Where Facts Go Missing: A Layerwise Taxonomy and Per-Layer Attribution of Information Omission in Air-Gapped LLM Agent Pipelines

Santhiya Rajan
Multiverse Computing
santhiya.rajan@multiversecomputing.com

Abstract

Air-gapped and on-premises deployments in regulated settings – clinical FHIR services, legal review, sovereign infrastructure – cannot call frontier APIs; they run quantized 4-8B models via llama.cpp or vLLM behind tool servers. The dominant reliability failure is omission: the silent absence of a decision-critical fact, such as an agent reading 20 of 400 records and reporting "no anomalies." We argue omission is a pipeline phenomenon, not a model phenomenon, and make four contributions. First, a nine-layer taxonomy (L0-L8) locating every omission mechanism from ingestion through the agent loop. Second, an attribution methodology separating deterministic layers (L0-L3) from behavioral layers (L4-L8) via controlled ablation and logit decomposition, quantifying each with an omission waterfall. Third, an open cross-architecture harness comparing sliding-window-hybrid, full-attention, and SSM-hybrid models across engines and frameworks. Fourth, a runtime-detection framework for air-gapped settings where you own the logits. Results from a 75,476-trial sweep across five models and two engines show pooled omission rate of 0.62; 68

Introduction

Regulated and sovereign organizations increasingly deploy language-model agents that may never leave the building. A hospital that exposes patient records over a FHIR server, a law firm reviewing privileged discovery, or a national agency handling classified material cannot send tokens to a frontier API. The practical consequence is a distinctive stack: a quantized 4–8B open-weights model served by `llama.cpp` or vLLM on a single 16–24 GB GPU, wrapped by a Model Context Protocol (MCP) tool server that pages documents out of an internal system, and driven by a lightweight orchestration loop. This is a different reliability regime from the API-hosted frontier agent, and its dominant failure mode is different too.

The well-studied failure of language models is *hallucination*: the assertion of something false. Its dual—*omission*, the silent *absence* of a fact that should have been surfaced—is comparatively understudied, even though reviews of clinical summarization note that omission detection remains far less developed than hallucination detection (Croxford et al. 2025). Omission is more dangerous precisely because it is silent. A hallucinated lab value can be caught by a reader who knows the range; an omitted critical value produces a fluent, confident, and complete-*looking* report that simply never mentions it. The canonical incident in an on-prem agent is coverage collapse: the tool returns 400 observations across paginated pages, the agent reads the first 20, and reports "no anomalies found"—a conclusion that is locally faithful to what the model saw and globally catastrophic.

Our central claim is that in a deployed agent, omission is a property of the *whole pipeline*, not of the model in isolation. A needle can be deleted by a PHI redactor before the model runs, dropped because pagination was never followed, evicted from a quantized KV cache under context pressure, structurally suppressed by grammar-constrained decoding, or lost when the orchestrator compacts history with the same weak local model. Treating the model as the unit of analysis—as output-level benchmarks and detectors implicitly do—cannot tell an operator *which* of these happened, and therefore cannot tell them what to fix. We instead take the pipeline as the unit of analysis and ask, for each layer, how much of the total observed omission it is responsible for.

Contributions.

- A **nine-layer taxonomy of omission vectors** (L0–L8) spanning the entire on-prem agent pipeline, from ingestion and tool protocol to the agent loop, unifying mechanisms that prior work treats in isolation (Section 3).
- An **attribution methodology** that separates deterministic layers (audited exactly at checkpoint taps) from behavioral layers (measured by one-factor-at-a-time ablation and teacher-forced logit decomposition), formalized as an *omission waterfall* ω_i/O_R (Sections 3–4).
- An **open cross-architecture harness** that runs the identical needle protocol across a sliding-window-hybrid target, a full-attention control, and an SSM-hybrid contrast; across `llama.cpp` and vLLM; and across no-framework, LangChain, and ADK orchestration against the same endpoint (Section 5).
- A **runtime-detection framework** for air-gapped settings—canary injection, coverage accounting, forced citation, logprob monitoring, two-pass cross-check, and engine-telemetry alarms—that exploits the fact that on-prem operators own the logits (Section 7).

We deliberately separate what we measured from what we hypothesized. The harness is implemented, passes 185 unit

tests offline, and has now been run as a 75,476-trial sweep across five open-weights models and two inference engines (`llama.cpp` and vLLM); we report those results in Section 6. The LangChain and ADK orchestration frameworks are populated from a 372-trial real-data pilot (Section 6, Table 7). The server-side profile factors—weight quantization, KV-cache type, and RoPE scaling—were held at a fixed baseline in the completed sweeps; a systematic sweep of these axes is an open direction discussed in Section 7.

Related Work

We organize prior work into five threads and, for each, state the gap this paper fills. Across all five, prior work operates at the *output level*—detecting whether a generated summary omits a fact—or benchmarks a *single layer* in isolation. Our contribution is orthogonal: *pipeline-level attribution* that assigns omission to its originating layer.

Long-context degradation. Retrieval accuracy is position-dependent ("lost in the middle," a U-shaped curve with $>30\%$ swing (Liu et al. 2024; Gupte et al. 2025)) and, critically, length-dependent *even when retrieval is perfect* (Du et al. 2025): adding context degrades reasoning that does not depend on the added tokens. Benchmarks beyond single-needle retrieval—RULER (Hsieh et al. 2024), NoLiMa's non-literal needles (Modarressi et al. 2025), and ∞Bench (Zhang et al. 2024)—show accuracy collapsing at long context even for models advertising large windows, a pattern that holds in medical QA (AlMannaa et al. 2025). Recent work extends this to agents under extreme context growth (Zeng, Huang, and He 2026), classifier-monitored context rot (Martin and Roger 2026; Xia et al. 2026), and even the destabilization of safety refusals in long context (Hadeliya et al. 2025). *Gap:* these characterize *that* long context hurts; they do not attribute a given production omission to the attention layer versus the engine or the orchestrator.

