-token | Client + TEE error codes | None | High |
| Quality Proxies | AI Judge scores (numeric), self-verification pass/fail | In-TEE | Minimal (numeric) | Very High |
| Engagement Signals | Response accepted/rejected/edited, feedback rating | Client | Low (behavioral) | High |
| Cascade Signals | Escalation rate, escalation depth, cascade abort rate | Client | None | High |
| Compression Signals | Compression ratio, tokens before/after, perplexity distribution | Client | None | Medium |

Six signal categories are collectible without user content access. Quality proxies require an in-TEE AI Judge; all other categories are collected on the client device. Local differential privacy is applied before any signal leaves the client for fleet-wide aggregation.

The aggregate information value of these six categories is substantial enough to support meaningful router optimization without any content exposure. Routing signals reveal query complexity distribution across the fleet, calibrating the on-device router threshold. Performance signals identify latency bottlenecks by tier. Quality proxy scores provide ground-truth quality evaluation from inside the enclave. Engagement signals offer user-level quality feedback correlated with judge scores to validate judge calibration. Cascade signals reveal whether the tiered routing system is accurately matching complexity to capability; a persistently high escalation rate indicates the router threshold is set too conservatively. Compression signals track whether LLMLingua quality degrades over time for specific query categories, triggering model updates.

### 5.2 Differential Privacy for Aggregate Signals

Even content-free metadata signals benefit from local DP protection before fleet-wide transmission. The randomized response protocol applies: each client reports its true routing signal with probability p and a uniformly random alternative with probability 1-p. For ε-differential privacy (ε-DP) with ε=1, the reporting probability is $p = e^{\varepsilon}/(e^{\varepsilon}+1) \approx 0.73$ (calculated from the randomized response formula per the Duchi et al. framework [9]), providing strong individual privacy while maintaining aggregate statistical utility across millions of devices; individual randomization noise cancels in expectation at scale. The Duchi et al. framework

[9] provides the formal privacy accounting for this approach, and it integrates directly with the FedAvg pipeline in Section 6. The practical result is a layered privacy architecture in which content never leaves the device (the fundamental TEE guarantee), metadata about query processing is randomized before leaving the device (local DP), and aggregated signals are processed by a server seeing only statistical summaries (FedAvg with secure aggregation). Each layer provides an independent guarantee, and the guarantees compose under standard differential privacy composition theorems [15].

### 5.3 In-TEE AI Judge Architecture

Quality proxy signals require a quality evaluation step running inside the TEE and emitting only numeric scores. Figure 3 illustrates the in-TEE AI Judge architecture. Zheng et al. [11] demonstrated that LLM-as-Judge evaluation using strong models achieves over 80% agreement with human evaluators on quality scoring benchmarks. Drawing on this finding, a fine-tuned scoring-focused model, smaller than a generation-scale LLM, for example, in the 3B-parameter range, is proposed for the TEE adaptation: a model focused specifically on evaluation rather than generation can achieve high scoring accuracy with substantially lower compute than the primary inference model. The judge evaluates each primary model response on four dimensions: relevance (0–5), coherence (0–5), safety (binary pass/fail), and complexity (1–3), and emits only these four values alongside the response text. The judge's internal reasoning, the original query, and all intermediate computations remain inside the enclave. The judge model is trained on public benchmarks (MT-Bench, AlpacaEval) without any private user data, making its outputs non-sensitive beyond the standard DP protection applied to all signals in the Learn phase.

**Figure 3: In-TEE AI Judge Architecture**


Recreated from the client-provided diagram. The AI Judge and Primary LLM both operate within the TEE enclave boundary. Only four numeric quality scores exist in the enclave: no query content, no judge reasoning, and no model uncertainty estimates. The 3B-parameter judge size is an illustrative proposal; actual sizing depends on quality requirements and hardware constraints.

