# PROMPT-LEVEL ADAPTIVE MEMORY COMPRESSION FOR DETECTING UNRELIABLE MEMORY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

LLM agents rely on long contexts held in working memory, from requirements retained by coding agents to evidence gathered by research agents. Compressing this memory lowers reading cost, but lost or altered details can make it unreliable for a given prompt, letting errors propagate through later actions. Existing compression methods treat compressed memory as a fixed input and offer little way to tell when it should not be trusted. We formulate prompt-level adaptive compression, in which an agent uses compressed memory while identifying, prompt by prompt, when it is unreliable, and propose Gated Context Memory (GCM). GCM stores context as continuous latent states produced by a *writer* and consumed by a *reader* that answers the prompt: a simple writer retains prompt-relevant information without converting it into text, and the confidence of a reader trained to use this memory can indicate when the memory falls short of what the prompt requires. The writer and reader share one frozen LLM with a small set of trained parameters. The trained reader supplies this signal at its first decoding step, without generating a complete answer, consulting an external verifier, or training a separate failure predictor, and the signal can drive any correction strategy. Across tool use, multi-hop reasoning, and long-document question answering, GCM outperforms existing compression baselines, and with raw-context fallback it matches or exceeds raw-context performance for most readers. Transfer experiments across tasks, readers, and compressors show where the signal generalizes and where it breaks down.

## 1 INTRODUCTION

Large language model (LLM) agents increasingly rely on long contexts, accumulating a working memory that carries information across actions: coding agents retain requirements while debugging (Yang et al., 2024), and research agents connect evidence across documents (Shao et al., 2024). As this history grows, reading it at every step becomes increasingly costly. Compression reduces this cost but is inherently lossy: it may drop or distort exactly the information that a later prompt depends on, leaving the compressed memory unreliable for that prompt (Yu et al., 2026a; Cim et al., 2026). In Figure 1, for example, a meeting time altered during compression leads the agent to take an incorrect action. Such errors can be corrected only if they are detected before they propagate to later steps (Belikova et al., 2026; Zhao et al., 2026; Min et al., 2026). Because whether compressed memory is reliable depends on what the current prompt requires, such detection must be made for each prompt. We introduce *prompt-level adaptive compression*, a setting in which an agent must rely on compressed memory while recognizing, for each prompt, when that memory is unreliable.

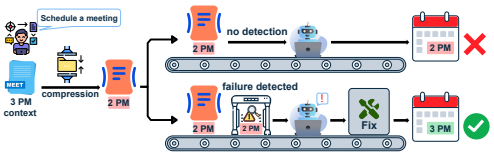



Figure 1: Prompt-level adaptive memory compression detects when compressed memory is unreliable for the current prompt and enables corrective action before an error propagates.

Existing approaches fall into five broad categories: retrieval, summarization, token pruning, KV-cache compression, and latent encoding. Retrieval selects relevant stored text (Chhikara et al., 2025); summarization rewrites history into a shorter account (Cim et al., 2026); token pruning removes less informative input tokens (Jiang et al., 2023; 2024; Pan et al., 2024); KV-cache compression retains



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the cached states of selected tokens (Xiao et al., 2024; Zhang et al., 2023; Kim et al., 2025); and latent encoding maps context to a shorter sequence of continuous vectors (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024; Kim et al., 2024; Zhang et al., 2026; Li et al., 2026). Whatever the design, a limited budget cannot guarantee retaining everything a prompt needs, so compressed memory may support a correct answer to one prompt yet mislead the model on another. However, these methods largely treat compressed memory as a fixed input and offer little means of assessing whether it is reliable for the current prompt (Belikova et al., 2026; Zhao et al., 2026; Wang et al., 2026).

We assess reliability through two components: a *writer*, the compressor that produces memory, and a *reader*, the language model that consumes it to answer the current prompt. Their design starts with memory's form: it need not be tied to individual tokens. Retrieval, summarization, token pruning, and KV-cache compression all represent memory through tokens, whether as selected text, rewritten text, a pruned subset, or the cached states of retained tokens. Latent encoding removes this constraint by storing context directly as continuous vectors. We build on this latent form of memory with a simple writer that retains information without first interpreting it into text, leaving interpretation to the reader. When the reader cannot recover what the prompt requires, this difficulty can then surface as low confidence rather than as a fluent but incorrect account. This couples information retention with a prompt-level reliability signal, which the agent can use to decide when to trust compressed memory and when to fall back on the raw (uncompressed) context.

Gated Context Memory (GCM) instantiates this division of work for prompt-level adaptive compression. It requires only lightweight training: the same frozen LLM serves as both the writer and the reader, and only a small set of added parameters is trained. Once trained, the reader assesses reliability as it begins answering. Using this initial response avoids generating a full answer and then checking it with an external verifier, or training a separate model to predict when compression fails. Each prompt thus receives a reliability decision at little additional inference cost. Because the writer and reader are modular, the same reader can also assess memory produced by other compressors. Detection likewise leaves correction open: unreliable memory can be handled by retrieving evidence, revising memory, or requesting verification. We adopt the simplest strategy: GCM gates between the two inputs: confident predictions use compressed memory; otherwise, the agent reads raw context.

We evaluate GCM on three tasks that reflect how agents rely on memory: tool use (BFCL), which requires retaining precise function specifications; multi-hop reasoning (HotpotQA), which requires connecting evidence across documents; and long-document question answering (NarrativeQA), which requires drawing on an extended history. We further test whether GCM generalizes along three axes: to new tasks, through transfer to LongBench without target-specific tuning; to other compressors, by pairing the GCM reader with an alternative writer; and to other models, across LLM readers from multiple model families. GCM is the strongest compression-only method on all three tasks and achieves the highest mean on LongBench transfer. With gating, it improves over raw context on HotpotQA for most tested readers while answering part of the prompts from compressed memory. These results suggest that compressed memory should be designed both to retain useful information and to reveal when it is unreliable. Our contributions are threefold:

1. **Detectable compression failures.** We introduce prompt-level adaptive compression, a setting in which an agent must both preserve useful information in compressed memory and detect, for each prompt, when that memory is unreliable. We frame it as a writer–reader co-design problem, enabling the agent to adapt its reliance on memory.

2. **Low-overhead adaptive memory.** We develop GCM, a simple, modular framework that requires only lightweight training around a frozen LLM. It assesses reliability as the reader begins answering, without an external verifier or a separate failure predictor; this signal outperforms alternatives, including learned failure predictors, in most comparisons. The framework also supports alternative compressors and correction strategies.

3. **Performance recovery and transfer.** Experiments show that GCM outperforms existing compression baselines on tool use, multi-hop reasoning, and long-document question answering, and transfers to LongBench without target-specific tuning. With gating, it matches or exceeds raw-context performance for most readers while answering part of the prompts from compressed memory. Cross-task, cross-reader, and alternative-writer evaluations demonstrate transferability and identify its limits.

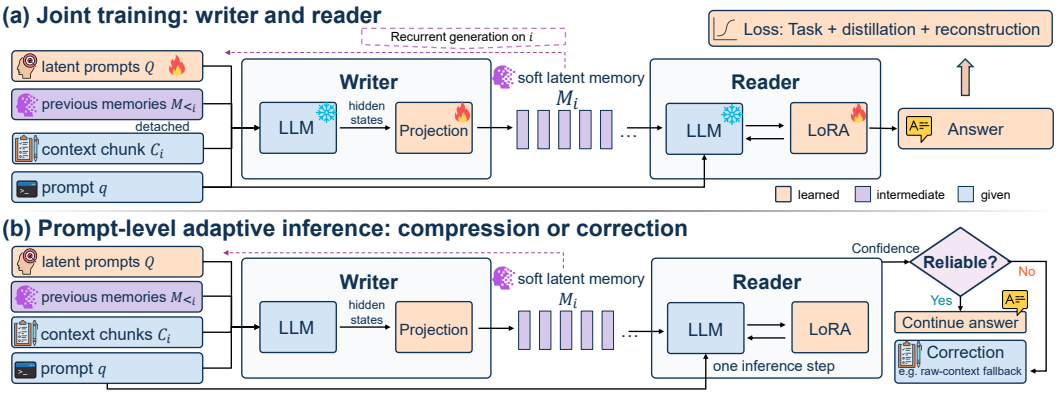



Figure 2: **Gated Context Memory.** (a) Joint writer-reader training for soft latent memory. Flames mark learned parameters; snowflakes mark fixed parameters during training. (b) Confidence enables adaptive use before full generation, with raw-context fallback as one correction strategy.

## 2 Prompt-Level Adaptive Memory Compression

**Assess compression for each prompt.** We study prompt-level adaptive memory compression, where an LLM answers a prompt using compressed context and assesses whether that compression is reliable for the prompt. We define a compression failure as a decrease in task performance relative to using raw context for the same prompt. Our goal is to design compressed memory that preserves useful information while making such failures easier to detect.

### 2.1 Definition: Compression Reliability

We define reliability relative to raw context to identify when compression harms an answer. For context $C$, prompt $q$, and reference answer $y$, let $\hat{y}_{\text{mem}}$ and $\hat{y}_{\text{raw}}$ be predictions from compressed memory and raw context, respectively. With higher-is-better task score $u$, the detection target is

$$z(C, q) = \mathbf{1}[u(\hat{y}_{\text{mem}}, y) \geq u(\hat{y}_{\text{raw}}, y)]\,. \tag{1}$$

Here $\mathbf{1}$ is the indicator function. A positive label ($z = 1$) means compression does not lower the task score for this prompt, rather than guaranteeing a correct answer: both predictions can be wrong. Although reference answers establish this relative-performance target for evaluation, they are unavailable at inference. The detector must therefore infer reliability from the compressed-memory LLM (reader), without comparing completed answers against a reference.

**Retain useful information and expose failures.** This objective couples two requirements. Memory must retain information that enables useful answers, and the reader must provide an informative signal when compression lowers answer quality. Optimizing only answer quality does not explicitly distinguish a confidently incorrect interpretation from a reliable one. Conversely, a detection signal alone cannot restore evidence that the representation has discarded. We therefore design memory and its reader together, while treating correction as a separate choice.

**The writer retains; the reader interprets.** Gated Context Memory (GCM; Figure 2) implements this joint design through a context compressor (writer) and the reader. To support both requirements, the writer retains information as soft latent states rather than interpreting it into a textual account. This leaves the reader to determine how the retained information supports the current prompt. Our hypothesis is that this division also helps expose failures: when the reader cannot use the compressed information reliably, that difficulty can remain visible in its confidence. We implement this division through the writer architecture and losses for source reconstruction and prediction consistency, without failure labels for training either component; detection instead uses the reader's response to the resulting memory.

### 2.2 PROMPT-CONDITIONED WRITER

To support useful answers and detectable failures, the writer retains prompt-relevant information without first interpreting it into text. It collects intermediate states at a small set of learned positions. Let $C = (C_1, \ldots, C_S)$ contain $S$ context chunks and $Q = [Q_1; \ldots; Q_K] \in \mathbb{R}^{K \times d}$ contain $K$ learned latent prompts of hidden dimension $d$, with brackets denoting sequence concatenation. For token embeddings $e(\cdot)$ and frozen LLM parameters $\theta$, the first $D$ layers $F_\theta^{1:D}$ compute

$$\begin{aligned} X_i &= [\mathrm{sg}(M_{<i}); e(C_i); e(q); Q], \quad H_i = F_\theta^{1:D}(X_i), \\ M_i &= \mathrm{Norm}(\mathrm{MLP}_\phi(H_i[Q_* : Q_* + K])) \,. \end{aligned} \tag{2}$$

Here $X_i$ is the writer input, $H_i$ its hidden states, and $M_i$ the chunk memory. sg stops gradients; $Q_*$ starts the slice of $K$ latent-prompt states. The multilayer perceptron (MLP) has parameters $\phi$; Norm is defined below. Placing $Q$ after the context and prompt lets its hidden states collect prompt-relevant information. We extract these states halfway through the LLM, then project and normalize them for use as reader inputs. For later chunks, earlier memories $M_{<i} = [M_1; \ldots; M_{i-1}]$ (empty for $i = 1$) provide context without cross-chunk gradients. Their concatenation $M = [M_1; \ldots; M_S]$ forms the reader's $SK$-state input, rebuilt for each new prompt.

**Select for the prompt and carry context forward.** Although the learned vectors $Q$ are shared across examples, their hidden states depend on the preceding context and user prompt. Causal attention provides this dependence by allowing each latent-prompt position to attend to the current chunk, the prompt, and earlier memories. Retention is thus prompt-dependent without requiring task-specific learned vectors or explicit failure categories. Instead, patterns shared across prompts can influence the reader through the latent states that these shared vectors collect.

