When Is More Context Worth It? Risk-Controlled Expansion for Compressed-Context Inference

Anonymous authors
Paper under double-blind review

Abstract

Context compression reduces the cost of long-context inference but can remove information needed for a particular query. Selective expansion can recover such information, yet uncertainty or context-sufficiency estimates only indicate whether the current answer should be trusted; they do not determine which part of the context would improve it or whether the expected improvement justifies its cost. We propose Risk-Controlled Progressive Expansion (RCPE), a cost-sensitive sequential framework for answering, restoring compressed context, or abstaining. RCPE separately estimates current-answer reliability and the value of candidate restoration actions, learning the latter from task-score changes observed after controlled context additions. After each restoration, it regenerates the answer and reassesses the resulting state. RCPE calibrates complete policies on independent data using simultaneous confidence bounds, controlling absolute task loss, compression-induced loss, and minimum coverage under i.i.d. sampling. Across LongBench and ZeroSCROLLS, RCPE recovers quality lost under fixed compression and improves the quality–cost trade-off over relevance-only restoration, while processing fewer context tokens and achieving lower end-to-end latency than full-context inference. We will release our code and checkpoints on GitHub.

1 Introduction

Context compression reduces the cost of processing long inputs by replacing context with latent memories, selected tokens, or reduced key–value (KV) caches (Chevalier et al., 2023; Ge et al., 2023; Jiang et al., 2024; Li et al., 2024). Its failures, however, are query dependent: a compressed experimental report may preserve the main finding while omitting the learning rate, and thus support one question about the report but not another. Restoring the original hyperparameter table could repair the latter answer without processing the full report. Yet each restoration introduces additional access to and processing of the context, as well as answer generation, so repeated expansion can erase the computational savings of compression. The central question is therefore not simply whether more context is needed, but *when additional context is worth its computational cost*.

Answer confidence, semantic uncertainty, and context-sufficiency estimates provide natural signals for selective answering or additional retrieval (Kadavath et al., 2022; Farquhar et al., 2024; Asai et al., 2024; Jeong et al., 2024; Joren et al., 2024; Xie et al., 2025; Moskvoretskii et al., 2025); latent-context models additionally provide a mechanism for restoring compressed context segments on demand (Li et al., 2026). These signals can indicate that the current answer should not yet be trusted, but they do not determine whether restoration can repair the answer, which candidate context chunk will help, or whether its expected gain justifies its cost. Relevance alone is insufficient: evidence may already be represented in the compressed state, may not be used effectively after restoration (Liu et al., 2024), or may provide less benefit than another context chunk with a similar retrieval score. Figure 1 illustrates this mismatch and shows that ordering restorations by benefit per token can improve answer quality more rapidly than ordering them by relevance. We therefore formulate restoration as a cost-sensitive sequential decision and propose *Risk-Controlled Progressive Expansion* (RCPE), which learns the expected change in task score caused by adding each candidate context chunk in the current inference state.

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.

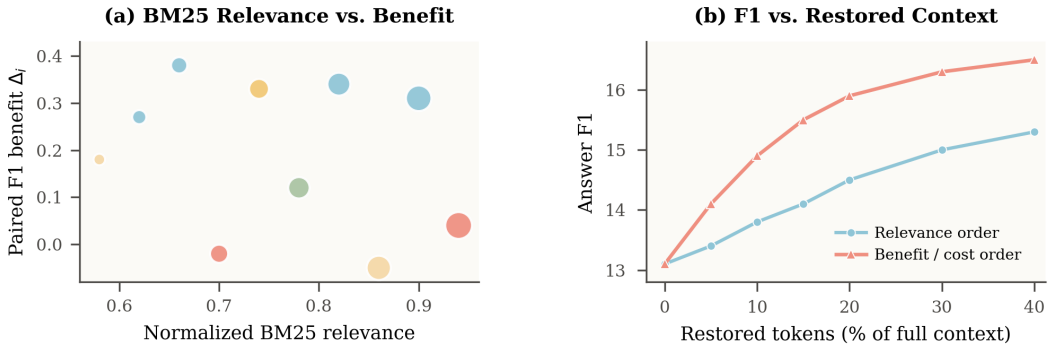

Figure 1: Relevance is not restoration value. On the LongBench single-document QA tasks (Bai et al., 2023), we independently compress non-overlapping 512-word chunks with `AutoCompressor-Llama-2-7b-6k` (Chevalier et al., 2023) and restore original chunks before regenerating the answer. **(a)** Across the ten highest-ranked BM25 chunks, mean normalized relevance is not consistently aligned with the paired change in token-level answer F1, Δ_i; bubble area also encodes relevance. **(b)** Ordering chunks by estimated benefit per token yields higher answer F1 at the same restored-token budget than ordering them by BM25 relevance.