The four numeric scores exiting the TEE pair with the client-observable routing signals to form a complete quality-routing observation: tier selected, processing latency, and quality assessment. This observation is the primary training signal for the federated router improvement pipeline in Section 6. The judge's binary safety flag serves a secondary function: a safety failure triggers a client-side alert and a cascade signal indicating that the routed tier produced an unsafe response, enabling router threshold adjustment for similar query patterns.

## 6. Federated Learning for Continuous Router Improvement

### 6.1 On-Device Router Training

McMahan et al. [13] introduced Federated Averaging (FedAvg) at AISTATS 2017 as a communication-efficient protocol for training from decentralized data: each client device trains a local model update using its own data, transmits only the gradient (not the data), and the server aggregates gradients to produce a refined global model. For the reasoning allocation context, FedAvg enables router improvement using three signal sources entirely available on the client: routing decisions paired with engagement outcomes (whether the routed response was accepted, rejected, or edited), escalation events paired with quality score deltas (whether escalating to a higher tier improved the judge score), and judge scores paired with tier selection (whether the selected tier consistently produced adequate quality for query patterns at a given complexity level). All three training signals are content-free; they describe what happened to the routing decision and the response, not what either contained.

The router training pipeline operates asynchronously. Client devices accumulate signal observations across sessions, compute local gradient updates during low-activity periods, and contribute DP-protected gradients to the fleet aggregation server. The updated global router is pushed periodically to all devices. A critical operational property of FedAvg in this context is that the trained router generalizes to new users from their first session: a router refined on aggregate complexity preference patterns from millions of devices accurately classifies query complexity for a new user without that user having contributed any signal. Public Chatbot Arena data provides initial calibration; FedAvg refinement continuously improves it against the fleet's actual query distribution, maintaining accuracy as the fleet's usage patterns evolve.

### 6.2 Secure Aggregation

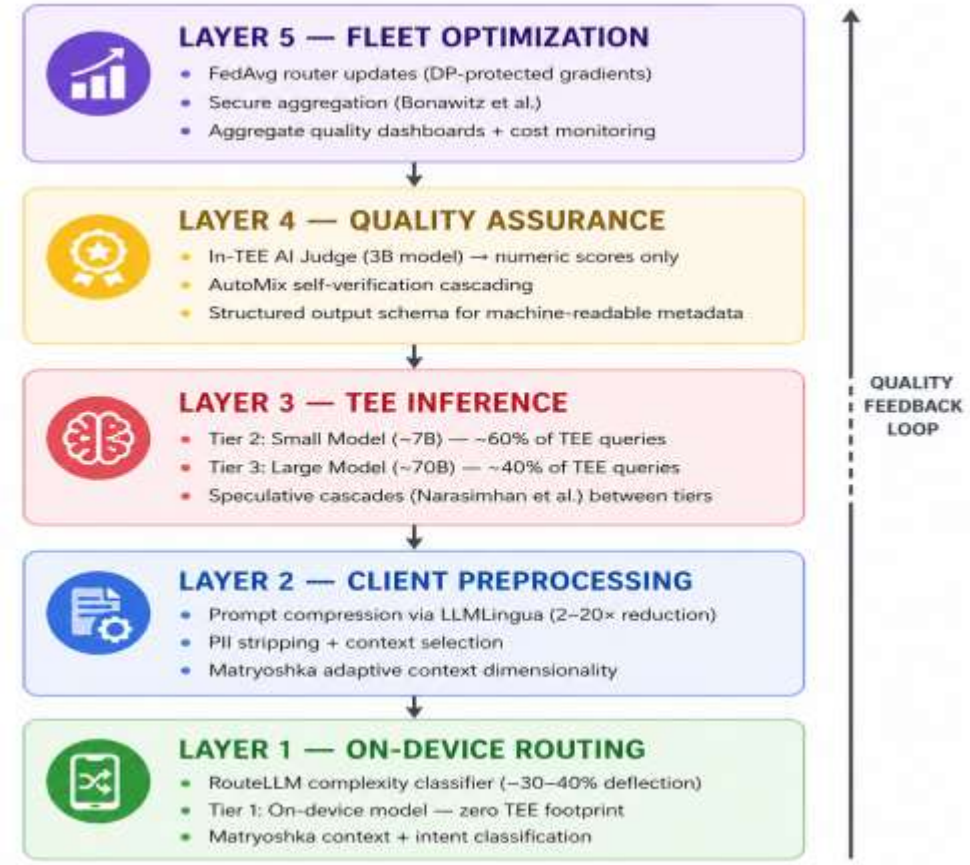
Bonawitz et al. [14] introduced a practical secure aggregation protocol at ACM CCS 2017, ensuring the aggregation server sees only the sum of client gradients, not individual contributions. The protocol uses secret sharing and masking so that even a fully compromised aggregation server cannot extract any individual device's gradient. Combined with local DP applied before gradient computation, secure aggregation ensures that the federated learning pipeline provides formal privacy guarantees at every layer: routing signals are DP-noised (local DP), individual gradients are cryptographically hidden (secure aggregation), and the aggregated update is a differentially private statistical summary of the fleet's routing behavior.