For long inputs, earlier memories inform later chunks while remaining available to the final reader in document order. Thus $K$ is a per-chunk budget, not a document-level limit. Detachment stops cross-chunk gradients without removing earlier memories from later writer inputs.

**Leave intermediate states for the reader to interpret.** The choice of writer depth follows the same division of work. Intermediate representations may retain source details that later, output-oriented layers suppress; extracting them before the final layer leaves their interpretation to the reader. We therefore use the first half of the frozen LLM, which also avoids computing the remaining layers during writing. Appendix B connects this choice to information bottleneck analyses and compares writer depths through answer accuracy and detection AUROC.

**Match latent states to the reader's input scale.** To make these intermediate states usable as reader inputs, we apply a two-layer $d \to 2d \to d$ MLP projection with GELU activation. For projected state $a$, vocabulary $\mathcal{V}$, token embedding $e_v = e(v)$, and numerical stabilizer $\varepsilon > 0$, normalization computes

$$\mathrm{Norm}(a) = \frac{a}{\|a\|_2 + \varepsilon} \mathbb{E}_{v \in \mathcal{V}} \|e_v\|_2. \tag{3}$$

Matching this magnitude to the LLM's token embeddings lets the reader learn the latent input format without relying on a change in activation scale. The resulting memory retains an ordered sequence structure, although its states need not correspond one-to-one to original token positions.

### 2.3 COMPRESSED-MEMORY READER

The reader must turn these latent states into useful answers while keeping the same LLM's pretrained weights fixed. We adapt it using a low-rank adaptation (LoRA) adapter $\psi$ (Hu et al., 2022), defining $p_{\mathrm{mem}} = F_{\theta,\psi}(\cdot \mid [M; q])$, its answer distribution given memory and prompt (with $q$ embedded by $e$). Training updates the latent prompts $Q$, projection $\phi$, and reader adapter $\psi$, with the LLM parameters $\theta$ fixed throughout optimization to preserve its pretrained knowledge.

**Adapt latent reading while keeping the LLM frozen.** The writer stops at intermediate layers, whereas the reader uses the complete LLM with its adapter active. Keeping this adapter separate from the frozen weights allows it to be disabled for raw-context evaluation. Each LLM learns its own latent prompts, projection, and reader adapter. Cross-task reuse keeps these learned components fixed while changing the input task.

### 2.4 COMPLEMENTARY TRAINING LOSSES

We train the writer and reader jointly to retain source information and use it consistently, beyond fitting the answer alone. We combine answer cross-entropy $\mathcal{L}_{\text{task}}$, raw-context prediction distillation $\mathcal{L}_{\text{distill}}$, and source reconstruction $\mathcal{L}_{\text{recon}}$. With nonnegative loss weights $\lambda_{\text{distill}}$ and $\lambda_{\text{recon}}$, the total training loss is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{distill}}\mathcal{L}_{\text{distill}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}}. \tag{4}$$

**Learn the answer and preserve the LLM's predictions.** For a reference answer of length $T$, $\mathcal{L}_{\text{task}}$ applies teacher-forced cross-entropy at the reader's answer positions, sending gradients through both the reader and current memory construction. Let $p^{(t)}_{\text{raw}}$ and $p^{(t)}_{\text{mem}}$ denote next-token distributions from raw context and compressed memory under the same reference-answer prefix at position $t$. Distillation minimizes their Kullback–Leibler divergence:

$$\mathcal{L}_{\text{distill}} = \frac{1}{T}\sum_{t=1}^{T} \text{KL}\Big(p^{(t)}_{\text{raw}} \,\|\, p^{(t)}_{\text{mem}}\Big). \tag{5}$$

The frozen raw-context teacher uses the dataset-specific input limit (Appendix H), anchoring training to the same model knowledge. Unlike the single reference token, this distribution also supervises the relative probabilities of alternative outputs, encouraging consistency between reading latent memory and reading the original context.

**Preserve source information beyond the answer.** Answer distributions do not constrain every source detail. Reconstruction therefore asks whether memory can also reproduce the source. Let $(c_1, \ldots, c_J)$ be the $J$ source tokens selected for reconstruction (up to 512). The reader receives memory followed by $J$ repetitions of a learned, content-free input vector. For its hidden state $h_j$ at slot $j$ and the LLM's input embedding matrix $E$, reused for output prediction, the objective is

$$\mathcal{L}_{\text{recon}} = -\frac{1}{J}\sum_{j=1}^{J} \log[\text{softmax}(Eh_j)]_{c_j}. \tag{6}$$

Under causal attention, each slot can use memory and earlier slots, but not the source tokens themselves: these appear only as prediction targets. Each slot predicts its corresponding source position directly, without the next-token shift or teacher-forced source prefix used in generation.

**Preserve both information and its use.** Reconstruction and distillation address complementary ambiguities in the compressed representation. Different memories can produce similar answer distributions despite omitting source details, which reconstruction asks the reader to recover. Conversely, recoverable information may still be used differently from raw context, which distillation addresses by aligning predictions. Together, the objectives encourage information retention and predictive consistency without requiring the writer to turn every pattern into text. Figures 6 and 7 test these losses' effects on answer quality, training variability, and failure detection.

### 2.5 FAILURE DETECTION AND ADAPTIVE USE

With the writer and reader trained, we use the reader's initial confidence to assess memory reliability without generating a complete answer. Specifically, first-token confidence is the maximum probability in the first output distribution, where $y_1$ is the first answer token. Gating compares this score with a threshold $\tau$ to decide whether to continue using compressed memory (Figure 2(b)):

$$\begin{aligned} s_{\text{conf}}(M, q) &= \max_{v \in \mathcal{V}} p_{\text{mem}}(y_1 = v \mid M, q), \\ p_{\text{GCM}}(y \mid C, q) &= \begin{cases} F_{\theta,\psi}(y \mid [M; q]), & s_{\text{conf}}(M, q) > \tau, \\ F_{\theta}(y \mid [C_{\le B}; q]), & s_{\text{conf}}(M, q) \le \tau. \end{cases} \end{aligned} \tag{7}$$

Here $C_{\le B}$ is context truncated to the dataset's $B$-token limit, and $p_{\text{GCM}}$ is the adaptive answer distribution. Raw-context fallback provides the simplest correction: when confidence is at or below the threshold, we disable LoRA and read at most $B$ context tokens. We determine a threshold range empirically by comparing compressed-memory and raw-context scores (Figure 4), with model parameters fixed. This policy uses confidence because it leads the alternative signals in 15 of 17 model-task runs (Appendix A.4); other harnesses may choose a different correction.

**Assess reliability before generating the answer.** Computing this score requires processing memory and the prompt, but stops at the first output distribution rather than generating a complete answer for a second model to judge. If memory is accepted, answer generation continues from the same computation; otherwise, the evaluation policy switches to raw context with the adapter disabled. The extra work in the latter case makes end-to-end savings depend on fallback frequency, even though detection itself occurs early.

**Train the writer and reader together.** To obtain this reader signal, all three losses supervise the same memory (Figure 2(a)). The reader predicts the reference answer from the concatenated chunk memories, the frozen raw-context LLM supplies distillation targets, and a separate slot-based pass reconstructs source tokens. Gradients from these losses update the latent prompts, projection, and reader adapter; reconstruction also learns the input slot. Detachment blocks gradients between writer calls, but answer supervision reaches every chunk memory used by the reader.

**Calibrate when to use compressed memory.** With training complete, confidence provides a continuous reliability score; threshold selection determines how the policy uses it. We select a threshold on held-out compressed-memory and raw-context scores to reduce fallback while limiting score decreases on prompts using memory, then apply it unchanged to evaluation examples. The threshold thus controls compressed-memory use without changing the score or retraining the writer and reader.

Separating the score from its threshold also separates detection from the response to unreliable memory. Our fallback policy tests whether the signal can guide a direct correction, while other harnesses could use it to retrieve evidence, revise memory, or invoke verification. Our evaluation measures detection with threshold-independent AUROC. Task performance and fallback ratio describe the resulting use policy rather than the score itself.

**Definition: calibration criterion.** Let $A_\tau = \mathbf{1}[s_{\text{conf}} > \tau]$ select compressed memory and $\Delta = u(\hat{y}_{\text{raw}}, y) - u(\hat{y}_{\text{mem}}, y)$ denote its score decrease. For an allowed mean score decrease $\eta$, the policy minimizes fallback:

$$\min_{\tau} \Pr(A_\tau = 0) \quad \text{subject to} \quad \frac{\mathbb{E}[A_\tau \max(\Delta, 0)]}{\mathbb{E}[A_\tau]} \leq \eta. \tag{8}$$

Here $\Pr$ and $\mathbb{E}$ denote probability and expectation, estimated on held-out data with at least one prompt using memory. Counting only decreases prevents gains on other prompts from canceling failures. The selected threshold changes the use policy, not the writer, reader, or continuous detection scores. At inference, only memory construction and the reader's first output distribution are needed to apply this policy; neither retention-loss computation nor reference answers are required.

**Detect failures without reference answers.** Training and deployment therefore use the same writer-reader computation but different supervision. Training compares memory-conditioned predictions with source tokens and raw-context predictions; deployment uses only the learned representation and the reader's response to it. In particular, the detector does not first generate a raw-context answer to assess compression. Raw-context computation is deferred until the selected policy requests it. This distinction allows a harness to assess each prompt before committing to a complete answer, while choosing its own response to unreliable memory. Because the writer is prompt-conditioned, a new prompt requires rebuilding memory; shared learned parameters do not imply reusing the same latent states when the prompt changes which information is relevant.

**Discussion: retention and detection.** Information bottleneck analyses motivate retaining intermediate information before output-oriented compression (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017). Our simple writer preserves useful distinctions for the reader to interpret. We hypothesize that this division preserves answer evidence while allowing patterns unfamiliar to the reader to remain visible in confidence. Reconstruction supports information retention; distillation supports its consistent use. Writer-depth, budget, and loss ablations test these choices, while transfer tests their reuse across tasks and writers. The common test is whether the reader can both use retained evidence and signal when that evidence is insufficient for the prompt (Appendix B).

Table 1: Comparison with compression baselines using Qwen3-8B and one context chunk ($S = 1$). BFCL reports accuracy, HotpotQA answer F1, and NarrativeQA ROUGE-L, all on a 0–100 scale. Bold marks group bests; context budgets exclude prompts.

| compression method | BFCL (%) | HotpotQA (%) | NarrativeQA (%) | context budget |
|---|---|---|---|---|
| LLMLingua (Jiang et al., 2023) | 60.1 | 22.2 | 12.1 | 128 tokens |
| LongLLMLingua (Jiang et al., 2024) | 15.2 | 11.7 | 4.4 | 128 tokens |
| DAC (Zhao et al., 2025) | 62.3 | 26.6 | 13.6 | 128 tokens |
| CriticalKV (Feng et al., 2026) | 70.6 | 28.6 | 12.2 | 128 KV positions |
| LagKV (Liang et al., 2025) | 54.7 | 27.7 | 12.9 | 128 KV positions |
| ICAE (Ge et al., 2024) | 48.4 | 27.5 | 13.8 | 128 states |
| COCOM (Rau et al., 2024) | 71.5 | 27.5 | 17.4 | 128 states |
| PISCO (Louis et al., 2025) | 70.6 | 18.5 | 8.9 | 128 states |
| OSCAR (Louis et al., 2026) | 67.4 | 18.2 | 9.4 | 128 states |
| GCM w/o gating (ours) | **72.3±0.5** | **28.8±0.4** | **16.9** | $128S$ states |
| GCM w/ gating (ours) | 91.7±0.4 | **53.6±0.1** | **18.1** | $128S$ / $B$ |
| Raw context | **92.1** | 53.1 | 14.2 | $\leq B$ tokens |

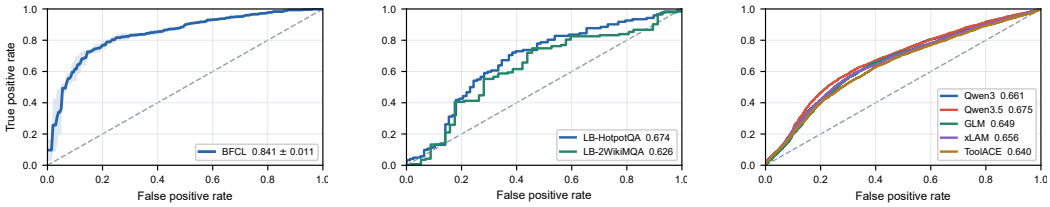



Figure 3: **Confidence detects compression failures.** Left: Qwen3.5-9B on BFCL Live Multiple. Center: Qwen3-8B on LB-HotpotQA and LB-2WikiMQA. Right: five HotpotQA readers. Legends report AUROC; positives score at least as well as raw context.