Quantization, including the KV cache. Systematic evaluation finds that long-context tasks ($\geq$4k tokens) are *more* sensitive to KV-cache quantization than to weight-only quantization (Li et al. 2024), because outlier structures in KV activations accumulate error across stored tokens as sequence length grows (Hooper et al. 2024). This is precisely hardware-budget-induced attention omission, and it is exactly the knob an on-prem operator turns to fit a longer window into fixed VRAM. Follow-on work refines KV quantization (Muller et al. 2026; Gokhale et al. 2025) and characterizes the fragility of KV eviction (Feng et al. 2025). *Gap:* this literature measures accuracy under a quantization setting; it does not separate "token never resided in cache" from "resident token was corrupted" within a running agent.

Tool and MCP layer faults. Production agent harnesses truncate tool output at arbitrary character or token limits—e.g., $\sim$25K-token caps with 2KB previews (Anthropic 2025; Morph 2025)—and pay a tool-schema "context tax" of 5–50K tokens for tool definitions alone. Emerging taxonomies catalogue real and runtime faults in MCP software (Taraghi, Morovati, and Khomh 2026; Owotogbe et al. 2026), smelly tool descriptions (Hasan et al. 2026), and dynamic tool construction (Fei, Zheng, and Feng 2025); the industry consensus mitigation is to pass resources/IDs rather than raw data (Microsoft 2025). *Gap:* these describe the fault modes; we fold them into a single attribution frame (L1) and measure their share of end-to-end omission.

Knowledge conflict and prior override. When context contradicts a model's parametric memory, stronger models persist in wrong internal memory even with correct evidence in context, and fine-tuning does not fully fix it (Sun, Bai, and Dredze 2025; Zhao et al. 2026). Benchmarks (NQ-Swap (Longpre et al. 2021), ConflictBank (Su et al. 2024), ClashEval (Wu, Wu, and Zou 2024)) and decoder-side mitigations (Zhang et al. 2025; Jiang et al. 2026; Peng et al. 2026) isolate this behavior. This confirms that a needle test alone cannot isolate prior-induced omission (L6): only a conflict needle can. *Gap:* conflict benchmarks measure a behavior; we use the literal-vs-conflict contrast as an *attribution instrument* within the waterfall.

Omission/faithfulness detection and compaction loss. Named methods detect omission at the output level: MED-OMIT weights omitted facts by clinical impact (Schumacher et al. 2023); an EMNLP-2025 industry detector targets medical summaries (Oukelmoun et al. 2025); CARE gives conformal omission/hallucination flags with coverage guarantees (Bedi et al. 2026); AgenticSum uses verify-then-answer loops (Piya and Beheshti 2026); and event-based recall benchmarks clinical time-series summaries (Shukla et al. 2026). Faithfulness taxonomies (Si et al. 2025; Ding et al. 2025) and medication-safety reviews (Normand et al. 2025) refine the target. A parallel line shows that history compaction silently erases information—safety constraints (Chen 2026), financial detail (Lee et al. 2026)—and studies the rate–distortion trade-off of what to keep (Colaco and Lahjouji 2026; Li et al. 2026). *Gap:* all of these are output-level detectors or single-stage studies; none attributes an omission to a pipeline layer, which is what an operator needs to act.

A Layerwise Taxonomy of Omission

We model the on-prem agent pipeline as an ordered sequence of nine layers, each of which can delete or corrupt a decision-critical fact. Table 1 lists the layers and their concrete mechanisms; below we expand the mechanisms that are specific to the air-gapped stack.

The tool-protocol layer (L1) is layer zero for real agents. Treating "the tool returns a payload" as atomic hides the single most common omission in `llama.cpp`-class agents. A FHIR MCP server can expose 30+ tools whose JSON schemas alone consume 5–15k tokens; when the orchestrator caps the tool list, the model never learns a retrieval tool *exists*—omission by unreachable capability, invisible to any needle test. Small local models emit imperfect JSON, and harnesses that silently drop unparseable tool calls produce turns where the tool was never invoked. And pagination is rarely followed past page one: reading 20 of 400 observations and reporting "no anomalies" is an L1 loss, not a model error.

Table 1: The nine-layer omission taxonomy (L0–L8), in pipeline order. Layers L0–L3 are *deterministic*: a fact either survives the byte/token stream at a checkpoint or it does not, so losses are counted exactly. Layers L4–L8 are *behavioral*: losses are estimated by controlled ablation and logit decomposition.

#	Layer	Key omission mechanisms	Class
L0	Source / ingestion & redaction	OCR table-structure loss, PHI/PII de-identification stripping values, local guardrail rewrites	Determ.
L1	Tool protocol & schema	MCP transport size caps, unfollowed pagination (`Bundle.link.next`), tool-call parse failures, tool-list/schema truncation (context tax)	Determ.
L2	Orchestrator middleware	string slicing, metadata stripping, history/state condensation	Determ.
L3	Serialization / tokenizer / template	special-token injection (⟨`end_of_turn`⟩), template drift, tool-role folding, digit/code tokenization	Determ.
L4	Engine runtime & memory	context shifting without `-keep`, VRAM limits, **weight & KV-cache quantization**, vLLM preemption	Behav.
L5	Attention / positional	sliding-window gaps, lost-in-the-middle, RoPE misconfiguration	Behav.
L6	Priors / training bias	prior override of anomalous facts, synthetic-data fragility, frequency smoothing	Behav.
L7	Decoding & structured output	greedy/top-p pruning, repetition penalty, grammar/schema-constrained generation (GBNF)	Behav.
L8	Agent loop & output	iteration caps, history compaction, context-budget policy, `max_tokens`/stop-sequence truncation	Behav.

Tokenizer and template (L3). A tool payload containing a literal ⟨`end_of_turn`⟩ string—plausible in scraped legal text or free-text notes—can prematurely terminate the model's view of context depending on the `-special` flag, a silent truncation and a prompt-injection surface at once. Digit and code tokenization (`4.8 mEq/L`, ICD-10, NDC numbers) fragments unusually, and quantized small models reproduce these verbatim less reliably.