Starting from a compressed state and its candidate answer, RCPE separates two decisions that uncertainty alone conflates. Reliability estimators assess whether the current answer can be returned, including its absolute quality and the performance retained relative to full-context inference. A restoration action-value model instead predicts the task-score change produced by adding a candidate context chunk, using beneficial, neutral, and harmful restoration outcomes as supervision. A cost-aware controller combines these estimates with incremental processing cost to answer, restore a context chunk or a small group of chunks, or abstain; after restoration it regenerates the answer and reassesses the new state. Because neither reliability nor a predicted action value is itself a guarantee, RCPE calibrates complete policies from a fixed candidate family on independent data using simultaneous statistical bounds (Angelopoulos et al., 2021). The constraints limit absolute task loss, enforce minimum coverage, and limit normalized compression risk—the score lost relative to full context on returned examples. For binary scores, the latter reduces to error on examples that full-context inference answers correctly. Among certified policies, RCPE selects the lowest-cost one; if none is certified, the requested operating point remains uncertified. Reference answers and full-context scores are used only for training and calibration, never as inference-time inputs, and evaluation charges compression, estimation, context access, and every generation round to the complete trajectory.

Across LongBench and ZeroSCROLLS, and across latent and textual compressors, RCPE recovers quality lost under fixed compression while reading fewer context tokens and achieving lower end-to-end latency than full-context inference with Qwen3-4B-Instruct-2507. At matched risk and answer coverage, restoration-outcome supervision provides a better quality–cost trade-off than relevance-only restoration, with the estimator and unsuccessful-restoration overheads included. Our contributions are threefold: **(1) A cost-sensitive restoration formulation:** We formulate compressed-context expansion as a sequential decision over answering, restoring context chunks, and abstaining, where each restoration is valued by its expected task-score gain relative to incremental cost; **(2) Restoration-outcome learning:** RCPE separates answer reliability from recoverability and learns candidate-specific action values from measured changes after controlled context restoration; and **(3) Policy-level risk control and evaluation:** We calibrate the complete policy for risk and coverage and evaluate it against full-context, fixed-compression, and relevance-only baselines using matched constraints and full trajectory cost accounting.

2 Related Work

Context Compression and Selective Restoration. Long-context methods reuse or compress hidden states (Dai et al., 2019; Rae et al., 2020; Wu et al., 2022; Bulatov et al., 2022), learn continuous

summaries (Chevalier et al., 2023; Ge et al., 2023; Mu et al., 2023), or organize memory hierarchically (Munkhdalai et al., 2024; He et al., 2025; Chen et al., 2025). Textual compressors select tokens or spans (Li et al., 2023; Jiang et al., 2023; 2024), while KV-cache methods retain selected internal states (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024). Query-dependent latent models further support selective access to or restoration of context segments (An et al., 2025; Li et al., 2026). These methods determine how information is compressed and made available. RCPE instead learns, for a particular inference state, which available restoration is expected to improve the answer enough to justify its cost.

Adaptive Retrieval and Cost-Aware Computation. Self-RAG and Adaptive-RAG decide whether and how much external information to retrieve (Asai et al., 2024; Jeong et al., 2024), while uncertainty-based retrieval allocates additional evidence to difficult queries (Moskvoretskii et al., 2025). Model cascades similarly route examples to more expensive computation (Chen et al., 2023; Narasimhan et al., 2025). RCPE shares this adaptive-computation perspective, but its actions restore chunks from the original compressed context. Rather than using relevance or difficulty alone, it supervises each action with the task-score change observed after restoration and compares the predicted gain with incremental cost.

Selective Prediction and Risk Control. Selective prediction formalizes the trade-off between answer risk and coverage (Geifman & El-Yaniv, 2017). Language models provide useful but imperfect signals of correctness or semantic uncertainty (Kadavath et al., 2022; Farquhar et al., 2024), and context-sufficiency estimators use related signals to answer selectively or stop reading additional context (Joren et al., 2024; Xie et al., 2025). Conformal risk control and learn-then-test methods provide finite-sample tools for selecting operating rules (Angelopoulos et al., 2021; 2024). RCPE applies these tools to the complete multi-step policy, jointly controlling absolute risk, compression-relative risk, and minimum coverage rather than restoration actions.

3 Problem Formulation

Let $\mathbf{X}$ denote a textual context, $\mathbf{q}$ a query, and $\mathbf{y}^*$ a reference answer. We partition $\mathbf{X}$ into an ordered sequence of k non-overlapping chunks $\{\mathbf{x}_i\}_{i=1}^{k}$, indexed by $[k] = \{1, \dots, k\}$. A fixed compressor maps $\mathbf{X}$ to an initial compressed memory $\mathbf{M}_0$.