## 3 Experiments

We first test whether compression preserves task performance and whether detecting failures supports adaptive recovery. We then examine transfer across tasks, readers, and writers, before isolating the role of the training objectives.

**Tasks and models.** BFCL tests tool selection (Patil et al., 2025), HotpotQA multi-hop QA (Yang et al., 2018), and NarrativeQA long-document QA (Kočiský et al., 2018). LongBench transfer (Bai et al., 2024) uses HotpotQA-trained writers and readers without target-task tuning on LB-HotpotQA and LB-2WikiMQA across five readers. The LB prefix identifies the LongBench evaluations; HotpotQA denotes the standalone evaluation.

**Budgets and baselines.** The default is $K = 128$ latent states per chunk, with $S = 1$ in the main comparison and $SK$ reader positions for multi-chunk inputs. Table 1 compares text, KV-cache, and latent compression; LongBench additionally includes BM25 (Robertson & Zaragoza, 2009) and truncation. Budgets count reader input positions and exclude prompts. Compression-only results test information retention; adaptive results test whether the reader's reliability signal supports recovery relative to raw context.

**Metrics and policy.** We report accuracy on BFCL, ROUGE-L (Lin, 2004) on NarrativeQA, and answer F1 on QA, all on a 0-100 scale. The task-format diagnostic uses first-line F1. AUROC (0-1) evaluates the paired reliability target in Equation 1; fallback ratio is the fraction answered from raw context. Thresholds are fixed before held-out evaluation. Controlled multi-run results report mean and standard deviation. Full training and evaluation settings are given in Appendix H.

**Compression quality and adaptive recovery.** Table 1 compares compression quality and adaptive recovery. Adaptive use recovers most raw-context performance on BFCL and HotpotQA and improves NarrativeQA, linking information retention to failure detection.

**Confidence distributions and adaptive use.** BFCL positives concentrate at high confidence, whereas HotpotQA failures shift toward low confidence (Figure 4), supporting reader-based detection. AUROC evaluates detection; fallback ratio depends on the threshold. Fallback is the simplest evaluated correction, not a required response.

Table 2: HotpotQA-trained GCM on LongBench without target tuning (answer F1, %). Each input fills $S$ chunks of up to 4,096 tokens: GCM uses $128S$ states; textual baselines use 128 tokens total. Budgets exclude prompts. Bold marks group-best scores.

| dataset | compression method | Qwen3.5 | Qwen3 | xLAM | GLM-4 | Ministral | avg. perf. |
|---|---|---|---|---|---|---|---|
| LB-HotpotQA | no context (0) | 8.0 | 8.3 | 8.8 | 12.5 | 16.1 | 10.7 |
| | truncation (128) | 4.7 | 8.5 | 6.3 | 11.8 | 16.6 | 9.6 |
| | BM25 retrieval | 21.7 | 16.0 | 15.5 | **24.4** | 25.9 | 20.7 |
| | LLMLingua | 10.7 | 8.5 | 8.9 | 12.1 | 18.6 | 11.8 |
| | GCM w/o gating (ours) | **27.4** | **29.9** | **33.1** | 24.0 | **26.8** | **28.2** |
| | raw context (16K) | 4.1 | 19.2 | 17.3 | 31.2 | 27.6 | 19.9 |
| | raw context (32K) | 2.9 | 18.2 | 17.3 | 30.8 | 26.8 | 19.2 |
| | GCM w/ gating (ours) | **27.4** | **31.4** | **33.1** | **35.1** | **34.4** | **32.3** |
| LB-2WikiMQA | no context (0) | 10.6 | 9.9 | 8.0 | 12.8 | 15.9 | 11.4 |
| | truncation (128) | 3.2 | 5.5 | 8.8 | 14.5 | 12.7 | 8.9 |
| | BM25 retrieval | 10.0 | 10.1 | 10.9 | 13.7 | 18.6 | 12.7 |
| | LLMLingua | 7.4 | 6.1 | 8.8 | 9.2 | 14.9 | 9.3 |
| | GCM w/o gating (ours) | **28.1** | **26.9** | **29.9** | **29.6** | **20.7** | **27.1** |
| | raw context (16K) | 3.2 | 12.4 | 14.4 | 25.9 | 24.2 | 16.0 |
| | raw context (32K) | 3.2 | 12.5 | 14.4 | 25.8 | **24.2** | 16.0 |
| | GCM w/ gating (ours) | **28.1** | **26.9** | **30.8** | **34.1** | 22.6 | **28.5** |

Table 3: HotpotQA answer F1 (%) with GCM trained per reader. Gating selects compressed or raw context; Δ raw is gated minus raw F1. Fallback ratio (%) measures raw-context use.

| reader | GCM (ours) w/o gating | raw | GCM (ours) w/ gating | Δ raw (pp) | fallback ratio (%) | AUROC |
|---|---|---|---|---|---|---|
| Qwen3-8B | 31.04 | 53.11 | **54.06** | +0.95 | 78.4 | 0.661 |
| Qwen3.5-9B | 30.44 | 53.86 | **56.01** | +2.15 | 80.2 | 0.675 |
| GLM-4-9B | 32.87 | 53.66 | **55.09** | +1.42 | 74.5 | 0.649 |
| xLAM-8B | 32.71 | 44.80 | **50.95** | +6.16 | 60.4 | 0.656 |
| ToolACE-8B | 31.15 | 46.95 | **51.92** | +4.98 | 68.2 | 0.640 |

**Long-context compression transfer.** Beyond in-domain recovery, GCM trained on HotpotQA achieves the highest compression-only mean on both LongBench QA datasets without target-task training (Table 2). Gains on LB-2WikiMQA show that useful compressed answers extend beyond the training dataset without relying on fallback. This tests reuse of the learned compression design, complementing the in-domain comparison with raw context.

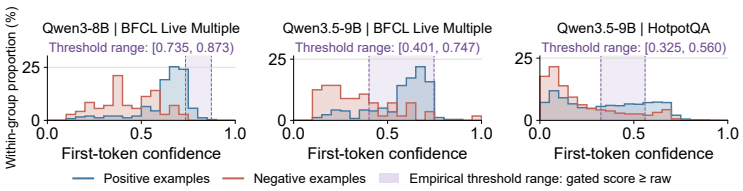



Figure 4: **Confidence distributions separate relative outcomes.** Positives/negatives score at least as well as/worse than raw context. Groups are normalized separately. Shading marks empirical threshold intervals with gated mean score ≥ raw, not held-out guarantees.

**Detection with a replacement writer.** We retain the GCM-trained reader and confidence-based gate, replacing only the writer with matched-budget Cartridges (Eyuboglu et al., 2026) (Figure 5). Detection remains informative with Cartridges, supporting reuse of the trained reader and gate beyond the original writer.

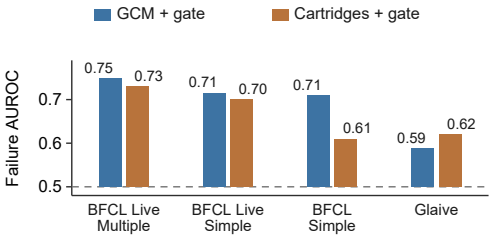



Figure 5: Detection AUROC: replace only the writer; keep the GCM-trained reader and gate.

**Detection across readers and tasks.** Across five separately trained HotpotQA readers, confidence attains AUROC 0.640-0.675, supporting gains of 0.95-6.16 points over raw context (Table 3; Figure 3). With its HotpotQA-trained writer and reader fixed, Qwen3-8B reaches AUROC 0.674 on LB-HotpotQA and 0.626 on LB-2WikiMQA, extending detection beyond training.

**Writer transfer.** Table 4 shows gains on TriviaQA for both readers, supporting reuse of answer-relevant evidence. Passage retrieval instead requires exact positions, while meeting summaries require broader coverage. This contrast suggests that retaining salient evidence transfers more readily than preserving document structure or exhaustive detail.

Table 4: HotpotQA-trained GCM without gating on LongBench, with no target-task tuning. Answer F1 (%); Δ is the compressed score minus raw context in F1 points.

| | Qwen3-8B | | | Qwen3.5-9B | | |
|---|---|---|---|---|---|---|
| task | raw | GCM (ours) w/o gating | Δ | raw | GCM (ours) w/o gating | Δ |
| LB-TriviaQA | 74.3 | **89.7** | +15.4 | 78.9 | **88.1** | +9.3 |
| LB-PassageCount | 0.6 | **3.0** | +2.4 | **3.3** | 3.0 | −0.3 |
| LB-PassageRetrieval | **0.3** | 0.1 | −0.2 | **8.1** | 0.0 | −8.1 |
| LB-QMSum | **26.6** | 14.5 | −12.1 | **19.8** | 10.6 | −9.2 |

**Complementary supervision and stability.** To examine what supports this performance, we ablate training objectives. Removing task gradients changes the mean by 0.9 points but raises standard deviation from 0.5 to 2.9 (Figure 6). The larger effect on variance than on the mean suggests that task supervision chiefly stabilizes learning, rather than expanding writer capacity. This matches its role in maintaining usable reader inputs.

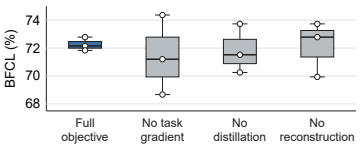



Figure 6: BFCL accuracy across runs with each loss removed.

**Retention supervision and failure detection.** The complementary question is whether these objectives support detection. Removing distillation or reconstruction reduces AUROC to 0.45 or 0.48 (Figure 7). Preserving source information and aligning its use by the reader thus serve complementary roles beyond task accuracy. The loss ablations therefore test both parts of the design: useful compressed inputs and detectable failures.

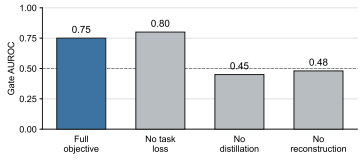



Figure 7: BFCL AUROC with each loss removed.

**Experimental takeaways.** (1) Task and writer transfer suggest shared compression-failure patterns (Figures 3 and 5). Consistent detection across separately trained readers further motivates this hypothesis, in line with shared representations (Huh et al., 2024). (2) Task gradients stabilize answer quality, while retention losses support failure detection (Figures 6 and 7). Together with adaptive recovery, these results support a writer that retains information and a reader that both uses it and signals unreliability. Further experimental details appear in the appendix.

## 4 Related Work

Text-selection methods such as BM25, LLMLingua, and LongLLMLingua (Robertson & Zaragoza, 2009; Jiang et al., 2023; 2024), and KV-cache methods such as StreamingLLM, $H_2O$, and KVzip (Xiao et al., 2024; Zhang et al., 2023; Kim et al., 2025), decide what information to retain. GCM adds reader-based detection that also supports an alternative writer. Recent work detects overflow in compressed representations (Belikova et al., 2026), learns compression-aware abstention (Khodabandehlou & Krishnamachari, 2026), or uses first-token uncertainty to activate context-specific adapters (Zheng et al., 2026). GCM instead jointly designs prompt-conditioned memory and its reader for useful compression and early failure detection, without prescribing a correction policy. Latent methods learn compact representations (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024; Rau et al., 2024; Louis et al., 2025; 2026; Li et al., 2026; Zhou & Sang, 2026; Zhang et al., 2026); LatentPress combines a small writer with reconstruction and distillation (Zhou & Sang, 2026). GCM emphasizes reader-oriented training for both useful answers and detectable failures. Cartridges instead optimizes compact KV memory per corpus (Eyuboglu et al., 2026). GCM instead constructs memory for new prompts without target-task training.

## 5 Conclusion

We introduced Gated Context Memory (GCM) for prompt-level adaptive compression, combining a simple latent writer and adapted reader. Initial confidence detects unreliable memory without a separate verifier or full generation. GCM improves over compression baselines and, with fallback, recovers raw-context performance for most tested readers. Transfer and ablations support jointly preserving information and exposing failures, while leaving correction open.