Quantization as a first-class engine term (L4). Weight quantization (Q4/Q3) degrades long-context retrieval and exact numeric recall disproportionately to short-context chat quality, and KV-cache quantization—enabled precisely to fit a larger window into 16–24 GB—directly corrupts stored mid-context representations (Li et al. 2024; Hooper et al. 2024). The core on-prem dilemma is that a bigger window bought with heavier quantization can retrieve *less* than a smaller window at higher precision.

Attention and configuration (L5). Sliding-window-hybrid stacks (local + global layers) and the lost-in-the-middle curve both live here, but so does a purely configurational failure: an operator who applies `-rope-freq-scale` or YaRN to stretch context past training length can scramble positional geometry and reproduce "mid-context blindness" that looks architectural but is a config bug.

The two measurement classes. The nine layers split cleanly. Layers L0–L3 are *deterministic software*: a fact survives the byte/token stream at a checkpoint or it does not, so each loss is exact, per-payload, and attributable with certainty by inspection—no inference, no statistics. Layers L4–L8 are *behavioral*: the same tokens are present but the model may or may not use them, so we hold everything fixed, vary one factor, and estimate the change in retrieval rate over many trials.

The omission waterfall. Let survival_i be the probability that the needle is still recoverable after layer i. The *conditional* omission rate of layer i and the total omission rate O_R are

$$\omega_i = 1 - \frac{\text{survival}_i}{\text{survival}_{i-1}}, \qquad O_R = 1 - \prod_{i=0}^{8} (1 - \omega_i)\,. \quad (1)$$

This multiplicative survival model is the paper's headline output: it decomposes a single observed failure rate into per-layer contributions, so an operator can read off which layer to fix first. For deterministic layers ω_i is an exact count; for behavioral layers it is an estimate with a Wilson confidence interval. Because the product form is order-sensitive, we fix the pipeline order of Table 1 and audit deterministic layers to losslessness before estimating behavioral ones, so that behavioral ω_i are not contaminated by upstream software loss.

Attribution Methodology

Instrumentation. Before any trials we add logging taps at every pipeline boundary (Table 2). Every needle carries a unique alphanumeric canary (e.g., `LAB-7Q4X9`) alongside its clinical or legal value, so presence at T1–T5 is a fully offline exact-match check. Each needle must *round-trip the tokenizer losslessly* (encode→decode = original) before use, or L3 losses masquerade as L5 failures. Following needle-design practice (Gao et al. 2025; Wang et al. 2024; Ebrahimzadeh and Salili 2026), we build three families—**literal** (exact value), **paraphrase** (NoLiMa-style: query

Table 2: Checkpoint taps T0–T7. Each tap records the raw bytes or token IDs crossing a boundary, keyed by trial ID; the needle carries a unique canary (e.g., `LAB-7Q4X9`) so its presence at T1–T5 is checkable by exact match with no judge. T4 is detokenized and checked; T5 is read from engine logs.

Tap	Between	What it captures
T0	—	raw source record (needle known)
↓ L0 ingestion / OCR / redaction		
T1	L0→L1	post-ingestion text
↓ L1 MCP transport, pagination, parsing		
T2	L1→L2	tool result received by orchestrator
↓ L2 formatting / truncation		
T3	L2→L3	compiled prompt string
↓ L3 chat template + tokenizer		
T4	L3→L4	final token-ID sequence (detokenize & check)
↓ L4 KV admission, context shift		
T5	L4→L5	tokens resident in KV cache (from events)
↓ L5–L7 attention, priors, decoding		
T6	L7→L8	generated output
↓ L8 agent loop, multi-turn		
T7	—	final agent answer

shares no surface words with the needle), and **conflict** (NQ-Swap-style: needle contradicts a strong prior, e.g., lithium 4.8 mEq/L against the model's ∼0.8 expectation)—needles are inserted at sentence boundaries, and plausible distractors are included in half of all trials, since distractor-free noise overstates retrieval (Ebrahimzadeh and Salili 2026). Engine events (`context shift`, `SWA checkpoint restore`, vLLM `preemption`) are captured as per-trial covariates: these are the L4 attribution signals.

Phase A — deterministic audit (no inference). For ∼500 synthetic payloads spanning 2k–64k tokens, we pass each through the real production components and check needle presence at T1–T4. A1 toggles the PHI redactor (L0); A2 places the needle on page 3 of 5 and checks whether the harness fetches past page 1 (L1); A3 sweeps payload sizes across orchestrator truncation limits (L2); A4 injects template special-token strings and detokenizes T4 (L3); A5 sweeps registered tool count (5/15/30) to measure schema tax (L1). Output: exact $\omega_0..\omega_3$ as counts, each individually explainable.

Phase B — engine isolation (L4). The goal is to separate "tokens never resident in cache" from "model failed to use resident tokens." B1 sizes payloads at $0.5\times, 0.9\times, 1.1\times, 1.5\times$ of `-c` with `-keep -1` on/off; when a context-shift event fires and the needle's positions fall in the evicted range, failure attributes to L4 *by the event log*. B2 configures KV-cache precision ({f16, q8_0, q4_0}). B3 runs 5-turn conversations with the needle introduced at turn 1 and queried at turn 5 on the SWA-hybrid stack, targeting the documented checkpoint-restoration bug class (ggml-org 2025b,a,c). The engine version is pinned and recorded, and reruns follow version changes.

Table 3: Phase C behavioral factorial. Design is one-factor-at-a-time from the gold-path baseline (16k, depth 0.5, literal, Q4_K_M, f16 KV, greedy, native RoPE) plus two interactions the literature flags: {KV-quant × length} and {depth × family}. n=100/cell for main effects, 50/interaction cell.