TEE-assisted aggregation provides an additional hardening option: placing the gradient aggregation step itself inside an attested enclave. In this configuration, contributing device gradients are combined inside a CVM rather than on a standard server, providing hardware-enforced isolation for the aggregation computation. The combination of local DP, secure aggregation, and TEE-based aggregation creates a three-layer privacy guarantee for the fleet-wide optimization pipeline that mirrors the three-layer privacy guarantee of the inference pipeline: content isolation in the TEE, transport anonymization via OHTTP [12], and verifiable transparency through published attestation artifacts.

## 7. The Five-Layer Optimization Stack

The individual research streams surveyed in Sections 3–6 integrate into a five-layer architecture for privacy-preserving TEE optimization, illustrated in Figure 4. Layer 1 (On-Device Routing) consists of the complexity classifier, a RouteLLM-class router [6] pre-trained on public preference data and continuously refined via FedAvg, that evaluates each incoming query before the TEE is contacted. Drawing on RouteLLM cost-reduction ratios [6] as a conservative basis, a meaningful fraction of queries can be deflected to on-device models, generating zero TEE footprint and the richest client-observable signal stream; the specific deflection rate is a deployment parameter requiring calibration against the actual query distribution. Layer 2 (Client Preprocessing) applies LLMLingua compression [7] and Matryoshka context truncation [8] to queries routed onward to the TEE, minimizing the payload entering the enclave. Layer 3 (TEE Inference) handles the remaining queries across two sub-tiers, a smaller model for lighter queries and a larger model for complex queries, with tier sizes (illustratively, approximately 7B and 70B parameters) representing deployment parameters that teams should calibrate based on hardware constraints and quality requirements, with speculative cascades [5] and AutoMix self-verification [4] governing escalation. Layer 4 (Quality Assurance) runs the in-TEE AI Judge [11] and AutoMix verification, emitting numeric scores. Layer 5 (Fleet Optimization) operates entirely outside the TEE, aggregating DP-protected routing signals and quality scores from client devices via FedAvg [13] with secure aggregation [14] to refine the Layer 1 router.

**Figure 4: Five-Layer Reasoning Allocation Optimization Stack**




Recreated from the client-provided diagram. Layers 1 and 2 operate on the client device; Layer 3 operates inside the TEE; Layer 4 spans the TEE boundary (computation inside, scores outside); Layer 5 operates at the fleet aggregation level with secure aggregation and differential privacy protection. Model tier sizes and intra-TEE query splits shown are illustrative parameters requiring deployment-specific calibration.