### AI USE STATEMENT

In this work, we used generative AI tools primarily to assist with testing code modules and to improve the presentation of the manuscript, including language editing and the refinement of figures and tables. All AI-assisted outputs were reviewed and verified by the authors. We take responsibility for the final content, claims, code, and artifacts.

### REPRODUCIBILITY STATEMENT

All implementation details required to reproduce our method and experiments are provided in the paper and appendices, including the writer and reader, training objectives, gate calibration, training budgets, ablations, and transfer settings. We will publicly release our code and trained model checkpoints. After the paper and accompanying artifacts are publicly available, we will continue to provide technical support to the community to facilitate reproduction and further research.

## REFERENCES

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025. doi: 10.1214/24-AOAS1998.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172.

Julia Belikova, Danila Rozhevskii, Dennis Svirin, Konstantin Polev, and Alexander Panchenko. Detecting overflow in compressed token representations for retrieval-augmented generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 797–810, 2026. doi: 10.18653/v1/2026.eacl-srw.59.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023. doi: 10.18653/v1/2023.emnlp-main.232.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. arXiv preprint arXiv:2504.19413, 2025.

Musa Cim, Burak Topcu, Chita Das, and Mahmut Taylan Kandemir. Parallel context compaction for long-horizon LLM agent serving. arXiv preprint arXiv:2605.23296, 2026.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Atri Rudra, James Y. Zou, Azalia Mirhoseini, and Christopher Ré. Cartridges: Lightweight and general-purpose long context representations via self-study. In *International Conference on Learning Representations*, 2026.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. CriticalKV: Optimizing KV cache eviction from an output perturbation perspective. In *International Conference on Machine Learning*, 2026. URL `https://arxiv.org/abs/2502.03805`.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, 2017.

Glaive AI. Glaive function calling v2. Hugging Face dataset, 2023. URL https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580.

Yubo Hou, Zhisheng Chen, Tao Wan, and Zengchang Qin. FlashMem: Distilling intrinsic latent memory via computation reuse. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4687–4705, 2026. doi: 10.18653/v1/2026.findings-acl.230.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 20617–20642, 2024.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023. doi: 10.18653/v1/2023.emnlp-main.825.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024. doi: 10.18653/v1/2024.acl-long.91.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient context: A new lens on retrieval augmented generation systems. In *International Conference on Learning Representations*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1601–1611, 2017.

Mohammadali Khodabandehlou and Bhaskar Krishnamachari. Compression-aware abstention: Teaching LLMs to refuse when KV-compression masks remove answer evidence. arXiv preprint arXiv:2608.29934, 2026.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *International Conference on Learning Representations*, 2024.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems*, 2025.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023.

Yuri Kuratov, Mikhail Arkhipov, Aydar Bulatov, and Mikhail Burtsev. Cramming 1568 tokens into a single vector and back again: Exploring the limits of embedding space capacity. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19323–19339, 2025. doi: 10.18653/v1/2025.acl-long.948.

Ang Li, Sean McLeish, Haozhe Chen, Nimit Kalra, Zaiqian Chen, Artem Gazizov, Venkata Anoop Suhas Kumar Morisetty, Bhavya Kailkhura, Harshitha Menon, Zhuang Liu, Brian R. Bartoldson, Tom Goldstein, Sanae Lotfi, Micah Goldblum, and Pavel Izmailov. End-to-end context compression at scale. arXiv preprint arXiv:2606.09659, 2026.

Manlai Liang, JiaMing Zhang, Xiong Li, and Jinlong Li. LagKV: Lag-relative information of the KV cache tells which tokens are important. arXiv preprint arXiv:2504.04704, 2025. URL https://arxiv.org/abs/2504.04704.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. In *International Conference on Learning Representations*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Maxime Louis, Hervé Déjean, and Stéphane Clinchant. PISCO: Pretty simple compression for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://aclanthology.org/2025.findings-acl.800/.

Maxime Louis, Thibault Formal, Hervé Déjean, and Stéphane Clinchant. OSCAR: Online soft compression for RAG. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ideKAUWvFE.

Lvzhou Luo, Yixuan Cao, and Ping Luo. AttnComp: Attention-guided adaptive context compression for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8456–8472, 2025. doi: 10.18653/v1/2025.findings-emnlp.449.

Guanghui Min, Liang Wu, Mayank Darbari, Chen Chen, and Liangjie Hong. Toward reliable context compression for long-horizon agents: An empirical study of execution instability. arXiv preprint arXiv:2608.06503, 2026.

Mistral AI Team. Ministral-8b-instruct-2410 model card. Hugging Face model card, 2024. URL https://huggingface.co/mistralai/Ministral-8B-Instruct-2410.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL*, pp. 963–981, 2024. doi: 10.18653/v1/2024.findings-acl.57.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391/.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 48371–48392, 2025.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, et al. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. arXiv preprint arXiv:2504.03601, 2025.

Qwen Team. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Qwen Team. Qwen3.5-9b model card. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018. doi: 10.18653/v1/P18-2124.

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. Context embeddings for efficient answer generation in RAG. *arXiv preprint arXiv:2407.09252*, 2024. URL https://arxiv.org/abs/2407.09252.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. Assisting in writing Wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6252–6278. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.347. URL https://aclanthology.org/2024.naacl-long.347/.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810, 2017.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, et al. ChatGLM: A family of large language models from GLM-130b to GLM-4 all tools. arXiv preprint arXiv:2406.12793, 2024.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015. doi: 10.1109/ITW.2015.7133169.

Yufeng Wang, Lu Wei, and Haibin Ling. Retrieval as a decision: Training-free adaptive gating for efficient RAG. *Transactions on Machine Learning Research*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026a. doi: 10.18653/v1/2026.acl-long.981.

Yijiong Yu, Shuai Yuan, Jie Zheng, Huazheng Wang, and Ji Pei. Density-aware soft context compression with semi-dynamic compression ratio. arXiv preprint arXiv:2603.25926, 2026b.

Zeyu Zhang, Rui Li, Xiaoyan Zhao, Yang Zhang, Wenjie Wang, Xu Chen, and Tat-Seng Chua. NextMem: Towards latent factual memory for LLM-based agents. arXiv preprint arXiv:2603.15634, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Runsong Zhao, Shilei Liu, Jiwei Tang, Langming Liu, Haibin Chen, Weidong Zhang, Yujin Yuan, Tong Xiao, Jingbo Zhu, Wenbo Su, and Bo Zheng. PoC: Performance-oriented context compression for large language models via performance prediction. arXiv preprint arXiv:2603.19733, 2026.

Yi Zhao, Zuchao Li, Hai Zhao, Baoyuan Qi, and Guoming Liu. DAC: A dynamic attention-aware approach for task-agnostic prompt compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19395–19407, 2025. doi: 10.18653/v1/2025.acl-long.952. URL https://aclanthology.org/2025.acl-long.952/.

Ziyang Zheng, Zeju Li, Xiangyu Wen, Jianyuan Zhong, Junhua Huang, Lei Chen, Mingxuan Yuan, and Qiang Xu. Context distillation as latent memory management. arXiv preprint arXiv:2605.28889, 2026.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 5905–5921, 2021.

Zhengze Zhou and Hejian Sang. LatentPress: Context compression beyond text and vision. arXiv preprint arXiv:2609.01507, 2026.

# APPENDIX

## A METHOD AND IMPLEMENTATION DETAILS

### A.1 MEMORY CONSTRUCTION AND READER

The writer uses the first half of the frozen LLM to process each chunk. Latent prompts $Q$ collect $K$ hidden states per chunk, followed by a two-layer $d \to 2d \to d$ GELU projection. Earlier chunk memories condition later chunks and are detached between recurrent updates. The reader receives the concatenation $[M_1; \ldots; M_S]$, so the memory length is $SK$ rather than a fixed document-level $K$.

Projection outputs are normalized to the mean norm of the LLM input embeddings. The model preserves token order within chunks and chunk order across the document, but not exact original token positions. Each LLM learns separate latent prompts, projection, and reader adapter parameters.

The reader branch activates a rank-64 LoRA reader adapter, while raw-context fallback disables it and uses the same frozen LLM. The two paths therefore differ in representation and reader adaptation, not in the underlying language model.

Writing uses half of the LLM depth in the main configuration. The projection and normalization in Figure 2 compose as $\rho_\phi = \mathrm{Norm} \circ \mathrm{MLP}_\phi$:

$$\rho_\phi(h) = \frac{\mathrm{MLP}_\phi(h)}{\| \mathrm{MLP}_\phi(h) \|_2 + \varepsilon} \mathbb{E}_{v \in \mathcal{V}} \|e_v\|_2. \tag{9}$$

For a LLM matrix $W_0$, the reader adapter computes $W_0 x + (\alpha/r) B_\mathrm{L} A_\mathrm{L} x$, where $A_\mathrm{L}$ and $B_\mathrm{L}$ are trainable low-rank matrices. It is applied to attention and feed-forward projections and is not merged into the LLM. A fixed-state variant retains only the most recent recurrent states; it is not the concatenated-memory configuration reported in the main experiments.

### A.2 TRAINING OBJECTIVE

GCM minimizes

$$\mathcal{L} = \mathcal{L}_\mathrm{task} + \lambda_\mathrm{distill} \mathcal{L}_\mathrm{distill} + \lambda_\mathrm{recon} \mathcal{L}_\mathrm{recon}. \tag{10}$$

$\mathcal{L}_\mathrm{task}$ is teacher-forced answer cross-entropy under the reader branch, with gradients flowing through both the reader and memory construction. $\mathcal{L}_\mathrm{distill}$ matches the reader to raw-context predictions under the stated bound over the answer span. $\mathcal{L}_\mathrm{recon}$ asks a learned, content-free slot repeated across source positions to predict the corresponding source tokens through the tied language-model head. Source tokens are targets rather than decoder inputs, so this objective cannot exploit a teacher-forcing shortcut. Ablations separately remove the auxiliary losses, recurrence, and writer task gradients.

For an answer of length $T$, the first two terms are

$$\begin{aligned} \mathcal{L}_\mathrm{task} &= -\sum_{t=1}^{T} \log p_\mathrm{mem}(y_t \mid y_{<t}, M, q), \\ \mathcal{L}_\mathrm{distill} &= \frac{1}{T} \sum_{t=1}^{T} \mathrm{KL}\Big( p_\mathrm{raw}^{(t)} \,\|\, p_\mathrm{mem}^{(t)} \Big). \end{aligned} \tag{11}$$

The teacher is the frozen LLM with raw context under the same cap used in evaluation. For reconstruction, let $(c_1, \ldots, c_J)$ be the source tokens within the reconstruction cap, and let $r$ be a learned content-free slot. Memory is constructed for this source and prompt. The decoder receives $[M; r; \ldots; r]$ with $J$ slots under causal attention. If $h_j$ is the hidden state at slot $j$ and $E$ the tied token-embedding matrix, then

$$\mathcal{L}_\mathrm{recon} = -\frac{1}{J} \sum_{j=1}^{J} \log \left[ \mathrm{softmax}(E h_j) \right]_{c_j}. \tag{12}$$

Slot $j$ attends to memory and preceding content-free slots; its output predicts $c_j$ directly, without the next-token shift used for answer generation. Source tokens are never decoder inputs.

**Complementary retention objectives.** Distillation and reconstruction preserve different aspects of the context. With the underlying LLM knowledge fixed, distillation aligns the reader's answer distribution from latent memory with the frozen LLM's distribution from raw context. It preserves how the model uses the context for the prompt, including predictive information beyond the reference answer. Reconstruction instead requires memory to support reproduction of the source tokens. Its objective is broader information retention: preserve as much of the source as possible within the memory budget, rather than only what determines the current answer. Here reproduction is supervised through the content-free slots above, not by feeding source tokens back into the decoder.

These constraints are complementary. Different memories can produce similar answer distributions while omitting source details; reconstruction penalizes such omissions. Conversely, retaining reconstructable content does not by itself ensure that the reader uses it as effectively as raw context; distillation aligns that use. Together, the objectives encourage both source-information retention and consistency of the reader's predictions. This supports our division of work: the writer preserves informative states, while the reader learns to interpret them. The loss ablations in Figures 6 and 7 examine how these complementary constraints affect training stability and failure detection.

### A.3 Signals and calibration

The candidate model-derived signals include reader first-token confidence,

$$s_{\mathrm{conf}}(M, q) = \max_{v} p_{\mathrm{mem}}(v \mid M, q). \tag{13}$$

We also evaluate the probability margin (the difference between the two highest first-token probabilities), TARG's prompt-only uncertainty (Wang et al., 2026), a logistic probe, and a performance predictor, defined below. Signals distinguish whether compressed memory is at least as accurate as raw context under the stated cap; AUROC evaluates detection independently of the threshold.