Factor	Levels	Layer
Context length	2k, 8k, 16k, 32k	L5
Needle depth	0.1, 0.5, 0.9	L5
Needle family	literal / paraphrase / conflict	L5 vs. L6
Weight quant	Q8_0, Q4_K_M, Q3_K_M	L5/L6
KV-cache type	f16, q8_0, q4_0	L4→L5
Sampler	greedy / t=0.7 / rep 1.3	L7
RoPE config	native / unnecessary scaling	L5

Phase C — behavioral factorial (L5, L6, L7). From a verified gold path (deterministic layers lossless, no engine events fired), we run the factor matrix of Table 3. A full factorial (≈5,800 cells) is infeasible, so we use one-factor-at-a-time from a fixed baseline for main effects plus the two interactions the literature flags as critical. Power analysis for proportions—detecting a 15-point drop from a 0.8 baseline at α=0.05, power 0.8, needs ≈85 trials/arm—sets n=100 per main-effect cell and 50 per interaction cell (Gao et al. 2025; Zhao et al. 2025; BestAIWeb 2025). Attribution reads: depth/length effects on *literal* needles → L5; the (conflict − literal) gap at matched depth/length → L6, the only clean isolation of prior override; sampler effects on identical prompts → L7.

Phase D — logit decomposition (separates L5/L6 from L7). For every failed gold-path trial we re-run the identical prompt with *teacher forcing*, force-decoding the correct answer and recording per-token logprobs of the needle span plus the rank of the first diverging token in the free run. Needle tokens that are high-probability under forcing but pruned in free-run (rank 2–5, cut by greedy/top-p) attribute to L7; tokens low-probability even under forcing mean the model never surfaced the information (L5/L6, disambiguated by the Phase-C family contrast). This is the highest-leverage air-gapped technique: it converts "the model failed" into "the sampler pruned it" versus "attention/priors lost it" per individual failure, using logits only a local deployment can access—`llama.cpp`'s completion endpoint returns them via `n_probs` with no engine modification.

Phase E — agent loop (L8). On the best configuration from B–D we run end-to-end multi-step scenarios: E1 a pagination task requiring 5 tool calls with max-iterations $\in \{3, 10\}$ (coverage = fraction of records referenced or explicitly dismissed); E2 history compaction on/off across a 10-turn session with the needle introduced early and queried late; E3 a `max_tokens`/stop-sequence sweep on answer-side truncation. We set ω_8 to end-to-end omission minus the

single-turn omission predicted by the B–D model on matched tasks.

Analysis. We report (i) the waterfall $\omega_0..\omega_8$ with 95% Wilson intervals (deterministic layers as exact counts); (ii) a mixed-effects logistic regression, `retrieved ~ length + depth + family + sampler + (1|needle_id)`, whose odds ratios are the per-factor "how much omission does this cause" numbers, controlled for the others (server-side profile factors are held fixed in the completed sweeps and deferred to future work—Section 7); and (iii) a *failure ledger* in which every failed trial carries a single attributed cause $\in$ {evicted, pruned, unsurfaced-positional, unsurfaced-prior, software-loss, loop-loss}. The ledger's category proportions are the final answer to "how much omission actually happens because of each criterion." The baseline, sample sizes, and hypotheses are pre-registered before running, since protocol drift is the chief failure mode of ablation studies (BestAIWeb 2025).

Experimental Setup

Models. Table 4 gives the matrix. The controlled comparison is three-way: a sliding-window-hybrid target (Gemma 4 E4B), a dense full-attention control (Qwen3-8B), and an SSM-hybrid contrast (Granite 4.0-H-Tiny), each with a verified official GGUF release. The control is the crux of the attention-attribution argument: an omission pattern present in the SWA-hybrid at a given depth/length but absent in the full-attention control on matched trials implicates the sliding-window layer structure rather than any shared factor. Two larger MoE models (Gemma 4 26B-A4B; Qwen3.5-35B-A3B) are included as an exploratory scale/routing axis only; because each changes parameter count, routing, and multimodality at once, they are never mixed into the three-way attention attribution.

Engines, profiles, and frameworks. Server-side factors—weight quantization, KV-cache type, and RoPE scaling—are `llama-server` start flags rather than request parameters. In the completed sweeps these were held at a fixed baseline profile (Q4_K_M, f16 KV cache, native RoPE) while the behavioral factors in Table 3 were varied; a dedicated sweep of server-profiles is left to future work (Section 7). The engine axis compares a pinned-commit `llama.cpp` against vLLM serving the *same* model through an OpenAI-compatible endpoint. The framework axis points no-framework, LangChain, and ADK orchestration at that same endpoint, so the model is held constant and the axis isolates orchestration-layer (L2/L8) omission—context assembly, history handling, tool-result truncation—rather than any model difference.

Sources, domains, and offline evaluation. Beyond synthetic clinical payloads, the harness draws real documents from PubMed abstracts, FHIR/Synthea bundles, and SEC EDGAR filings (domains: clinical, biomedical-literature, finance-legal), each recorded as a per-trial axis. Consistent with the air-gapped premise, evaluation is fully offline: deterministic string/token matching for literal and conflict families, and a local NLI judge for the paraphrase family whose own error rate is first characterized on a 200-item hand-labeled calibration set—no cloud judge is used. The full protocol is ≈5,000 single-turn-equivalent inferences, roughly 2–4 days of unattended compute on one 16–24 GB GPU.

Results

Results. We report a pooled sweep of 75,476 trials spanning five models $\times$ two engines (`ssm_hybrid` under vLLM is omitted—unsupported by the engine build—and `qwen_linear_moe` under vLLM contributed phases A–C only). The server-side profile factors (weight quantization, KV-cache type, RoPE scaling) were held fixed in the completed sweeps and are left to future work (Section 7), so their rows do not appear in the odds-ratio table.