At restoration step t, let $\mathcal{S}_t \subseteq [k]$ be the indices of chunks already restored, and let $\mathbf{R}_t = \mathrm{Concat}^{\uparrow}_{i \in \mathcal{S}_t} \mathbf{x}_i$ denote their original contents concatenated in increasing index (i.e., context) order. Let $\mathbf{M}_t$ be the current compressed memory, obtained from $\mathbf{M}_0$ by removing the blocks corresponding to $\mathcal{S}_t$. The frozen backbone produces the current answer and its full-context counterpart as

$$\mathbf{y}_t = F(\mathbf{q}, \mathbf{M}_t, \mathbf{R}_t), \qquad \mathbf{y}_{\mathrm{full}} = F(\mathbf{q}, \mathbf{X}). \tag{1}$$

Let $a_t \in [0, 1]$ be the normalized task score of $\mathbf{y}_t$ against $\mathbf{y}^*$, and define a_{full} analogously for $\mathbf{y}_{\mathrm{full}}$. Depending on the task, this score may be accuracy, F1, BLEU, ROUGE, or another official metric; binary correctness is the special case $a_t \in \{0, 1\}$.

A policy π repeatedly chooses among answering, restoring more chunks, and abstaining. Let T_π be its stopping step and $D_\pi \in \{0, 1\}$ indicate whether it returns an answer. We define normalized compression risk as the expected score lost relative to full context, divided by the full-context score available on answered examples:

$$R_{\mathrm{comp}}(\pi) = \frac{\mathbb{E}[D_\pi [a_{\mathrm{full}} - a_{T_\pi}]_+]}{\mathbb{E}[D_\pi a_{\mathrm{full}}]}. \tag{2}$$

The denominator is required to be positive. We also measure absolute task loss over returned answers and the probability of answering:

$$R_{\mathrm{abs}}(\pi) = \frac{\mathbb{E}[D_\pi (1 - a_{T_\pi})]}{\mathbb{E}[D_\pi]}, \qquad \mathrm{Cov}(\pi) = \mathbb{E}[D_\pi]. \tag{3}$$

Given target risks ϵ and δ and minimum coverage κ, we seek the least-cost policy satisfying

$$\min_{\pi} \ \mathbb{E}[\mathrm{Cost}(\pi)] \quad \text{s.t.} \quad R_{\mathrm{comp}}(\pi) \le \epsilon, \ R_{\mathrm{abs}}(\pi) \le \delta, \ \mathrm{Cov}(\pi) \ge \kappa. \tag{4}$$

Here $\mathrm{Cost}(\pi)$ includes context processing, estimator overhead, and all generation rounds along the policy trajectory. Solving this problem requires estimating both whether the current answer is reliable and which additional chunks are worth restoring.

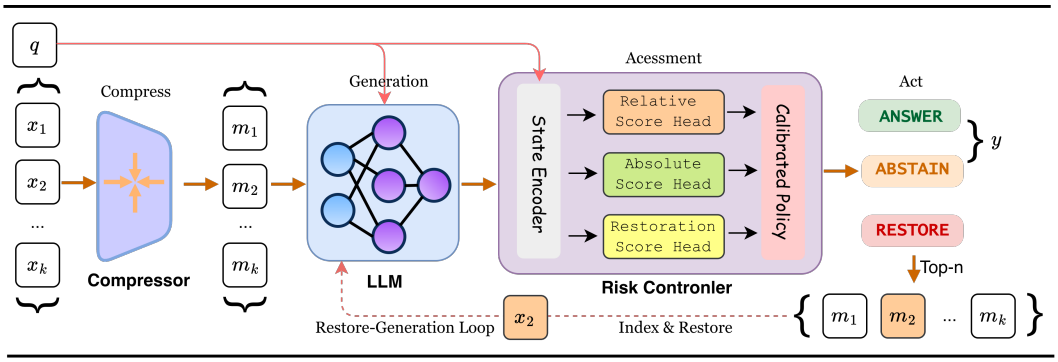

Figure 2: Overview of RCPE online inference. The query and compressed chunk memories produce a candidate answer and state representation. Relative-, absolute-, and restoration-score heads assess the current answer and the value of expanding individual chunks. The calibrated policy then answers, abstains, or restores a selected context chunk and repeats generation and assessment.

4 Risk-Controlled Progressive Expansion

RCPE solves Equation 4 with a progressive policy learned from paired restoration trajectories. Offline, it constructs task-score and restoration-gain labels, trains reliability and utility estimators, and selects a risk-controlled policy on an independent calibration split. Online, the selected policy alternates between assessment, restoration, and regeneration. Figure 2 summarizes this online loop.

4.1 Training Data Construction

For each labeled example $(\mathbf{X}, \mathbf{q}, \mathbf{y}^*)$, we run the frozen backbone with full context and compressed memory to obtain $(\mathbf{y}_{\text{full}}, a_{\text{full}})$ and $(\mathbf{y}_0, a_0)$. We then construct restoration trajectories at several compression ratios. To cover both effective and ineffective expansions, chunks are restored in annotated-evidence order when available, retriever order, current utility-model order, and random order. At every trajectory step t, we record

$$(\mathbf{q}, \mathbf{M}_t, \mathcal{S}_t, \mathbf{y}_t, a_t, a_{\text{full}}).$$

Thus, the training data contain states in which restoration improves, preserves, or reduces the task score, as well as states for which full context itself performs poorly.