For context $C$, prompt $q$, and reference answer $y$, let $\hat{y}_{\mathrm{mem}}$ and $\hat{y}_{\mathrm{raw}}$ denote the compressed-memory and raw-context predictions. Let $u(\hat{y}, y) \in [0, 1]$ be the task score; reported percentages rescale this quantity by 100. Compression failure is $u(\hat{y}_{\mathrm{mem}}, y) < u(\hat{y}_{\mathrm{raw}}, y)$. The positive label for the reliability score is $z = \mathbf{1}[u(\hat{y}_{\mathrm{mem}}, y) \geq u(\hat{y}_{\mathrm{raw}}, y)]$. Thus the failure target concerns relative task scores, not an annotation that identifies a particular missing fact.

Reference answers define evaluation labels and calibration scores, but are not inputs to the detection signal at inference. Ties are non-failures: the target compares task scores, not answer correctness.

Let $A_\tau$ indicate that the reader branch is selected. We constrain positive compression harm on accepted examples:

$$R_{\mathrm{harm}}(\tau) = \frac{\mathbb{E}[A_\tau \{u(\hat{y}_{\mathrm{raw}}, y) - u(\hat{y}_{\mathrm{mem}}, y)\}_+]}{\mathbb{E}[A_\tau]} \leq \epsilon, \tag{14}$$

where $\{a\}_+ = \max(a, 0)$. Score gains cannot offset losses when selecting the threshold.

To compare reader confidence with learned prediction, the learned baselines use inexpensive statistics available after memory construction: memory, prompt, and context lengths; the memory-to-context ratio; means, standard deviations, and maxima of memory and prompt state norms; mean and maximum prompt-memory cosine similarities; and the concentration of memory states around their mean direction. Within each calibration fold, documents are divided again into disjoint probe-fit and threshold-selection subsets. Features are standardized using only the probe-fit subset and clipped to $[-20, 20]$. The logistic probe is an $\ell_2$-regularized logistic regressor for the event that raw outperforms memory. The performance predictor is an $\ell_2$-regularized linear regressor for the signed task-score gap

$$g(x) = u(\hat{y}_{\mathrm{raw}}, y) - u(\hat{y}_{\mathrm{mem}}, y), \qquad s_{\mathrm{pred}}(x) = -\hat{g}(x), \tag{15}$$

so a larger gating score predicts safer compression. Neither learned baseline is part of the compressor checkpoint; both are refit from calibration data for each split.

All candidate signals are evaluated through document-disjoint five-fold cross-validation. In each fold, the signal and threshold are selected on four folds and then applied unchanged to the held-out fold. Threshold selection minimizes fallback ratio subject to the positive-harm constraint in Equation 14. A separate document-level Learn-then-Test analysis evaluates a fixed threshold family (Angelopoulos et al., 2025); if it certifies no nonzero coverage, the formal policy returns all raw.

Table 5: Held-out gating signals on predictor-enabled runs. Values in the five signal columns are compression-failure AUROC on a 0-1 scale. "Fallback ratio" is the fraction of examples answered using raw context by the performance predictor, and Δ raw is its gated-score change relative to raw context on the 0-1 task-score scale. Each entry averages 20 document-disjoint splits.

| reader | task | confidence | margin | TARG | logistic probe | predictor | Fallback ratio (fraction) | Δ raw (score) |
|---|---|---|---|---|---|---|---|---|
| Qwen3-8B | BFCL | **0.827** | 0.771 | 0.508 | 0.615 | 0.622 | 0.724 | −0.041 |
| Qwen3-8B | BFCL | **0.836** | 0.794 | 0.480 | 0.568 | 0.574 | 0.882 | −0.024 |
| Qwen3-8B | BFCL | **0.811** | 0.682 | 0.490 | 0.621 | 0.592 | 0.740 | −0.052 |
| Qwen3-8B | HotpotQA | **0.630** | 0.595 | 0.585 | 0.547 | 0.553 | 0.838 | −0.020 |
| Qwen3-8B | SQuAD v2 | **0.665** | 0.623 | 0.519 | 0.531 | 0.506 | 0.945 | −0.021 |
| Qwen3.5-9B | BFCL | **0.849** | 0.811 | 0.541 | 0.562 | 0.583 | 0.496 | −0.055 |
| Qwen3.5-9B | BFCL | **0.848** | 0.836 | 0.551 | 0.536 | 0.594 | 0.586 | −0.050 |
| Qwen3.5-9B | HotpotQA | **0.672** | 0.624 | 0.588 | 0.583 | 0.592 | 0.677 | −0.023 |
| Qwen3.5-9B | HotpotQA | **0.678** | 0.641 | 0.592 | 0.589 | 0.591 | 0.693 | −0.023 |
| Qwen3.5-9B | SQuAD v2 | **0.620** | 0.593 | 0.544 | 0.514 | 0.517 | 0.943 | −0.019 |
| Qwen3.5-9B | SQuAD v2 | **0.625** | 0.598 | 0.543 | 0.504 | 0.496 | 0.945 | −0.021 |
| Qwen3-8B | QuALITY | **0.622** | 0.617 | 0.596 | 0.512 | 0.505 | 0.917 | −0.023 |
| Qwen3-8B | QuALITY | **0.629** | 0.611 | 0.592 | 0.510 | 0.505 | 0.900 | −0.025 |
| Qwen3-8B | QuALITY | **0.634** | 0.624 | 0.575 | 0.516 | 0.507 | 0.884 | −0.026 |
| Qwen3.5-9B | QuALITY | 0.610 | **0.614** | 0.562 | 0.504 | 0.506 | 0.926 | −0.027 |
| Qwen3.5-9B | QuALITY | **0.610** | 0.607 | 0.558 | 0.501 | 0.497 | 0.924 | −0.031 |
| Qwen3.5-9B | QuALITY | 0.588 | **0.599** | 0.568 | 0.493 | 0.491 | 0.920 | −0.028 |

### A.4 Gating-signal selection and learned predictors

**Comparison design.** Table 5 compares first-token confidence, margin, TARG, the logistic probe, and the performance predictor. The probe predicts the compression-failure event; the predictor estimates the signed raw-minus-compressed task-score gap. Both use the state and length features defined above and are fitted separately within each document-disjoint split. This tests whether extra supervised prediction improves on reader confidence.

**Reader confidence versus learned prediction.** First-token confidence achieves the highest AUROC in 15 of 17 runs; margin leads by only $0.004$ and $0.011$ in the remaining two. On BFCL, confidence reaches $0.811$-$0.849$, compared with $0.574$-$0.622$ for the performance predictor. These results support joint writer-reader training as a source of useful failure information: the reader's output distribution is more informative here than a separate predictor built from aggregate state statistics.

**Boundary of feature-based prediction.** The performance predictor approaches chance on SQuAD v2 and QuALITY and produces negative gated-score changes in every reported run. Its features summarize state norms, similarities, and input lengths, but do not explicitly test whether prompt-relevant evidence survives. This suggests a limitation of the tested feature-based predictors, not of learned failure detection in general. The comparison favors first-token confidence as the default while identifying where a more evidence-sensitive predictor would need to improve.

### A.5 Cost accounting

The recurrent writer still scans the available source, and prompt-conditioned memory generally must be rebuilt for a new prompt. We therefore report writer latency, compressed-reader latency, memory length, and incremental peak allocation separately, since reader input length alone cannot establish whether the complete system is faster.

**Recurrent writing.** Let $B$ bound the chunk length, $P$ be the prompt length, $d$ the hidden width, and $D_{\mathrm{w}}$ the number of executed writer layers. At chunk $i$, the concatenated-memory writer processes the chunk, the prompt, the $(i-1)K$ previous states, and $K$ new latent prompts. Its input length is

$n_i \leq B + P + iK$. Under dense attention, writing costs

$$C_{\text{write}} = O\left( D_{\text{w}} \sum_{i=1}^{S} [n_i d^2 + n_i^2 d] \right). \tag{16}$$

With $A = B + P$, this is $O(D_{\text{w}}[d^2(SA + KS^2) + d(SA^2 + AKS^2 + K^2S^3)])$. The recurrent prefix is recomputed rather than reused as a KV cache; detaching it limits gradient propagation, not forward sequence length. The final latent memory stores $SKd$ values. Attention workspace depends on the implementation; the quadratic attention term above applies to dense rather than linear attention.

**Reading and detection.** For $L$ reader layers and $m = SK + P$ input positions, dense-attention prefill costs $O(L[md^2 + m^2 d])$. The confidence score is obtained from this first output distribution, so detection needs no draft-answer decoding but still requires memory construction and reader prefill. Raw-context fallback adds a raw-context reader pass. Consequently, end-to-end cost depends on both the number of chunks and the fallback ratio; reducing reader input alone does not determine the total latency or peak allocation.

## B Design principles and tradeoffs

Our central principle is to preserve diverse, informative representations without prematurely resolving unfamiliar patterns. Intermediate states provide the starting point; a simple writer and information-retention losses aim to retain their useful distinctions within a limited memory budget. Interpretation remains the reader's responsibility. We hypothesize that preserving these distinctions lets reader confidence reflect whether it has learned to handle the retained patterns. Familiarity here means being able to interpret a pattern, not having seen the exact training example.

**Preserve informative variation, not arbitrary uncertainty.** The high-entropy intuition concerns retaining distinctions in the context that could matter for answering, instead of forcing them prematurely into a single textual account. Increasing variation through noise would not serve this purpose. Memory must retain semantic information and depend on the prompt. GCM implements this preference through prompt-conditioned latent states, task supervision, and auxiliary information-retention objectives, rather than an explicit entropy-maximization loss. This places the emphasis on information available to the reader, not the entropy of the representation in isolation.

**Use a simple writer and retain intermediate representations.** The information bottleneck view describes representations through the input information they retain and its relevance to prediction (Tishby & Zaslavsky, 2015). Empirical information-plane analyses observe compression of input information while predictive information is retained (Shwartz-Ziv & Tishby, 2017). These analyses motivate our hypothesis that intermediate states may retain more diverse information than representations shaped by the final output-oriented transformations. Our interest is the diversity of information still available for interpretation, rather than an already resolved answer. GCM therefore extracts states halfway through a frozen LLM and applies a small projection instead of generating a textual summary. The writer reduces the number of states without asking a separate generator to explain away unfamiliar structure. This trades additional writer processing for greater reliance on the reader's learned interpretation; half depth is our practical implementation of that choice.

**Intermediate versus final-layer extraction.** Table 6 reports exploratory BFCL Live Multiple runs using intermediate states at half the LLM depth or states from its final layer. Intermediate extraction attains higher compression-only accuracy for both readers, by 2.08 and 3.65 points, while reducing writer computation. For the same input, half-depth extraction executes only the first half of the LLM layers; the reader computation remains unchanged. These runs therefore favor a practical accuracy-computation tradeoff: the shallower writer produces better compressed answers while avoiding the remaining writer layers. This supports leaving interpretation to the reader rather than increasing writer depth alone. Only writer computation is reduced; total latency remains unmeasured.

Confidence remains informative at both depths, with slightly higher AUROC for final-layer extraction. The evidence thus concerns the viability of a shallower writer, rather than a detection advantage of

Table 6: Exploratory writer-depth comparison on BFCL Live Multiple. Accuracy is reported in percent; AUROC uses first-token confidence. Each row includes its raw-context reference; run configurations are described in Appendix H.

| Reader | Writer states | GCM w/o gating (ours, %) | Confidence AUROC | Raw context (%) |
|---|---|---|---|---|
| Qwen3-8B | Intermediate (half depth) | 73.96 | 0.763 | 91.67 |
| Qwen3-8B | Final layer (full depth) | 71.88 | 0.802 | 93.75 |
| Qwen3.5-9B | Intermediate (half depth) | 70.83 | 0.796 | 83.33 |
| Qwen3.5-9B | Final layer (full depth) | 67.19 | 0.804 | 87.50 |

intermediate states. Representation diversity motivates the design as a hypothesis; this comparison measures answer accuracy and detection AUROC.

The proposed connection between representation diversity and detection is that premature interpretation can hide difficulty. If compression turns an unfamiliar pattern into a familiar but incorrect account, the reader may confidently continue that account. Preserving the pattern instead leaves its interpretation to the reader, where difficulty can appear as uncertainty. This is the mechanism we seek to encourage through intermediate-state extraction and limited writer processing. Projection normalization matches the LLM's embedding scale, so the intended diversity concerns information content rather than merely larger activations.