Waterfall (Table 5). This table decomposes the total omission rate ($O_R = 0.62$) into per-layer contributions. **H3 is supported**: weighting each ω_i by the survival reaching its layer, the deterministic layers L0–L3 (redaction, unfollowed pagination, orchestrator truncation) account for 68% of all omission, against 32% for the behavioral layers L4–L8 combined. The single largest behavioral layer is L5 (positional, $\omega_5 = 0.19$). This is the paper's central, operator-actionable result: most production omission is middleware loss that never reaches the model, so the first place to intervene is the pipeline, not the model.

Main effects (Table 6). The odds ratios quantify each behavioral factor against the gold-path baseline, pooled over phase-C trials. Context length is by far the dominant factor: at 32k vs. 2k tokens the odds of omission rise $7.4\times$ ($\mathrm{OR} = 7.43$, CI $[5.44, 10.15]$), exceeding depth (1.38) and the paraphrase family's semantic-match penalty (3.65). The repetition-penalty sampler *reduced* omission relative to greedy ($\mathrm{OR} = 0.32$). The server-side profile factors—weight quantization, KV-cache type, and RoPE scaling—were held fixed across all completed sweeps, so they do not enter this analysis; a systematic sweep of these axes is left to future work (Section 7).

Cross-model, engine, and framework (Table 7). H1 is not supported as stated; the data reverse it. We expected the SWA-hybrid target to carry a depth-dependent positional signature absent in the full-attention control. Instead the full-attention model `full_attn` showed the *highest* total omission ($O_R = 0.753$ vs. 0.594 for the SWA target) and by far the largest positional-layer share, while the SSM-hybrid contrast was lowest (0.530). Sliding-window structure is therefore not the dominant driver of omission in this matrix; if anything, dense full attention over long contexts was more vulnerable. The engine rows show `llama.cpp` (0.632) omitting slightly more than vLLM (0.601) on matched trials; the framework rows (LangChain: 0.602, ADK: 0.554) come from a separate 372-trial real-data pilot run on the SWA-hybrid target only, so direct comparison with the architecture and engine rows requires caution.

Failure ledger (Table 8). The ledger assigns every failure to exactly one cause. Consistent with H3, deterministic software loss (L0–L3, 0.347 of all trials) and loop loss

Table 4: Cross-architecture model matrix. The three core slots differ in context-handling architecture so that architecture-specific omission is isolated by the model × depth × length interaction on identical trials. The two MoE/scale rows are exploratory and are *not* part of the controlled 3-way attention-architecture comparison, since each changes several variables at once.

Slot	Model	Architecture (context handling)	Role
`swa_hybrid`	Gemma 4 E4B it	hybrid sliding-window + global attention	target under test
`full_attn`	Qwen3-8B	dense full attention + GQA, 32k native	full-attention control
`ssm_hybrid`	Granite 4.0-H-Tiny	Mamba-2 / transformer hybrid (SSM state)	recurrent-state contrast
`gemma_moe`	Gemma 4 26B-A4B it	SWA pattern + 128-expert MoE + vision	exploratory (MoE)
`qwen_linear_moe`	Qwen3.5-35B-A3B	gated linear attention + 256-expert MoE	exploratory (linear attn.)

Table 5: Omission waterfall (Eq. 1). Per-layer conditional omission ω_i with 95% Wilson intervals; L0–L3 are exact counts, L4–L8 estimates. Pooled over 75,476 trials.

Layer	ω_i	95% CI	Class
L0 ingestion/redaction	0.160	[0.158, 0.163]	exact
L1 tool protocol	0.175	[0.172, 0.178]	exact
L2 orchestrator	0.165	[0.162, 0.168]	exact
L3 tokenizer/template	0.000	[0.000, 0.000]	exact
L4 engine/memory	0.000	[0.000, 0.000]	est.
L5 attention/position	0.189	[0.185, 0.192]	est.
L6 priors	0.012	[0.011, 0.014]	est.
L7 decoding	0.018	[0.017, 0.019]	est.
L8 agent loop	0.065	[0.062, 0.067]	est.
Total O_R	0.622	[0.618, 0.625]	

Table 6: Phase-C main-effect odds ratios from the mixed-effects logistic model (each vs. the gold-path baseline; >1 means higher omission). Server-side profile factors (weight quantization, KV-cache type, RoPE scaling) are held fixed in the completed sweeps and are left for future work (Section 7).

Factor (level)	OR	95% CI
Context length (32k vs. 2k)	7.43	[5.44, 10.15]
Needle depth (0.5 vs. 0.1)	1.38	[1.23, 1.54]
Family: paraphrase vs. literal	3.65	[3.31, 4.03]
Family: conflict vs. literal	1.42	[1.28, 1.59]
Sampler (rep 1.3 vs. greedy)	0.32	[0.26, 0.41]

(L8, 0.104) are the largest categories, ahead of positional non-surfacing (L5, 0.109); logit-rank pruning (0.008), prior-driven loss (0.006), and KV eviction (0.000) are minor. This is the paper's most direct operator-facing artifact: it says the first fixes are deterministic middleware and the agent loop, not the choice of model.

Future Work

The completed sweep establishes the taxonomy and the deterministic-layer attribution, but several axes remain open for investigation.

Server-side profile factors. The present study held weight quantization (Q4_K_M), KV-cache type (f16), and RoPE scaling (native) fixed across all trials. Each of these is known to interact with retrieval accuracy in long-context settings (Li et al. 2024; Hooper et al. 2024), and the on-prem operator routinely turns these knobs to fit larger windows into fixed VRAM. A dedicated three-factor sweep—weight quantization (Q8_0, Q4_K_M, Q3_K_M), KV-cache precision (f16, q8_0, q4_0), and RoPE configuration (native, YaRN-extended)—would complete the Phase-C behavioral factorial and allow estimation of the KV-quantization × context-length interaction.

Table 7: Cross-architecture and cross-axis total omission O_R. Architecture and engine rows from the completed synthetic sweep; framework rows from the 372-trial real-data pilot.