We obtain direct supervision for each available restoration action. For an unrestored chunk $i \notin \mathcal{S}_t$, let $a^+_{t,i}$ be the task score after adding $\mathbf{x}_i$ and regenerating the answer. Its observed utility is

$$\Delta_{t,i} = a^+_{t,i} - a_t. \tag{5}$$

Positive, zero, and negative values respectively identify beneficial, neutral, and harmful restorations. When computation permits, we also delete chunks from high-scoring states and measure the resulting score change. Hard cases are created by removing required evidence, replacing decisive entities or numbers, inserting plausible contradictions, and preserving query words while breaking the required relation. Contexts are split before trajectory construction so that related states cannot cross the training, validation, and calibration splits. Appendix D gives the complete counterfactual trajectory-construction and hard-negative procedure.

4.2 RCPE Architecture

Context Correspondence. Write the initial compressed memory as

$$\mathbf{M}_0 = (\mathbf{m}_1, \ldots, \mathbf{m}_k), \tag{6}$$

where $\mathbf{m}_i$ is the compressed representation associated with chunk $\mathbf{x}_i$. For textual compression, $\mathbf{m}_i$ contains the retained tokens from that chunk; for latent compression, it contains the corresponding memory vectors. This shared index provides a direct correspondence between each restoration candidate and the compressed representation it replaces.

State and Reliability. A shared encoder maps the current inference state to

$$\mathbf{r}_t = E(\mathbf{q}, \mathbf{M}_t, \mathbf{R}_t, \mathbf{y}_t). \tag{7}$$

The state representation may include answer uncertainty, semantic agreement across sampled answers, retrieval-score statistics, and answer–evidence compatibility. RCPE predicts two complementary reliability scores in $[0, 1]$:

$$s_t^{\mathrm{rel}} = g_\psi^{\mathrm{rel}}(\mathbf{r}_t) \approx \mathbb{E}[1 - [a_{\mathrm{full}} - a_t]_+ \mid \mathbf{r}_t], \tag{8}$$

$$s_t^{\mathrm{abs}} = g_\eta^{\mathrm{abs}}(\mathbf{r}_t) \approx \mathbb{E}[a_t \mid \mathbf{r}_t]. \tag{9}$$

The relative score is high when the current state preserves full-context performance; the absolute score estimates current answer quality. Keeping the two predictions separate prevents a low-quality full-context answer from being mistaken for a compression failure. Here ψ and η denote the learned parameters of the two heads.

Restoration Utility. For each unrestored chunk $\mathbf{x}_i$, let $\boldsymbol{\rho}_i$ collect lightweight metadata such as context position and retrieval similarity. The utility head predicts

$$u_{t,i} = h_\omega(\mathbf{r}_t, \mathbf{m}_i, \boldsymbol{\rho}_i) \approx \mathbb{E}[\Delta_{t,i} \mid \mathbf{r}_t, \mathbf{m}_i, \boldsymbol{\rho}_i], \qquad i \notin \mathcal{S}_t. \tag{10}$$

Here ω denotes the learned utility-head parameters. We constrain $u_{t,i} \in [-1, 1]$ to match the range of $\Delta_{t,i}$. Let $c_{t,i} > 0$ denote the estimated incremental cost of restoring $\mathbf{x}_i$, measured in the same units as the policy budget. The controller ranks actions by predicted gain per unit cost,

$$v_{t,i} = \frac{u_{t,i}}{c_{t,i}}. \tag{11}$$

4.3 Training and policy calibration

The reliability heads and utility head are trained against the continuous labels defined above:

$$\mathcal{L}_{\mathrm{rel}} = \sum_t \left(s_t^{\mathrm{rel}} - (1 - [a_{\mathrm{full}} - a_t]_+)\right)^2, \tag{12}$$

$$\mathcal{L}_{\mathrm{abs}} = \sum_t (s_t^{\mathrm{abs}} - a_t)^2, \tag{13}$$

$$\mathcal{L}_{\mathrm{util}} = \sum_t \sum_{i \notin \mathcal{S}_t} (u_{t,i} - \Delta_{t,i})^2. \tag{14}$$

The complete objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{rel}} + \beta \mathcal{L}_{\mathrm{abs}} + \gamma \mathcal{L}_{\mathrm{util}}, \tag{15}$$

where $\beta, \gamma \geq 0$ balance the three terms. We periodically regenerate trajectories with the current utility model to reduce the mismatch between training states and states visited by the learned policy.

Risk control is applied to the complete adaptive policy, not to an individual prediction. Let Θ be a finite candidate family whose members specify the two reliability thresholds, a utility threshold, restoration batch size, and maximum restoration budget. After fixing the learned components and Θ, we execute every $\theta \in \Theta$ on an independent calibration split. Let $U_{\mathrm{comp}}(\theta)$ and $U_{\mathrm{abs}}(\theta)$ be simultaneous upper confidence bounds on the two risks, and let $L_{\mathrm{cov}}(\theta)$ be a simultaneous lower confidence bound on coverage. Appendix F.1 reports the finite candidate family used in our experiments. We select

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \widehat{\mathrm{Cost}}(\theta) \tag{16}$$

subject to

$$U_{\mathrm{comp}}(\theta) \leq \epsilon, \qquad U_{\mathrm{abs}}(\theta) \leq \delta, \qquad L_{\mathrm{cov}}(\theta) \geq \kappa. \tag{17}$$

If no candidate satisfies all constraints, the requested operating point is reported as uncertified.