**Condition retention on the prompt, but learn how to read it.** Latent prompts $Q$ follow the context and user prompt, allowing their states to collect relevant information. These learned vectors provide a shared mechanism for writing across examples; they are not explicit labels for particular failure types. We hypothesize that recurring difficulties in interpreting the resulting states can yield reusable detection signals. The task-transfer and alternative-writer experiments test consequences of this hypothesis without assigning individual latent prompts to failure modes. Prompt conditioning improves relevance at the cost of rebuilding memory when the prompt changes. The budget ablation suggests that retaining more states helps only when the reader can integrate them (Table 7).

**Train for useful answers without assuming detection follows.** Task gradients connect retained information to the reader's answers. Distillation preserves predictive information from the raw-context LLM, while reconstruction supervises content beyond the target answer. Together, these losses aim to prevent the representation from being shaped only by the answer label, preserving information that the reader must learn to use. The writer's simplicity and the retention objectives thus serve the same purpose: preserve useful structure without requiring the writer to interpret every pattern it encounters. We expect the reader to remain more uncertain when it has not learned to handle the retained pattern, making confidence useful for detection. The stability and detection ablations provide evidence for these training choices (Figures 6 and 7); transfer experiments test whether the resulting signal extends beyond the training task. They assess the usefulness of the signal, rather than directly measuring familiarity with individual patterns.

**Assess reliability early and leave correction open.** First-token confidence uses the reader's initial prediction, avoiding full answer generation or a separate verifier. This accepts a tradeoff: an early signal cannot inspect errors that become apparent only later in the answer. Transfer AUROC and the signal comparisons evaluate when that low-cost choice is useful (Appendix G.1 and Appendix A.4). Raw-context fallback is a simple evaluation policy; retrieval, memory revision, and verification remain alternative uses of the same signal. The central principle is to make compressed memory both useful and assessable, without tying detection to one correction policy.

## C Additional compression-budget ablation

**Compression-budget ablation.** Table 7 favors $K = 128$ over both smaller and larger budgets. This non-monotonic response is consistent with the writer-reader division in Section 2: retaining more states is useful only if the reader can integrate their information. A smaller budget can remove relevant

Table 7: Compression-budget ablation on Qwen3-8B BFCL (%; mean ± standard deviation).

| $K$ | GCM w/o gating (ours) |
|---|---|
| 64 | 69.6±1.9 |
| 128 | **72.3±0.5** |
| 256 | 70.9±1.5 |

Table 8: Held-out gating on BFCL and HotpotQA. Task scores and fallback ratios are percentages; AUROC is on a 0-1 scale and Δ raw is in F1 or accuracy points.

| Qwen3-8B | BFCL | HotpotQA | Qwen3.5-9B | BFCL | HotpotQA |
|---|---|---|---|---|---|
| Raw context | **92.10** | 53.10 | Raw context | 84.50 | 53.90 |
| GCM w/o gating (ours) | 72.20±0.50 | 28.80±0.40 | GCM w/o gating (ours) | 72.10±0.70 | 30.50±0.30 |
| GCM w/ gating (ours) | 91.70±0.40 | **53.60**±0.10 | GCM w/ gating (ours) | **85.80**±0.30 | **54.10**±0.10 |
| AUROC | 0.54±0.29 | 0.59±0.00 | AUROC | 0.84±0.01 | 0.59±0.00 |
| Fallback ratio (%) | 95.00±5.00 | 91.80±0.60 | Fallback ratio (%) | 52.30±2.80 | 90.70±0.60 |
| Δ raw (pp) | $-0.40 \pm 0.40$ | $+0.40 \pm 0.10$ | Δ raw (pp) | $+1.30 \pm 0.30$ | $+0.20 \pm 0.10$ |

evidence; a larger one may distribute it across more states without improving its accessibility. This offers a possible explanation for the decline at $K = 256$ under the evaluated training configuration. Under this configuration, $K = 128$ balances information retention with a concentrated reader input, consistent with the design principle in Appendix B.

# D Additional gating results

## D.1 HotpotQA evaluation protocols

**LongBench context budgets.** For both LB-HotpotQA and LB-2WikiMQA in Table 2, the no-context reference uses zero context tokens. Truncation, BM25, and LLMLingua use 128 tokens; GCM without gating uses $128S$ latent states for $S$ chunks. Raw-context references are capped at 16K or 32K tokens as indicated by their row labels. GCM with gating uses $128S$ states when accepting compressed memory and a 16K-token raw-context cap otherwise. These budgets exclude the prompt.

The HotpotQA evaluation in Table 3 uses a 4,096-token raw-context window and evaluates gating across readers. LB-HotpotQA in Table 2 evaluates the HotpotQA-trained writer and reader on 200 LongBench examples without target-specific tuning, using a 16,384-token raw-context window; 32,768-token raw-context references are reported separately. LB-2WikiMQA uses the multi-hop QA dataset of Ho et al. (2020). GCM uses 128 states per 4,096-token chunk and passes all chunks' states to the reader, so longer inputs can also increase the compressed-context budget.

Both evaluations report answer F1 on a 0-100 scale. LongBench scoring extracts the first nonempty answer line, including a leading reasoning marker when present. This extracted text determines both answer F1 and the paired score comparison used to label compression failures.

## D.2 Additional reader comparisons

Table 9 extends the reader and task comparison, showing both successful gated recovery and settings where compression or detection limits the benefit.

**Recovery across readers and tasks.** Gating improves HotpotQA performance for five readers by 0.95-6.16 points. On NarrativeQA, compression alone exceeds raw context for both readers, and gating raises the gains to 3.85 and 5.76 points. The framework can therefore benefit from both useful compressed answers and detection-guided correction, with their contributions varying by task.

**Limits of gated recovery.** Ministral on HotpotQA and both SQuAD v2 readers use raw context for more than 99% of examples, with small residual losses. Here, recovery does not establish useful compression savings. Conversely, NarrativeQA's near-chance AUROC (0.531 and 0.511) shows that

Table 9: Additional gating results. Task scores use a 0-100 scale; fallback ratio is a percentage and AUROC is on 0-1. Gated scores and Δ raw use continuous task scores and five-fold calibration.

| reader | task | GCM (ours) w/o gating | Raw context | GCM (ours) w/ gating | Δ raw (pp) | Fallback ratio (%) | AUROC |
|---|---|---|---|---|---|---|---|
| Qwen3-8B | HotpotQA | 31.04 | 53.11 | 54.06 | +0.95 | 78.4 | 0.661 |
| Qwen3.5-9B | HotpotQA | 30.44 | 53.86 | 56.01 | +2.15 | 80.2 | 0.675 |
| GLM-4-9B | HotpotQA | 32.87 | 53.66 | 55.09 | +1.42 | 74.5 | 0.649 |
| Ministral-8B | HotpotQA | 22.71 | 64.24 | 64.23 | −0.01 | 99.9 | 0.632 |
| xLAM-8B | HotpotQA | 32.71 | 44.80 | 50.95 | +6.16 | 60.4 | 0.656 |
| ToolACE-8B | HotpotQA | 31.15 | 46.95 | 51.92 | +4.98 | 68.2 | 0.640 |
| Qwen3-8B | SQuAD v2 | 26.13 | 64.81 | 64.75 | −0.06 | 99.3 | 0.610 |
| Qwen3.5-9B | SQuAD v2 | 24.20 | 66.74 | 66.62 | −0.11 | 99.7 | 0.687 |
| Qwen3-8B | NarrativeQA | 16.92 | 14.24 | 18.09 | +3.85 | 31.4 | 0.531 |
| Qwen3.5-9B | NarrativeQA | 17.68 | 13.99 | 19.75 | +5.76 | 38.3 | 0.511 |

Table 10: SQuAD v2 diagnostic scores (%). Non-gated rows report F1; gated rows use an in-sample threshold sweep and a binarized score. Table 11 reports held-out gating on the common F1 scale.

| reader | method | Score (0-100) |
|---|---|---|
| Qwen3-8B | No context | 16.1 |
| Qwen3-8B | Raw context | 65.5 |
| Qwen3-8B | SFT reference | 93.1±0.3 |
| Qwen3-8B | Truncation | 49.6 |
| Qwen3-8B | GCM w/o gating (ours) | 26.5±0.4 |
| Qwen3-8B | GCM w/ gating (ours) | **62.8** |
| Qwen3.5-9B | No context | 20.7 |
| Qwen3.5-9B | Raw context | 66.8 |
| Qwen3.5-9B | SFT reference | 93.8±0.3 |
| Qwen3.5-9B | Truncation | 49.6 |
| Qwen3.5-9B | GCM w/o gating (ours) | 26.9±0.6 |
| Qwen3.5-9B | GCM w/ gating (ours) | **64.7** |

Table 11: SQuAD v2 held-out gating under the same five-fold cross-validated, continuous-scale protocol as Table 8 (%; three runs). The gate selects raw context for more than 99% of examples.

| reader | Raw context | GCM (ours) w/o gating | GCM (ours) w/ gating | fallback ratio (%) | Δ raw (pp) |
|---|---|---|---|---|---|
| Qwen3-8B | **64.9** | 26.4 | 64.8 | 99.8 | −0.04 |
| Qwen3.5-9B | **66.7** | 26.8 | **66.7** | 99.7 | −0.02 |

improved task scores alone do not establish reliable failure detection. These contrasts separate the boundaries of compression quality, detection, and correction.

## E SHORT-PASSAGE DIAGNOSTIC

SQuAD v2 (Rajpurkar et al., 2018) tests compressed memory on short passages requiring exact-text answers. The supervised fine-tuning (SFT) reference adapts the LLM to the task.

The held-out evaluation in Table 11 uses continuous F1 for both branches. Gating selects raw context on more than 99% of prompts, leaving deficits of 0.04 and 0.02 points for the two readers. Recovery here comes from bypassing compressed memory, with almost no use of the reader branch. The binarized, in-sample gated scores in Table 10 report binarized F1 with thresholds evaluated on the calibration examples.

Table 12: Qwen3.5 LLM-scale ablation on BFCL (%; mean ± standard deviation over three runs). Both scales use the same GCM training configuration. Grey type marks no-context and raw-context references; the best value in each column is bold.

| reader | No context (%) | Raw context (%) | GCM (ours) w/o gating |
|---|---|---|---|
| Qwen3.5-4B | 0.3 | 11.1 | **71.5±1.7** |
| Qwen3.5-9B | **1.3** | **84.5** | 71.4±1.9 |

Table 13: Training source: HotpotQA; evaluation targets: LongBench (first-line F1 %). Δ is GCM w/o gating minus raw context in F1 points. Bold marks the better result for each reader.

| | | Qwen3-8B | | | Qwen3.5-9B | | |
|---|---|---|---|---|---|---|---|
| target | answer | Raw context | GCM (ours) w/o gating | Δ (F1) | Raw context | GCM (ours) w/o gating | Δ (F1) |
| LB-TriviaQA | short entity | 74.3 | **89.7** | +15.4 | 78.9 | **88.1** | +9.3 |
| LB-PassageCount | a count | 0.6 | **3.0** | +2.4 | **3.3** | 3.0 | -0.3 |
| LB-PassageRetrieval | a passage index | **0.3** | 0.1 | -0.2 | **8.1** | 0.0 | -8.1 |
| LB-QMSum | a summary | **26.6** | 14.5 | -12.1 | **19.8** | 10.6 | -9.2 |

# F LLM-SCALE COMPARISON

**Compression performance across scales.** GCM achieves $71.5 \pm 1.7\%$ and $71.4 \pm 1.9\%$ on the 4B and 9B readers, respectively (Table 12), despite a large difference in their raw-context scores. The same training configuration thus produces comparable compression performance at both scales. This supports the training configuration's applicability beyond one model size, without implying that scaling necessarily improves the compressed reader.

# G ADDITIONAL CONTROLLED RESULTS

## G.1 CROSS-TASK FAILURE-DETECTION TRANSFER

The HotpotQA-trained writer and reader are frozen for evaluation on LongBench. First-token confidence is evaluated on 200 examples per target, using the relative task scores from compressed memory and raw context. AUROC is reported below:

| reader | LB-HotpotQA | LB-2WikiMQA |
|---|---|---|
| Qwen3-8B | 0.674 | 0.626 |
| Qwen3.5-9B | 0.717 | 0.491 |

**Transferable structure and its boundary.** Without target-task retraining, Qwen3-8B retains AUROC $0.674$ on LB-HotpotQA and $0.626$ on LB-2WikiMQA, while Qwen3.5-9B reaches $0.717$ on LB-HotpotQA. This supports the hypothesis that compression failures can share detectable structure across tasks. Qwen3.5-9B's $0.491$ on LB-2WikiMQA identifies a reader-task boundary: the shared structure is not sufficient for every writer and reader. This tests detection transfer, not threshold reuse without recalibration on the target task.