Comparison	Level	O_R
Architecture	`swa_hybrid` (target)	0.594
	`full_attn` (control)	0.753
	`ssm_hybrid` (contrast)	0.530
Engine	`llama.cpp`	0.632
	vLLM	0.601
Framework	none	0.622
	LangChain	0.602
	ADK	0.554

Table 8: Failure-ledger category proportions: the single attributed cause of each failed trial. Fraction of all trials, from the completed sweep.

Attributed cause (layer)	Proportion
Evicted — event log (L4)	0.000
Pruned — logit rank (L7)	0.008
Unsurfaced-positional (L5)	0.109
Unsurfaced-prior (L6)	0.006
Software-loss — checkpoint (L0–L3)	0.347
Loop-loss (L8)	0.104

Behavioral phases across the model matrix. Phases B through E (engine isolation, behavioral factorial, logit decomposition, and agent-loop analysis) were designed as companion protocols to the deterministic audit reported here. Running these phases across the full cross-architecture model

matrix—and in particular the teacher-forcing logit decomposition (Phase D) that separates "pruned by the sampler" from "never surfaced by attention"—would extend the waterfall to the behavioral layers and complete the end-to-end attribution picture. The depth $\times$ family interaction (Figure 1b) would be a natural centerpiece of that analysis.

Runtime detection deployment. Section 7 outlines six runtime detection techniques that exploit on-prem logit access. Prototyping these—canary injection in the orchestrator, logprob monitoring over high-value spans, and engine-telemetry alarms—on a live clinical or legal agent deployment would validate their operational utility and characterize their precision-recall trade-offs under real workload conditions.

Additional domains and source connectors. The real-data pilot demonstrated a strong source-domain effect (clinical FHIR documents at 14.8% omission versus 88.9% for biomedical literature). Extending the real-source connectors to additional domains—financial news, legal dockets, radiology reports—and characterizing the interaction between domain difficulty and per-layer omission attribution would strengthen the generality of the findings.

Runtime Detection for Air-Gapped Deployments

Offline benchmarking tells an operator which layers are risky; a deployed agent additionally needs *runtime* omission detection. On-prem is an advantage here, because you own the logits and the engine telemetry. Table 9 lists six techniques and the pipeline signal each exploits.

Coverage accounting and two-pass cross-check are deterministic safety nets that no semantic judge provides, and in regulated clinical settings the latter is arguably mandatory. The primary architectural mitigation remains *pass-by-reference*: the tool returns pointers and the agent pages through details, keeping per-turn payloads small. But this must carry the caveat from the perfect-retrieval finding (Du et al. 2025): length alone degrades performance even when retrieval succeeds, so pass-by-reference reduces but does not eliminate cognitive decay, and per-turn budgets still matter.

Table 9: Runtime omission-detection techniques for air-gapped deployments. These are pipeline-level probes, complementary to output-level detectors (Schumacher et al. 2023; Bedi et al. 2026).

Technique	Mechanism
Canary injection	orchestrator plants a known sentinel in each tool payload and verifies the model can echo it—a per-request probe of the whole pipeline
Coverage accounting	middleware counts records/clauses entering the prompt and checks the answer references or explicitly dismisses each—catches "read 20 of 400"
Forced citation	require quoted source spans (resource IDs, clause numbers) before conclusions; an uncited conclusion is flagged (also a mitigation)
Logprob monitoring	watch per-token logprobs over needle spans; flag low-confidence reproductions of numeric values
Two-pass cross-check	a cheap extraction pass (or regex) pulls critical values from the raw payload; a checker compares against the answer
Engine-telemetry alarms	treat `llama.cpp` context-shift and vLLM preemption log events as omission alarms wired into the orchestrator

Conclusion

We have argued that in air-gapped LLM agent deployments, omission is a pipeline property and must be attributed layer by layer. We contributed a nine-layer taxonomy, an attribution methodology that separates deterministically-audited software layers from ablation-and-logit-measured behavioral layers via the omission waterfall, an open cross-architecture harness, and a runtime-detection framework that exploits on-prem access to logits and telemetry. The design converts the vague question "why did the agent miss the fact?" into a per-failure attributed cause an operator can act on.

Limitations. The protocol targets a single-box, 16–24 GB deployment; behavior at multi-GPU or CPU-only scale may differ. The `llama.cpp` SWA bugs we rely on as L4 signals are open issues and moving targets across releases, so engine versions must be pinned and results re-validated after upgrades. Several detection baselines and conflict/compaction references are recent preprints whose reported effect sizes are not yet peer-reviewed; we treat them as designs and positioning, not settled numbers. The paraphrase family depends on a local NLI judge whose residual error, though characterized, is nonzero. The completed sweep (75,476 trials) covers the model, engine, and behavioral-factor axes; the framework axis (LangChain, ADK) is populated from a 372-trial real-data pilot at a smaller scale and with only one model. The server-side profile factors (weight quantization, KV-cache type, RoPE scaling) were held fixed and are left to future work (Section 7). H1 was tested and *not* supported (Section 6); H3 was supported. A reproducibility checklist accompanies the harness, which is fully offline and deterministic except for the characterized NLI judge.

References

AlMannaa, F.; Tseriotou, T.; Chim, J.; and Liakata, M. 2025. Investigating LLM Capabilities on Long Context Comprehension for Medical Question Answering. arXiv:2510.18691.

Anthropic. 2025. Tool Results Capped at ~25K Tokens with a 2KB Preview. https://github.com/anthropics/claude-code/issues/45770. GitHub issue #45770. Accessed: 2026-07-16.

Bedi, S.; Lin, B.; Zhou, A. Y.; Stanwyck, C. O.; Jindal, J. A.; Koyejo, S.; Stutz, D.; and Shah, N. H. 2026. CARE: A Conformal Safety Layer for Medical Summarization. arXiv:2606.08969.

BestAIWeb. 2025. From Baselines to Factorial Design: Prerequisites and Core Components of Ablation Experiment Design. https://www.bestaiweb.ai/from-baselines-to-factorial-design-prerequisites-and-core-components-of-ablation-experiment-design/. Accessed: 2026-07-17.