Proposition 1 (Finite-family risk and coverage control)**.** *Assume the calibration examples are i.i.d. from the target test distribution, every policy in Θ is fixed independently of the calibration sample, and simultaneous one-sided confidence bounds are constructed at total failure probability α. Then, with probability at least $1 - \alpha$, every policy declared feasible—including the selected policy—satisfies the population constraints in Equation 4.*

The result follows from simultaneous validity and a union bound over Θ; Appendix G gives a proof sketch. For R_{comp}, the lower confidence bound on $\mathbb{E}[D_\pi a_{\mathrm{full}}]$ must be positive so that Equation 2 is well defined.

4.4 INFERENCE

RCPE begins with compressed memory and no restored chunks, $\mathcal{S}_0 = \varnothing$. At step t, it generates the current answer, encodes $\mathbf{r}_t$, predicts the two reliability scores, and scores every available restoration. Let B_t be the cumulative restoration cost through step t. The calibrated policy acts as

$$\pi_{\hat{\theta}}(\mathbf{r}_t) = \begin{cases} \textsc{Answer}, & s_t^{\mathrm{rel}} \geq \tau_{\mathrm{rel}} \wedge s_t^{\mathrm{abs}} \geq \tau_{\mathrm{abs}}, \\ \textsc{Expand}, & s_t^{\mathrm{rel}} < \tau_{\mathrm{rel}} \wedge \exists i \notin \mathcal{S}_t : v_{t,i} \geq \tau_u,\ B_t + c_{t,i} \leq B_{\max}, \\ \textsc{Abstain}, & \text{otherwise.} \end{cases} \tag{18}$$

Here τ_{rel}, τ_{abs}, and τ_u are the thresholds specified by $\hat{\theta}$, while $B_{\max}$ is its restoration budget. For EXPAND, let b be the calibrated batch size. The controller greedily restores up to b highest-scoring available chunks whose cumulative incremental cost fits the remaining budget. It then updates $(\mathbf{M}_{t+1}, \mathbf{R}_{t+1})$, regenerates the answer, and repeats the assessment. The trajectory terminates when the answer thresholds are met, no feasible action exceeds the utility threshold, or the budget is exhausted. Full-context outputs, reference answers, and task-score labels are used only offline; they are never inference-time inputs. Appendix F.2 provides the complete test-time execution algorithm.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets and Evaluation. We evaluate RCPE on LongBench (Bai et al., 2023), covering its six standard task families: single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion. Following the LongLLMLingua protocol (Jiang et al., 2024), we use the official metric for each dataset, aggregate results by task family, and report the macro-average across the six families. We additionally evaluate on ZeroSCROLLS (ZeroS) (Shaham et al., 2023), a zero-shot benchmark spanning long-document question answering, summarization, information aggregation, and classification, and report its aggregate score using the official task metrics. Further details on evaluation, matched comparisons, and trajectory-level cost accounting are provided in Appendix H.1.

Model and Training. We apply RCPE to two primary backbone–compressor combinations: LCLM (Li et al., 2026) with Qwen3-4B-Instruct-2507, and AutoCompressor (Chevalier et al., 2023) with Llama-2-7B. To evaluate RCPE across a shared textual-compression interface, we additionally pair LLMLingua-2 (Pan et al., 2024) with each backbone. Every backbone–compressor pair is frozen before trajectory construction. After filtering overlaps with the evaluation documents, we repartition the source training pool at the example level into three disjoint subsets: *controller training*, used to optimize the prediction heads and input projections; *controller validation*, used for early stopping, checkpoint selection, and construction of the fixed policy family; and an independent *policy-validation* split, used only to replay complete policies, compute simultaneous calibration bounds, and select the lowest-cost certified policy. The controller uses a frozen shared embedding encoder, two MLP reliability heads, and a candidate-conditioned cross-attention utility head. Detailed data provenance, architectures, objectives, checkpoints, and optimization settings are provided in Appendices C and E. Compression ratios, context chunking, and restoration procedures are detailed in Appendix A; robustness results at $6\times$ and $8\times$ compression are reported in Appendix B.

Baselines. We compare against the methods reported in Table 1: full-context inference, BM25 retrieval (Robertson & Zaragoza, 2009), the textual compression methods LLMLingua (Jiang et al., 2023), LongLLMLingua (Jiang et al., 2024), and LLMLingua-2 (Pan et al., 2024), and the latent compression methods LCLM (Li et al., 2026) and AutoCompressor (Chevalier et al., 2023). LCLM is evaluated with Qwen3-4B-Instruct-2507 and AutoCompressor with Llama-2-7B, following their compatible backbones. All methods use the same task-specific prompts and decoding parameters; the exact prompts and decoding specification are provided in Appendix I. Methods that restore context chunks use the same chunking and restoration budget as RCPE.