## G.2 TASK-FORMAT TRANSFER CONTROLS

This diagnostic applies the HotpotQA short-answer evaluation protocol to every target: the first non-empty generated line is scored with normalized token F1. It tests transfer under a fixed answer protocol, not the official LongBench summarization or retrieval metrics. In particular, the LB-QMSum scores do not measure the quality of a complete generated summary.

**Information requirements constrain transfer.** Table 13 shows gains of $15.4$ and $9.3$ F1 points on LB-TriviaQA without target training, supporting transfer of useful compressed evidence to another

Table 14: HotpotQA training-data-budget ablation (%). The 2,000-example rows are means over three runs; 90,447-example rows are single runs. Fallback ratio is the percentage answered using raw context; AUROC is on a 0-1 scale.

| reader | training setting | train examples | GCM (ours) w/o gating | GCM (ours) w/ gating | Δ raw (pp) | Fallback ratio (%) | AUROC |
|---|---|---|---|---|---|---|---|
| Qwen3-8B | 2k, 3 epochs | 2,000 | 28.80±0.40 | 53.60±0.10 | $+0.40 \pm 0.10$ | 91.8 | 0.590 |
| Qwen3-8B | 1 epoch | 90,447 | 31.04 | 54.06 | $+0.95$ | 78.4 | 0.661 |
| Qwen3.5-9B | 2k, 3 epochs | 2,000 | 30.50±0.30 | 54.10±0.10 | $+0.20 \pm 0.10$ | 90.7 | 0.590 |
| Qwen3.5-9B | 1 epoch | 90,447 | 30.44 | 56.01 | $+2.15$ | 80.2 | 0.675 |

short-answer task. Passage retrieval and meeting summarization instead require positional fidelity or broader information coverage. Their losses delineate what the source-trained writer preserves. The low passage-count scores for both raw and compressed inputs further motivate preserving document structure or using external tools.

### G.3 Controlled held-out gating

Table 8 reports how held-out gating behaves across Qwen3-8B and Qwen3.5-9B on the controlled BFCL and HotpotQA experiments. Unlike Figure 3, which evaluates the confidence-based detection, this comparison emphasizes the policy induced by that signal: fallback ratio and downstream task scores. No test item is used to select its own signal or threshold.

Only Qwen3.5-9B on BFCL combines reduced fallback with an improvement over raw context: its fallback ratio is $52.3\%$ and it gains $1.3$ points. The remaining cells expose two boundary cases. On HotpotQA, both models require $90.7$-$91.8\%$ fallback to approach raw-context performance, so the reader branch is rarely used. On Qwen3-8B BFCL, fallback rises to $95.0\%$ and the gated score remains slightly below raw despite the stronger result of Qwen3.5-9B on the same task. Cross-model variation therefore matters: detection quality alone is insufficient unless the reader branch is useful and the chosen threshold permits a sufficiently low fallback ratio.

### G.4 Training-data budget

We compare repeated training on a smaller dataset with a single pass over 90,447 HotpotQA training examples. Table 14 keeps the architecture, reader budget, and held-out gating protocol fixed and reports each training setting separately.

**Data improves detectability beyond compression-only accuracy.** In Table 14, training on more examples raises AUROC from $0.590$ to $0.661$ and $0.675$, while reducing fallback for both readers. For Qwen3.5-9B, compression-only accuracy is nearly unchanged ($30.50$ versus $30.44$), yet the gated gain over raw increases from $0.20$ to $2.15$ points. This separates improved detection from improved average compression and supports evaluating both objectives. Training sizes, passes, and run counts are specified in Table 14.

### G.5 Systems measurements

We measure per-prompt cost on one H100 over 20 items per profile, reporting CUDA-event median latency and incremental read-phase peak allocation. Table 15 separates writing, reading compressed memory, and reading raw context phases.

**Reader savings and end-to-end limits.** On QuALITY, the compressed read takes $82.0$ versus $200.1$ ms on Qwen3-8B and $443.3$ versus $1052.1$ ms on Qwen3.5-9B (Table 15). Peak read-phase allocation falls from $996.2$ to $74.0$ MiB and from $1051.1$ to $125.4$ MiB, respectively. These measurements establish a concrete benefit of compressed reader inputs. However, encoding makes total latency higher than raw reading in these profiles. End-to-end efficiency therefore remains conditional on memory construction cost and fallback ratio, rather than an automatic consequence of compression.

Table 15: Systems measurements per prompt. Values are medians; peak allocation reports compressed/raw reader memory. Measurement details are in Appendix H.

| reader | task | source tokens | memory states | ratio | encode (ms) | memory read (ms) | total (ms) | raw read (ms) | peak (MiB) |
|---|---|---|---|---|---|---|---|---|---|
| Qwen3-8B | BFCL | 167.5 | 128 | 1.3× | 33.7 | 76.3 | 110.1 | 47.2 | 49.5/59.8 |
| Qwen3.5-9B | BFCL | 170.0 | 128 | 1.3× | 249.4 | 439.3 | 688.7 | 424.5 | 100.0/113.7 |
| Qwen3-8B | QuALITY | 4,141.0 | 192 | 24.0× | 165.7 | 82.0 | 247.7 | 200.1 | 74.0/996.2 |
| Qwen3.5-9B | QuALITY | 4,239.0 | 192 | 23.0× | 746.4 | 443.3 | 1189.7 | 1052.1 | 125.4/1051.1 |

## H Training recipe and inference configuration

**Evaluation inputs and scoring.** Raw-context caps are 4,096 tokens for BFCL, HotpotQA, and SQuAD v2; 8,192 for NarrativeQA and QuALITY; and 16,384 for LongBench, with 32,768-token raw-context references also reported. Caps are chosen by dataset input lengths. Context budgets exclude the prompt. The default writer uses $K = 128$ states per 4,096-token chunk, and the main comparison uses $S = 1$. Some auxiliary evaluations use multiple chunks to accommodate longer inputs under their context caps. LongBench textual baselines return 128 tokens; GCM passes $128S$ states to its reader. BFCL uses accuracy, NarrativeQA uses ROUGE-L, and QA uses answer F1, reported on a 0–100 scale. The task-format diagnostic uses first-line F1. Signal selection and threshold calibration use four folds, with unchanged evaluation on the fifth. Controlled multi-run results report the mean and standard deviation.

**Additional compression baselines.** COCOM, PISCO, and OSCAR are adapted to Qwen3-8B with 128 latent states using their released model designs. COCOM shares the 8B model between compression and generation with rank-16 adapters. PISCO and OSCAR use a Qwen3-0.6B writer, a two-layer projection, and rank-64 decoder adapters; OSCAR additionally conditions compression on the prompt. Each adaptation uses 32 warmup and 250 answer-training updates, a learning rate of $3 \times 10^{-4}$, and eight gradient-accumulation steps. BFCL uses 735 training examples; HotpotQA and NarrativeQA each use 2,000. COCOM uses reference answers, whereas PISCO and OSCAR use outputs from the frozen raw-context reader for answer supervision. These runs use the exported main-table inputs and scoring functions, with 64-token greedy decoding; they train the adapted modules rather than use the released Mistral weights.

ICAE uses Qwen3-8B with 128 latent states, 32 reconstruction updates and 250 answer-supervision updates. DAC uses the official dynamic attention–perplexity compressor with a Qwen2.5-0.5B-Instruct scoring model, ten compression iterations and attention weight 0.8. Its output is retokenized and capped at 128 reader tokens. CriticalKV uses the KVPress implementation with SnapKV scores and the default two-stage selection, retaining 128 KV positions per head. LagKV uses the KVPress implementation with lag size 128 and a budget of 128 KV positions per head. Budgets count reader input tokens, latent states, or retained KV positions per head, respectively. All four use the exported main-table inputs and scoring functions. The new evaluations use greedy decoding with a 64-token output limit. DAC, LagKV, and CriticalKV recompute paired raw-context answers; their BFCL reference is 92.7; the original main-table raw-context evaluation reports 92.1. GCM uses a trained writer and rank-64 reader adapter; DAC and the KV-cache baselines use the frozen reader. Each LLM has its own trained GCM writer and reader, which remain fixed during cross-task evaluation. The writer-depth comparison reports each run's raw-context reference alongside its compressed score and AUROC. The training-data comparison specifies the number of examples, training passes, and evaluation runs for each configuration.

**Training configuration.** Table 16 summarizes the default configuration. We keep the LLM frozen and optimize the latent prompts, projection, and rank-64 reader adapter with AdamW (Loshchilov & Hutter, 2019) at a learning rate of $3 \times 10^{-4}$, using eight gradient-accumulation steps. Task supervision is combined with distillation and reconstruction, each weighted by $0.5$; reconstruction is capped at 512 tokens. Training-set sizes and numbers of passes are specified separately in Appendix G.4.

**Models and auxiliary evaluations.** The readers are Qwen3-8B (Qwen Team, 2025), Qwen3.5-9B (Qwen Team, 2026), GLM-4-9B-0414 (Team GLM et al., 2024), Ministral-8B-Instruct-2410

Table 16: Training and inference configuration.

| item | setting | item | setting |
|---|---|---|---|
| chunk size | 4,096 tokens | compression budget | $K = 128$ states/chunk |
| writer depth | half depth | projection | $d \to 2d \to d$ |
| normalization | embedding norm | reader adapter | rank 64 |
| distillation | 0.5 | reconstruction | 0.5, cap 512 |
| optimizer | AdamW | learning rate | $3 \times 10^{-4}$ |
| batch | $1 \times 8$ accumulation | training | 1 epoch |
| runs | 3 independent runs | gate calibration | 5-fold CV |
| harm target | 2% | family-wise error level | 10% |

Table 17: GCM module ablations on Qwen3-8B BFCL (%; mean $\pm$ standard deviation).

| variant | BFCL (%) |
|---|---|
| w/o writer task gradient | 71.4±2.9 |
| w/o distillation | 71.8±1.8 |
| w/o reconstruction | 72.2±2.0 |
| w/o recurrence | 72.2±1.6 |
| GCM w/o gating (ours) | **72.3±0.5** |

(Mistral AI Team, 2024), Llama-xLAM-2-8B-FC-R (Prabhakar et al., 2025), and ToolACE-2-8B (Liu et al., 2025); tables use shortened names. The matched-writer evaluation also uses Glaive Function Calling V2 (Glaive AI, 2023). QuALITY (Pang et al., 2022) supplies long-document multiple-choice questions. Auxiliary transfer diagnostics use LB-TriviaQA (Joshi et al., 2017), LB-QMSum (Zhong et al., 2021), LB-PassageCount, and LB-PassageRetrieval under the LongBench protocol (Bai et al., 2024).

**Inference configuration.** The writer processes 4,096-token chunks and produces $K = 128$ states per chunk. The reader consumes their concatenation with the prompt and supplies the first-token confidence used for gating. When the gate selects raw context, we disable the reader adapter and use the same frozen LLM. Dataset-specific context caps are listed above; the held-out gating protocol is detailed in Appendix G.

## I Additional module and loss ablations

Removing recurrence changes the mean from 72.3 to 72.2% but raises standard deviation from 0.5 to 1.6 points. Together with the loss removals, this supports complementary stabilization rather than a large gain from any single component.

## J Limitations and Future Work

**Extending detection across tasks.** This paper focuses on low-overhead detection from the reader's initial prediction. Extending this signal to a broader range of tasks is a natural next step: first-token confidence measures predictive certainty, and can remain high after evidence is lost. The near-chance Qwen3.5 result on LB-2WikiMQA (Appendix G.1) motivates studying which task properties affect transfer and when later-generation signals add value.

**From detection to efficient correction.** We use raw-context fallback to evaluate a direct application of detection. Optimizing the correction policy for end-to-end efficiency is beyond this study's focus. Frequent fallback and prompt-conditioned memory construction can offset savings in reader computation. Future work can investigate targeted retrieval, memory revision, and reuse across related prompts, evaluating latency jointly with answer quality and detection AUROC.