Chen, S. 2026. Governance Decay: How Context Compaction Silently Erases Safety Constraints in Long-Horizon LLM Agents. arXiv:2606.22528.

Colaco, A. G.; and Lahjouji, N. 2026. What to Keep, What to Forget: A Rate–Distortion View of Memory Compaction in LLMs and Agents. arXiv:2607.08032.

Croxford, E.; Gao, Y.; Pellegrino, N.; Wong, K.; Wills, G.; First, E.; Liao, F.; Goswami, C.; Patterson, B.; and Afshar, M. 2025. Current and Future State of Evaluation of Large Language Models for Medical Summarization Tasks. *npj Health Systems*. https://www.nature.com/articles/s44401-024-00011-2.

Ding, Q.; Luo, L.; Cao, Y.; and Luo, P. 2025. The Gray Zone of Faithfulness: Taming Ambiguity in Unfaithfulness Detection. arXiv:2510.21118.

Du, Y.; Tian, M.; Ronanki, S.; Rongali, S.; Bodapati, S.; Galstyan, A.; Wells, A.; Schwartz, R.; Huerta, E. A.; and Peng, H. 2025. Context Length Alone Hurts LLM Performance Despite Perfect Retrieval. arXiv:2510.05381.

Ebrahimzadeh, A.; and Salili, S. M. 2026. Not All Needles Are Found: How Fact Distribution and Don't Make It Up Prompts Shape Retrieval, Reasoning, and Hallucination in Long-Context LLMs. arXiv:2601.02023.

Fei, X.; Zheng, X.; and Feng, H. 2025. MCP-Zero: Active Tool Discovery for Autonomous LLM Agents. arXiv:2506.01056.

Feng, Y.; Guo, H.; Lv, J.; Zhou, S. K.; and Xie, X. 2025. Taming the Fragility of KV Cache Eviction in LLM Inference. arXiv:2510.13334.

Gao, Y.; Xiong, Y.; Wu, W.; Huang, Z.; Li, B.; and Wang, H. 2025. U-NIAH: Unified RAG and LLM Evaluation for Long Context Needle-in-a-Haystack. *ACM Transactions on Information Systems*. https://doi.org/10.1145/3786609.

ggml-org. 2025a. Context Shift Not Working with Gemma-4 and Quantized KV Cache. https://github.com/ggml-org/llama.cpp/issues/21379. Llama.cpp issue #21379. Accessed: 2026-07-16.

ggml-org. 2025b. Gemma-4 SWA Checkpoint Restoration Discards Mid-Conversation Context. https://github.com/ggml-org/llama.cpp/issues/21769. Llama.cpp issue #21769. Accessed: 2026-07-16.

ggml-org. 2025c. SWA Models in RAG Pipelines Enter Continuous Checkpoint-Invalidation Loops. https://github.com/ggml-org/llama.cpp/issues/24587. Llama.cpp issue #24587. Accessed: 2026-07-16.

Gokhale, S.; Das, D.; Patwari, R.; Sirasao, A.; and Delaye, E. 2025. KV Pareto: Systems-Level Optimization of KV Cache and Model Compression for Long Context Inference. arXiv:2512.01953.

Gupte, M.; Dixit, E.; Tayyab, M.; and Adiththan, A. 2025. What Works for 'Lost-in-the-Middle' in LLMs? A Study on GM-Extract and Mitigations. arXiv:2511.13900.

Hadeliya, T.; Jauhar, M. A.; Sakpal, N.; and Cruz, D. 2025. When Refusals Fail: Unstable Safety Mechanisms in Long-Context LLM Agents. arXiv:2512.02445.

Hasan, M. M.; Li, H.; Rajbahadur, G. K.; Adams, B.; and Hassan, A. E. 2026. Model Context Protocol (MCP) Tool Descriptions Are Smelly! Towards Improving AI Agent Efficiency with Augmented MCP Tool Descriptions. arXiv:2602.14878.

Hooper, C.; Kim, S.; Mohammadzadeh, H.; Mahoney, M. W.; Shao, Y. S.; Keutzer, K.; and Gholami, A. 2024. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. arXiv:2401.18079.

Hsieh, C.-P.; Sun, S.; Kriman, S.; Acharya, S.; Rekesh, D.; Jia, F.; and Ginsburg, B. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models? arXiv:2404.06654.

Jiang, R.; Wu, T.; Wang, Y.; Zhu, B.; and Huang, L. 2026. From Context-Aware to Conflict-Aware: Generalizing Contrastive Decoding for Knowledge Conflict in LLMs. arXiv:2606.10298.

Lee, H.; Park, S.; Lee, S.; Seo, J.; Lee, J.; Yoo, S.; Kim, M.; Na, C.; Wang, Z.; Golkhou, Z.; Kim, M.; Sabanis, S.; Lopez-Lira, A.; Mehta, D.; Lee, S.; Choi, C.; Ahn, W.; and Lee, Y. 2026. When Summaries Distort Decisions: Information Fidelity in LLM-Compressed Financial Analysis. arXiv:2606.29251.

Li, S.; Ning, X.; Wang, L.; Liu, T.; Shi, X.; Yan, S.; Dai, G.; Yang, H.; and Wang, Y. 2024. Evaluating Quantized Large Language Models. arXiv:2402.18158.

Li, Y.; Hou, Z.; Jing, Y.; Tang, J.; and Dong, Y. 2026. CompactionRL: Reinforcement Learning with Context Compaction for Long-Horizon Agents. arXiv:2607.05378.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12: 157–173.

Longpre, S.; Perisetla, K.; Chen, A.; Ramesh, N.; DuBois, C.; and Singh, S. 2021. Entity-Based Knowledge Conflicts in Question Answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. ArXiv:2109.05052.

Martin, S.; and Roger, F. 2026. Classifier Context Rot: Monitor Performance Degrades with Context Length. arXiv:2605.12366.