The five layers interact as a closed feedback loop. Layer 1 routing decisions determine which queries reach Layers 2–4. Layer 4 quality scores are the primary training signal for Layer 5. Layer 5 updates improve Layer 1 accuracy, which progressively reduces the fraction of queries unnecessarily escalated through Layers 2–4. The projected steady-state compute reduction of approximately 55%, under the modeling assumptions stated in Table 2, assumes mature Layer 1 routing calibrated to the fleet's actual query distribution, a state reachable through several FedAvg rounds on a fleet of sufficient scale.

The five-layer stack is not tied to any specific TEE hardware, model family, or deployment context. Any stateless, non-targetable TEE-based LLM deployment can adopt the same architectural pattern, substituting the appropriate hardware (AMD SEV-SNP, Intel TDX, NVIDIA Confidential Compute), model family, and domain-appropriate routing thresholds. The architectural principle, minimize TEE exposure through pre-TEE routing and preprocessing; maximize privacy-safe signal collection at the client layer; and improve routing via federated learning, is hardware-agnostic, model-agnostic, and domain-agnostic.

## 8. Discussion: Implications and Open Problems

Three open problems require resolution before the five-layer stack can be considered fully specified for production deployment. The router cold-start problem arises at initial deployment before any FedAvg rounds have been completed. The pre-trained RouteLLM router provides reasonable calibration for general conversational AI but may not accurately reflect the query distribution of a domain-specific deployment. A healthcare assistant's query distribution differs materially from a general chat assistant's; medical queries tend toward longer, more information-dense content that may systematically score above the escalation threshold even when a smaller model is fully adequate. Cold-start mitigation requires either a domain-specific public

evaluation set for pre-deployment router calibration or a conservative initial threshold that accepts higher compute cost in exchange for quality assurance until sufficient FedAvg rounds have refined the router. A concrete mitigation for the cold-start period is to deploy a conservative two-tier system initially, routing all queries either to the on-device model or to the large TEE model, with no small TEE intermediate tier, and activating the full three-tier routing only after the first FedAvg round has produced a calibrated router. This approach accepts a higher compute cost during the cold-start window in exchange for quality assurance, and it avoids the risk of the uncalibrated router sending complex queries to the small TEE model during the period when the router's accuracy is lowest.

The adversarial routing problem concerns users crafting queries to manipulate tier selection. A user seeking to consume maximum infrastructure resources could craft queries that consistently trigger escalation to the large TEE model. Conversely, a user seeking to avoid TEE processing entirely could craft queries that consistently score below the on-device threshold, which is particularly concerning in safety-sensitive deployments where on-device model safety guardrails may be weaker than the large TEE model's. Both attack patterns are detectable through fleet-level anomaly monitoring of per-user routing distributions (available as a DP-protected aggregate signal) and mitigable through routing diversity requirements that prevent any user from achieving a sustained single-tier distribution. The binary safety flag from the in-TEE AI Judge provides an additional safeguard: any safety failure overrides the router's tier selection and triggers escalation regardless of the complexity score.

The quality floor guarantee problem concerns the consequences of routing safety-critical queries to weaker models. For general conversational tasks, the quality degradation from on-device deflection is acceptable; the user receives a fast, adequate response. For queries in medical, legal, or crisis-response contexts, routing to a weaker model may produce qualitatively inadequate responses with material consequences. The in-TEE judge's safety flag addresses this partially: any flagged response triggers escalation. More comprehensive quality floor guarantees require domain-specific judge calibration, per-query-category routing policy, and potentially unconditional routing of specific query categories to the large TEE model, a per-deployment policy decision that must be specified at CVM build time to remain compatible with the verifiable transparency requirements of the attestation model.

## 9. Conclusion

The optimization paradox of TEE-based AI, that the architecture preventing data collection is the same architecture that needs data to improve, is not a fundamental constraint but a framing problem. Reasoning allocation reframes the problem correctly: the question is not how to collect optimization data inside the TEE, but how to generate optimization data outside it. Every routing decision made before the TEE boundary is an event observable on the client device, generating timing, confidence, and quality signals that describe system performance without describing user content. The TEE's stateless, non-targetable, anonymous properties are irrelevant to signals collected before the enclave is contacted, which is precisely why the client-observable signal stream in Table 4 is rich enough to support continuous router improvement.