5.2 MAIN RESULTS

Table 1 supports our central claim that selective context restoration can recover quality lost under fixed context compression. At the $4\times$ initial compression ratio, the base compressors trail their full-

Table 1: Performance on LongBench and ZeroS at matched $(\epsilon, \delta, \kappa)$. Compressed methods use $4\times$ initial compression. LongBench columns are task families, ZeroS is its official-metric aggregate, and AVG averages the six LongBench families and ZeroS.

Methods	SingleDoc	MultiDoc	Summ.	FewShot	Synth.	Code	ZeroS	AVG
Qwen3-4B-Instruct-2507 backbone								
Full context	40.8	39.5	27.1	68.4	39.6	55.8	33.1	43.5
BM25	34.2	35.6	25.8	58.7	44.8	48.5	20.4	38.3
LLMLingua	33.5	37.1	26.3	66.5	15.2	53.4	28.2	37.2
LongLLMLingua	40.1	43.8	27.5	68.9	50.7	54.9	33.0	45.6
LLMLingua-2	37.2	39.0	26.8	68.1	27.6	57.3	33.7	41.4
LCLM	38.6	37.9	26.4	65.7	36.8	52.6	30.8	41.3
RCPE (LCLM)	**41.3**	**40.7**	**27.3**	**68.2**	**42.1**	**55.6**	**34.2**	**44.2**
RCPE (LLMLingua-2)	**40.9**	**40.3**	**27.2**	**68.7**	**41.5**	**56.9**	**34.5**	**44.3**
Llama-2-7B backbone								
Full context	31.8	32.4	25.7	61.5	35.2	50.6	28.4	37.9
BM25	28.9	30.8	24.5	52.7	39.6	44.1	17.6	34.0
LLMLingua	27.6	31.2	25.0	59.8	13.5	48.7	23.9	32.8
LongLLMLingua	32.5	35.7	26.1	62.3	45.8	50.4	28.1	40.1
LLMLingua-2	30.1	32.6	25.4	61.7	24.9	52.2	28.7	36.5
AutoCompressor	28.7	29.6	24.8	57.4	31.9	46.8	24.6	34.8
RCPE (AutoCompressor)	**31.1**	**31.7**	**25.5**	**59.8**	**35.1**	**49.0**	**27.8**	**37.1**
RCPE (LLMLingua-2)	**32.0**	**33.8**	**26.0**	**62.2**	**35.8**	**52.5**	**29.5**	**38.8**

context references by 1.4–3.1 AVG points. RCPE improves all four backbone–compressor pairs, adding 2.9 points with Qwen3-4B-Instruct-2507 and 2.3 points with Llama-2-7B. Three of the four RCPE configurations exceed their corresponding full-context AVG, while RCPE (AutoCompressor) reduces the remaining gap to 0.8 points. The consistent gains across Qwen3-4B-Instruct-2507 and Llama-2-7B, latent compression (LCLM and AutoCompressor), and textual compression (LLMLingua-2) strengthen the evidence that RCPE is not tied to a particular backbone or compressor.

An interesting trend is that the largest improvements appear on synthetic tasks, where RCPE gains 3.2–13.9 points over the underlying compressors. The gains also transfer to ZeroS, ranging from 0.8 to 3.4 points, and three of the four configurations improve every reported task group; the remaining configuration improves six of seven. These results indicate that the improvement is broadly distributed rather than driven by a single benchmark. LongLLMLingua nevertheless retains the highest overall AVG, reaching 45.6 with Qwen3-4B-Instruct-2507 and 40.1 with Llama-2-7B. Thus, the results validate RCPE primarily as a robust mechanism for repairing existing compressed-context systems, rather than as a method that universally dominates every query-aware compressor.

5.3 Risk-Control Analysis

We next test whether calibration controls the complete RCPE policy as specified in Equation 4. We use the $4\times$ LCLM–Qwen3-4B-Instruct-2507 configuration on LongBench as a representative audit. For fixed absolute-risk and minimum-coverage targets (δ, κ), we vary the requested compression-risk level ϵ. For each target, the policy is selected from the frozen candidate family in Appendix F.1 using only the calibration split. Table 2 reports all three calibration bounds, the corresponding held-out test quantities, and the resulting trajectory cost.

A policy is certified only when $U_{\mathrm{comp}} \leq \epsilon$, $U_{\mathrm{abs}} \leq \delta$, and $L_{\mathrm{cov}} \geq \kappa$. Successful risk control therefore requires more than a low test error: the policy must satisfy both risk constraints without reducing answer coverage below the requested level. As ϵ becomes more stringent, the selected policy incurs higher trajectory cost and returns answers on slightly fewer examples, while still maintaining the fixed coverage requirement. The $\epsilon = 0.10$ and 0.15 operating points are certified. At $\epsilon = 0.05$, the held-out compression risk remains below the target, but $U_{\mathrm{comp}} = 0.0601$ exceeds it, so the operating point is not certified. Among the certified targets, relaxing ϵ from 0.10 to 0.15

Table 2: Risk-control audit for $4\times$ LCLM–Qwen3-4B-Instruct-2507 on LongBench at $\delta = 0.20$ and $\kappa = 0.80$. U_{comp} and U_{abs} are simultaneous upper bounds; L_{cov} is a simultaneous lower bound. Full-trajectory cost is normalized to full-context inference (1.00).