**Separating representation and reader adaptation.** Our comparisons evaluate the complete writer-reader design, with model configurations specified in Appendix H. Transferring a fixed writer

between LLMs extends the cross-task reuse evaluated here. LongBench scores also depend on answer extraction. Evaluation that jointly measures retained evidence and full answers would separate representation quality from response-format effects.

# K Further differences from related work

## K.1 Differences in the detection problem

**Reliability under compression.** Sufficient Context and AttnComp examine evidence sufficiency or retrieval reliability (Joren et al., 2025; Luo et al., 2025). Our paired comparison instead asks whether compression lowers performance relative to raw context for the same prompt. Compression-Aware Abstention learns refusal from evidence-survival labels (Khodabandehlou & Krishnamachari, 2026); GCM uses task, distillation, and reconstruction supervision while leaving correction open. Overflow detection (Belikova et al., 2026) and first-token adapter activation (Zheng et al., 2026) provide related detection mechanisms. GCM focuses on designing the compressed representation and reader together so that useful compression and low-overhead failure detection coexist. The following comparisons distinguish the targets, supervision, and uses of these signals.

**Detection as a complementary objective.** Reliable compression can involve preserving more evidence, identifying unreliable memory, or deciding how to respond to it. We focus on identifying failure: whether compressed memory lowers answer quality relative to raw context for the same prompt. Our design question is how the writer and reader can preserve useful information while making such failures visible in the reader's uncertainty. Better compression can reduce failures without removing the need to detect those that remain. A detection signal can in turn inform abstention, retrieval, memory revision, or raw-context fallback. In this sense, detection is orthogonal to the choice of correction policy and complementary to many compression improvements. The Cartridges experiment tests this compatibility with a different writer; detection quality depends on both the writer and the task.

**Context sufficiency and answer reliability.** Sufficient Context (Joren et al., 2025) separates whether the supplied context contains enough evidence from whether a model can use it correctly. GCM asks a comparative question: does replacing raw context with compressed memory harm the answer? We compare scores from compressed memory and raw context, not an oracle annotation of evidence sufficiency. A lower memory score can reflect information loss or difficulty reading the representation; the signal does not identify which fact was lost. AttnComp (Luo et al., 2025) combines extractive compression with a confidence estimate based on attention to the instruction. Its score reflects retrieval relevance and is evaluated against answer quality. Our first-token signal is evaluated against the score decrease caused by replacing raw context, rather than the answer's absolute quality or difficulty.

**Compression-aware abstention.** Compression-Aware Abstention (Khodabandehlou & Krishnamachari, 2026) directly trains answering and refusal using labels derived from compressor survival masks and answer-bearing spans. It therefore shares our concern that compression can remove evidence without preventing a confident answer. The main difference is supervision: GCM trains a latent writer and reader with task, distillation, and reconstruction objectives, then uses the reader's output distribution to detect failures. It does not require evidence-survival labels or refusal targets. The resulting failure-detection score can support refusal or other correction policies.

**Memory activation and generation.** Context Distillation as Latent Memory Management (Zheng et al., 2026) stores contexts in separate LoRA adapters and uses first-token entropy to gate their activation, establishing a precedent for low-cost first-token memory gating. Its activation decision distinguishes prompts suited to a context-specific adapter from context-agnostic prompts. GCM instead asks whether a prompt-conditioned compressed state damages performance relative to raw input. Whereas these methods decide how to use existing memory, FlashMem (Hou et al., 2026) uses attention entropy to decide when to synthesize latent memory from the LLM's cached states. Its monitor precedes memory generation; ours evaluates the result of compression through the memory reader's first-token distribution.

**Failure prediction and selective prediction.** Soft-representation overflow (Belikova et al., 2026) already studies raw-success/compressed-failure events. PoC models sample-specific performance-compression curves (Zhao et al., 2026), while TARG gates retrieval using uncertainty from a short no-context draft (Wang et al., 2026). GCM studies a jointly trained writer and reader; the reader supplies the detection signal after compression. Selective prediction supplies established tools for assessing abstention and risk-coverage tradeoffs (Geifman & El-Yaniv, 2017). Learn-then-Test (Angelopoulos et al., 2025) can calibrate a fixed policy family; our deployed thresholds use held-out empirical calibration, with the formal policy evaluated separately.

### K.2 Differences in memory construction and reading

**Adaptive retrieval and self-reflection.** Self-RAG (Asai et al., 2024) learns reflection tokens to control retrieval and assess retrieved evidence and generated answers. Its decision concerns retrieval and generation quality; ours concerns performance loss when raw context is replaced by compressed memory. GCM uses the reader's ordinary next-token distribution rather than separately supervised reflection tokens. TARG (Wang et al., 2026) also uses model uncertainty without a separate predictor, but estimates it from a short draft generated without retrieved context. GCM evaluates the compressed input after reader prefill, without generating a draft. These are differences in the decision target and signal acquisition, not a claim that all adaptive retrieval methods require a completed answer.

**Latent writers and readers.** LatentPress (Zhou & Sang, 2026) shares the motivation that memory need not be human-readable. Its small, reader-matched writer produces soft tokens for a frozen decoder, with reconstruction and distribution-distillation training. GCM also adapts the reader through LoRA and evaluates whether its confidence reveals compression failures. Thus direct latent reading and a simple writer are shared ingredients; our focus is their use in a detectable compression framework. LCLM (Li et al., 2026) develops an encoder-adapter-decoder compression system through architecture and training studies at scale. GCM instead reuses part of the LLM for prompt-conditioned writing and studies task performance together with failure detection.

**Corpus-specific optimization and factual reconstruction.** Cartridges (Eyuboglu et al., 2026) optimizes a compact KV cache offline for each corpus using self-study and context distillation, amortizing this cost over subsequent prompts. GCM trains a reusable writer, but normally rebuilds its prompt-conditioned memory for each prompt. Cartridges amortizes corpus-specific optimization, whereas GCM performs prompt-conditioned encoding at inference. Memory construction also differs in what the compressed states must support. NextMem (Zhang et al., 2026) uses an autoregressive autoencoder and progressive latent substitution to preserve and reconstruct factual memory. GCM uses reconstruction as an auxiliary objective; its primary outputs are task answers and a prompt-dependent failure-detection score, rather than a reconstructed source for subsequent use.

**Additional compression designs.** Semi-Dynamic Context Compression varies the pooling ratio across inputs (Yu et al., 2026b); Cramming 1568 optimizes a compact vector separately for each sample (Kuratov et al., 2025). These address budget allocation and representation capacity, respectively. They complement the question of whether a particular compressed memory remains usable. Across these families, training sets, compression budgets, reader training, and memory reuse differ; our settings specify these choices for each comparison.

## L Discussion of empirical results

**Accuracy-first gating and detection quality.** Our policy prioritizes answer quality: it retains compressed memory only when confidence exceeds a threshold selected to limit score decreases. A conservative threshold can therefore send non-degrading examples to raw context as well as actual failures. Fallback ratio measures this policy choice, whereas AUROC evaluates whether confidence distinguishes the two outcomes before a threshold is chosen. On Qwen3.5-9B BFCL, held-out gating combines AUROC $0.84 \pm 0.01$ with accuracy $85.8 \pm 0.3$, compared with $84.5$ for raw context, at $52.3\%$ fallback (Table 8). Across the five HotpotQA readers, AUROC $0.640$–$0.675$ accompanies gains of $0.95$–$6.16$ F1 points (Table 3). Thus, the evidence for detection comes from discrimination between paired outcomes, while gated accuracy assesses the use of that signal for correction.

Table 18: Existing paired outputs from Figure 4. Non-degrading prompts score at least as well as raw context. Minimum fallback is found by an empirical threshold sweep requiring gated mean score $\geq$ raw on these same outputs; it is separate from held-out calibration.

| reader | task | non-degrading prompts (%) | AUROC | minimum fallback (%) |
|---|---|---|---|---|
| Qwen3-8B | BFCL | 77.5 | 0.830 | 92.4 |
| Qwen3.5-9B | BFCL | 82.3 | 0.825 | 26.3 |
| Qwen3.5-9B | HotpotQA | 53.0 | 0.675 | 60.1 |

**Useful compression leaves room to reduce fallback.** The paired outputs underlying Figure 4 show that compression preserves or improves the raw-context score on $77.5\%$ and $82.3\%$ of BFCL prompts for Qwen3-8B and Qwen3.5-9B, respectively. These are relative-performance labels from Equation 1, including tied scores. Their AUROCs of $0.830$ and $0.825$ show that confidence provides information for selecting among these outcomes. Lowering the threshold admits more compressed answers and reduces fallback. In an empirical sweep of the existing Qwen3.5-9B BFCL outputs, $\tau \approx 0.4014$ retains the raw mean accuracy with $26.3\%$ fallback; the analogous HotpotQA sweep reaches $60.1\%$ fallback at $\tau \approx 0.3179$ (Table 18). These retrospective operating points characterize the available accuracy–fallback tradeoff, while the held-out results evaluate thresholds selected before scoring the evaluation fold. Together, they show why a high fallback ratio need not mean that most compressed answers are worse: it also reflects how conservatively the policy uses an informative signal.

**Joint adaptation and compression budgets.** The main comparison evaluates the complete writer-reader design. GCM trains its latent prompts, projection, and reader adapter; text and KV-cache methods retain their specified reader configurations (Appendix H). Compression-only scores measure the resulting system's ability to answer from a short representation, while the loss ablations isolate the contribution of supervision within GCM. LongBench tests reuse without target-task training: GCM retains $128S$ states for $S$ chunks, whereas the textual baselines retain 128 tokens in total (Table 2). The gains therefore describe these explicit operating budgets. The first-line answer protocol is shared by compressed and raw inputs, making the paired score decrease a consistent evaluation target; the associated results concern short-answer performance under that protocol.

**Evidence selection in long multi-document inputs.** LongBench's multi-document QA tasks require finding and combining evidence within long inputs (Bai et al., 2024). A short truncation can remove supporting passages while retaining distracting material; conversely, a larger raw window exposes more evidence but also more material unrelated to the prompt. In Table 2, increasing the raw window from 16K to 32K changes mean F1 from $19.9$ to $19.2$ on LB-HotpotQA and leaves LB-2WikiMQA at $16.0$. More available context alone therefore does not improve these readers under the evaluated answer protocol. Prompt-conditioned compression offers a different tradeoff: it can select information across chunks and concentrate it into a shorter reader input. The observed gains are consistent with reducing distraction while retaining answer-relevant evidence, particularly when useful passages are dispersed. This interpretation connects the results to long-input evidence selection; isolating truncation, distraction, and answer-format effects remains a direction for further controlled analysis.

**An early signal for relative performance loss.** First-token confidence estimates whether compressed memory lowers the reader's task score, rather than locating an omitted fact or certifying that an answer is correct. The comparison uses the same prompt and reference answer for compressed and raw inputs. Confidence leads the tested signals in 15 of 17 runs, including comparisons with feature-based learned predictors (Table 5); removing distillation or reconstruction reduces detection AUROC to $0.45$ or $0.48$ (Figure 7). These results support the usefulness of reader confidence and retention supervision for the defined detection target. We further hypothesize that initial confidence reflects the reader's familiarity with the compressed pattern: structures encountered during training may support a more concentrated first-token distribution, whereas unfamiliar combinations may leave it uncertain. Familiarity concerns learned structure, not necessarily a previously seen example; confidence is not a test of training-set membership. The retention losses aim to preserve this structure rather than turn unfamiliar inputs into a familiar textual account. The ablation and transfer results

motivate this interpretation, while direct tests of pattern exposure and output-format effects are left to future work.

**What transfers across tasks, readers, and writers.** With the HotpotQA-trained writer and reader fixed, Qwen3-8B retains AUROC $0.674$ on LB-HotpotQA and $0.626$ on LB-2WikiMQA (Appendix G.1). This directly tests cross-task reuse. The five-reader results instead test the same design trained separately for each LLM. Replacing the writer with Cartridges while keeping the GCM-trained reader and confidence signal fixed tests a third form of reuse (Figure 5). These complementary observations motivate shared, detectable compression-failure patterns as a hypothesis: task transfer shows reuse of trained components, reader comparisons show applicability across LLMs, and writer replacement shows compatibility beyond the original compressor. We hypothesize that different LLMs learn partly shared semantic structure, allowing related failure patterns to remain detectable across models, consistent with the representational convergence proposed by Huh et al. (2024). The observed cross-reader consistency supports this possibility, rather than guaranteeing transfer of a fixed writer or detector between LLMs. Testing such direct cross-model transfer and identifying which patterns persist are natural next steps.