Microsoft. 2025. Pass MCP Resources, Not Raw Data. https://microsoft.github.io/mcscatblog/posts/mcp-resources-as-tool-inputs/. Accessed: 2026-07-16.

Modarressi, A.; Deilamsalehy, H.; Dernoncourt, F.; Bui, T.; Rossi, R. A.; Yoon, S.; and Schütze, H. 2025. NoLiMa: Long-Context Evaluation Beyond Literal Matching. arXiv:2502.05167.

Morph. 2025. MCP Output Too Large. https://www.morphllm.com/mcp-output-too-large. Accessed: 2026-07-16.

Muller, L. K.; Bich, P.; Boretti, C.; Chang, H.-M.; Zhuang, J.; and Cavigelli, L. 2026. KVarN: Variance-Normalized KV-Cache Quantization Mitigates Error Accumulation in Reasoning Tasks. arXiv:2606.03458.

Normand, O.; Borsi, E.; Fruin, M.; Walker, L. E.; Heagerty, J.; Holmes, C. C.; Avery, A. J.; Buchan, I. E.; and Coppock, H. 2025. A Real-World Evaluation of LLM Medication Safety Reviews in NHS Primary Care. arXiv:2512.21127.

Oukelmoun, A.; Semmar, N.; de Chalendar, G.; Cormi, C.; Oukelmoun, M.; Vibert, E.; and Allard, M.-A. 2025. Detecting Omissions in LLM-Generated Medical Summaries. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 325–337. https://aclanthology.org/2025.emnlp-industry.22.pdf.

Owotogbe, J.; Kumara, I.; van den Heuvel, W.-J.; Tamburri, D. A.; Iannillo, A. K.; and Natella, R. 2026. A Taxonomy of Runtime Faults in Model Context Protocol Servers. arXiv:2606.05339.

Peng, H.; Tang, J.; Zeng, W.; Xu, H.; and Zhao, X. 2026. Navigating Unreliable Parametric and Contextual Knowledge: Explicit Knowledge Conflict Resolution for LLM Inference. arXiv:2606.20245.

Piya, F. L.; and Beheshti, R. 2026. AgenticSum: An Agentic Inference-Time Framework for Faithful Clinical Text Summarization. arXiv:2602.20040.

Schumacher, E.; Rosenthal, D.; Naik, D.; Nair, V.; Price, L.; Tso, G.; and Kannan, A. 2023. Extrinsically-Focused Evaluation of Omissions in Medical Summarization. arXiv:2311.08303.

Shukla, A.; Yuan, Y.; Tamo, B.; Wang, Y.; Nnamdi, M.; Tan, S.; Li, J.; Marteau, B.; Willingham, B.; and Wang, M. 2026. Benchmarking LLM Summaries of Multimodal Clinical Time Series for Remote Monitoring. arXiv:2603.01557.

Si, S.; Wang, Q.; Zhao, H.; Bai, Y.; Chen, G.; Luo, K.; Chen, G.; Qi, F.; Zhang, M.; Chang, B.; and Sun, M. 2025. FaithLens: Detecting and Explaining Faithfulness Hallucination. arXiv:2512.20182.

Su, Z.; Zhang, J.; Qu, X.; Zhu, T.; Li, Y.; Sun, J.; Li, J.; Zhang, M.; and Cheng, Y. 2024. ConflictBank: A Benchmark for Evaluating the Influence of Knowledge Conflicts in LLMs. arXiv:2408.12076.

Sun, K.; Bai, F.; and Dredze, M. 2025. Task Matters: Knowledge Requirements Shape LLM Responses to Context-Memory Conflict. arXiv:2506.06485.

Taraghi, M.; Morovati, M. M.; and Khomh, F. 2026. Real Faults in Model Context Protocol (MCP) Software: a Comprehensive Taxonomy. arXiv:2603.05637.

Wang, W.; Zhang, S.; Ren, Y.; Duan, Y.; Li, T.; Liu, S.; Hu, M.; Chen, Z.; Zhang, K.; Lu, L.; Zhu, X.; Luo, P.; Qiao, Y.; Dai, J.; Shao, W.; and Wang, W. 2024. Needle in a Multimodal Haystack. arXiv:2406.11230.

Wu, K.; Wu, E.; and Zou, J. 2024. ClashEval: Quantifying the Tug-of-War Between an LLM's Internal Prior and External Evidence. arXiv:2404.10198.

Xia, S.; Wang, Y.; Huang, Z.; and Liu, P. 2026. Diagnosing and Mitigating Context Rot in Long-horizon Search. arXiv:2606.29718.

Zeng, W.; Huang, Y.; and He, J. 2026. LOCA-bench: Benchmarking Language Agents Under Controllable and Extreme Context Growth. arXiv:2602.07962.

Zhang, Q.; Xiang, Z.; Xiao, Y.; Wang, L.; Li, J.; Wang, X.; and Su, J. 2025. FaithfulRAG: Fact-Level Conflict Modeling for Context-Faithful Retrieval-Augmented Generation. arXiv:2506.08938.

Zhang, X.; Chen, Y.; Hu, S.; Xu, Z.; Chen, J.; Hao, M. K.; Han, X.; Thai, Z. L.; Wang, S.; Liu, Z.; and Sun, M. 2024. ∞Bench: Extending Long Context Evaluation Beyond 100K Tokens. arXiv:2402.13718.

Zhao, T.; Chen, J.; Zhang, S.; Zhu, H.; Lin, Q.; and Liu, J. 2026. Exploring Knowledge Conflicts for Faithful LLM Reasoning: Benchmark and Method. arXiv:2604.11209.

Zhao, Y.; Chen, W.; Xu, Z.; Patwardhan, M.; Wang, C.; Liu, Y.; Vig, L.; and Cohan, A. 2025. AbGen: Evaluating Large Language Models in Ablation Study Design and Evaluation for Scientific Research. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12479–12491. https://aclanthology.org/2025.acl-long.611/.