Target ϵ	U_{comp}	U_{abs}	L_{cov}	**Test** R_{comp}	**Test** R_{abs}	**Test coverage**	**Norm. cost**	**Certified**
0.05	0.0601	0.176	0.805	0.041	0.154	0.821	0.62	No
0.10	0.093	0.184	0.824	0.081	0.165	0.839	0.47	Yes
0.15	0.142	0.196	0.843	0.126	0.179	0.857	0.36	Yes

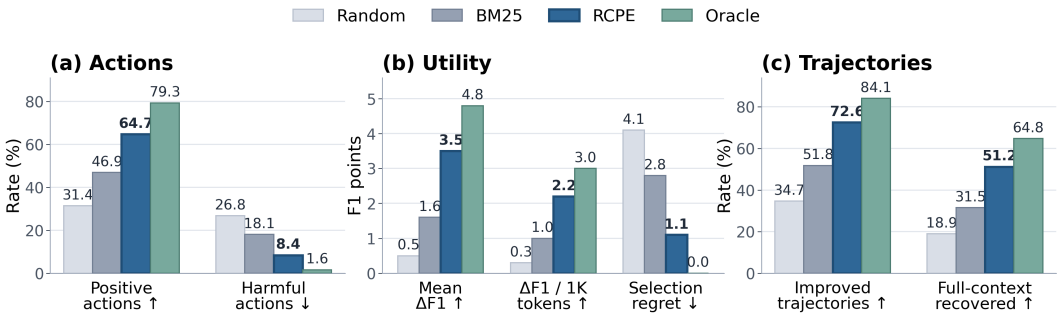

Figure 3: Restoration effectiveness at matched token budgets. **(a)** Positive and harmful action rates. **(b)** F1 gain, gain per 1K restored tokens, and selection regret. **(c)** Fractions of degraded trajectories that improve or recover the full-context score. Oracle selects the best counterfactual action.

reduces normalized cost from 0.47 to 0.36 and increases test coverage from 0.839 to 0.857, at the expected cost of higher compression risk. Appendix G.1 further audits this behavior over 100 alternative calibration samples, reporting certification frequency and simultaneous constraint-violation rates with confidence intervals.

5.4 Restoration Effectiveness

On held-out LongBench QA examples, for each executed restoration i_t, we measure the realized gain $\Delta_t^{\text{sel}} = a_{t+1} - a_t$, where a_t and a_{t+1} are the scores before and after restoration. We report positive and harmful action rates, mean gain, and gain per restored token. Counterfactual evaluation of all candidates yields the oracle gain $\Delta_t^* = \max_{i \notin \mathcal{S}_t} \Delta_{t,i}$ and selection regret $\Delta_t^* - \Delta_t^{\text{sel}}$. We compare random, BM25, RCPE, and oracle selection at matched token budgets and measure whether degraded answers improve or recover their full-context score.

Figure 3 shows that RCPE outperforms BM25 at both action and trajectory levels: it raises the positive-action rate from 46.9% to 64.7%, reduces harmful actions from 18.1% to 8.4%, and increases mean gain from 1.6 to 3.5 F1 points while lowering regret from 2.8 to 1.1. RCPE improves 72.6% of degraded trajectories and recovers the full-context score on 51.2%, versus 51.8% and 31.5% for BM25. The oracle gap leaves room for better candidate ranking and evidence use.

5.5 Efficiency and Ablation Study

Because repeated decoding can offset token savings, Figure 4 compares task performance with end-to-end latency and normalized trajectory cost. The latency panel uses the Qwen3-4B-Instruct-2507 methods in Table 1; ablations use RCPE (LCLM). Each method is independently calibrated at $(\epsilon, \delta, \kappa) = (0.10, 0.20, 0.80)$.

RCPE (LCLM) and RCPE (LLMLingua-2) have mean end-to-end latencies of 1.97 and 1.86 seconds, respectively, about 20% and 24% less than full-context inference. Single-pass retrieval and fixed-compression baselines remain faster, but both RCPE variants improve their underlying compressors while staying faster than full-context inference. Removing counterfactual utility supervision causes the largest drop, 2.5 AVG points, while removing the relative or absolute head costs 0.8