The five-layer optimization stack synthesizes five verified research cascading, RouteLLM on-device routing, LLMLingua prompt compression, in-TEE LLM-as-Judge evaluation, and federated learning with secure aggregation into a deployable architecture projecting approximately 55% compute reduction under the stated modeling assumptions at mature calibration. Apple's Private Cloud Compute demonstrates that on-device-first reasoning allocation is achievable at consumer scale within the same architectural constraints that define WhatsApp's Private Processing. The open problems, cold-start calibration, adversarial routing resistance, and domain-specific quality floor guarantees define a concrete research agenda for extending the framework to higher-stakes deployment contexts.

The broader implication for confidential AI system design is that privacy and optimization are not structurally opposed. Optimizing a TEE-based LLM does not require weakening its privacy guarantees; it requires designing the optimization strategy to operate on signals that the privacy architecture naturally permits to be observed. Reasoning allocation does exactly that, and its status as a production feature in Apple's PCC confirms that the approach is not theoretical. The architectural principle generalizes: any component of a confidential AI system that generates observable client-side signals without content exposure is a candidate optimization surface, and designing for that surface from the outset is preferable to retrofitting observability into a system built without it.

## References

[1] Meta Engineering, "Building private processing for AI tools on WhatsApp," Meta Engineering Blog, Apr. 2025. [Online]. Available: https://engineering.fb.com/2025/04/29/security/whatsapp-private-processing-ai-tools/

[2] Apple Security Research, "Private Cloud Compute: A new frontier for AI privacy in the cloud," Apple Security Research Blog, 2024. [Online]. Available: https://security.apple.com/blog/private-cloud-compute/

[3] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models while reducing cost and improving performance," arXiv:2305.05176, May 2023. [Online]. Available: https://arxiv.org/abs/2305.05176

[4] P. Aggarwal et al., "AutoMix: Automatically mixing language models," arXiv:2310.12963, Oct. 2023. [Online]. Available: https://arxiv.org/abs/2310.12963

[5] H. Narasimhan et al., "Faster cascades via speculative decoding," in Proc. International Conference on Learning Representations (ICLR), 2025. [Online]. Available: https://openreview.net/forum?id=vo9t20wsmd

[6] I. Ong et al., "RouteLLM: Learning to route LLMs with preference data," arXiv:2406.18665, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2406.18665

[7] H. Jiang et al., "LLMLingua: Compressing prompts for accelerated inference of large language models," in Proc. 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 13358–13376, 2023. [Online]. Available: https://aclanthology.org/2023.emnlp-main.825/

[8] A. Kusupati et al., "Matryoshka representation learning," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/c32319f4868da7613d78af9993100e42-Abstract-Conference.html

[9] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in Proc. 54th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 429–438, 2013. [Online]. Available: https://ieeexplore.ieee.org/document/6686179

[10] J. Zhu, H. Yin, P. Deng, and S. Zhou, "Confidential computing on NVIDIA Hopper GPUs: A performance benchmark study," arXiv:2409.03992, Sep. 2024. [Online]. Available: https://arxiv.org/abs/2409.03992

[11] L. Zheng et al., "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023. [Online]. Available: https://arxiv.org/abs/2306.05685

[12] M. Thomson and C. A. Wood, "Oblivious HTTP," RFC 9458, Internet Engineering Task Force, Jan. 2024. [Online]. Available: https://www.ietf.org/rfc/rfc9458.html

[13] H. B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR 54, pp. 1273–1282, 2017. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[14] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1175–1191, 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3133956.3133982

[15] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," Foundations and Trends in Theoretical Computer Science, vol. 9, no. 3–4, pp. 211–407, 2014. [Online]. Available: https://doi.org/10.1561/0400000042