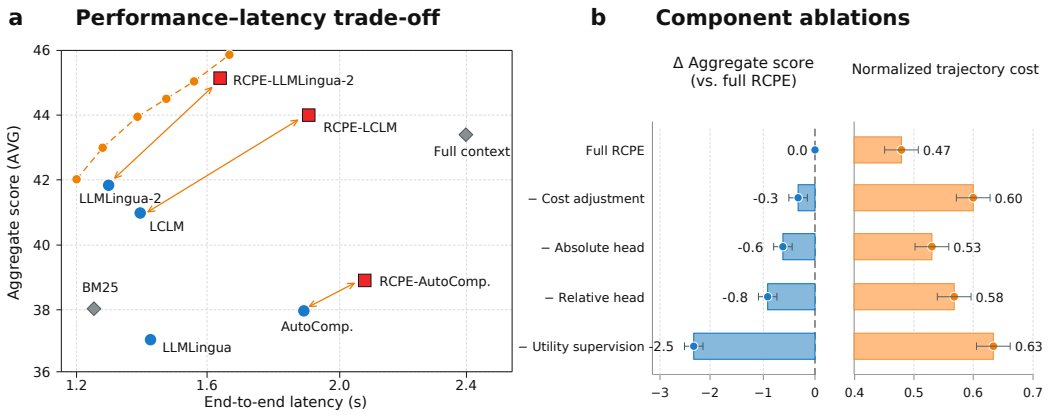

Figure 4: Performance–efficiency and ablation results. **(a)** Aggregate score versus end-to-end latency; the dashed curve is the empirical Pareto frontier, and arrows connect compressors to RCPE. **(b)** Change in aggregate score relative to full RCPE and normalized trajectory cost for matched component ablations.

and 0.6 points. Removing cost adjustment changes quality little but substantially increases trajectory cost. Appendix H.2 provides the detailed latency and memory measurement protocol and the component-level breakdown.

5.6 Error analysis

Controlled reruns assign each unsuccessful trajectory to its earliest failure stage. A *trigger* failure retains a degraded compressed answer without restoration; a *selection* failure occurs when a more beneficial chunk was available; and a *stopping* failure terminates despite a remaining helpful action. We assign *evidence-utilization* failure when the required evidence has been restored but regeneration does not use it, and *backbone* failure when even full-context inference fails. Figure 5 shows that evidence utilization (39%) and backbone failure (27%) dominate, so many errors persist after relevant evidence is available rather than arising only from expansion decisions. Controller-side failures comprise selection (17%), stopping (9%), and triggering (8%), identifying candidate ranking and effective evidence use as the main improvement opportunities.

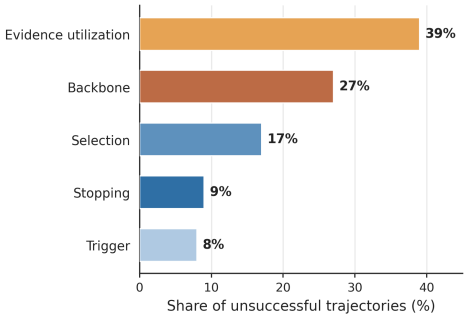

Figure 5: Failure-stage composition among unsuccessful RCPE trajectories.

6 Conclusion

We introduced RCPE, a sequential framework that decides whether to answer, restore compressed context, or abstain. RCPE separates current-answer reliability from restoration value: reliability heads assess the present answer, while an action-value model learns chunk-specific score changes from counterfactual restorations. Regeneration and reassessment after every action allow the controller to adapt its decisions to the resulting inference state. Calibration is applied to the complete policy, jointly controlling absolute task loss, compression-induced loss, and minimum answer coverage rather than certifying isolated predictions or restoration actions. Across LongBench and ZeroSCROLLS, RCPE consistently repairs quality lost by latent and textual compression and outperforms relevance-only restoration at matched targets. It remains effective under stronger compression while processing less context than full-context inference, supporting calibrated restoration as a practical quality–cost mechanism.

AI USE STATEMENT

Generative AI tools were used to assist with code implementation and manuscript polishing. The authors independently verified all code, technical claims, derivations, citations, and empirical results and take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The final release will include model and compressor versions, prompts, context chunking, paired trajectory construction, calibration splits, candidate policy families, confidence-bound implementation, random seeds, hardware, and complete risk–coverage–cost measurements. Code will reproduce every operating point in the main figures and tables, subject to model and dataset licenses.

REFERENCES

Sumin An, Junyoung Sung, Wonpyo Park, Chanjun Park, and Paul Hongsuck Seo. Lcirc: A recurrent compression approach for efficient long-form context and query dependent modeling in llms. *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International Conference on Learning Representations*, 2024.

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 et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. *arXiv preprint arXiv:2207.06881*, 2022.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Yinpeng Chen, DeLesley Hutchins, Aren Jansen, Andrey Zhmoginov, David Racz, and Jesper Andersen. Melodi: Exploring memory compression for long contexts. *International Conference on Learning Representations*, 2025.

Alexis Chevalier et al. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*, 2023.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 2024.

Tao Ge et al. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

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

Zifan He, Yingqi Cao, Zongyue Qin, Neha Prakriya, Yizhou Sun, and Jason Cong. Hmt: Hierarchical memory transformer for long context language processing. *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

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 *Annual Meeting of the Association for Computational Linguistics*, 2024.

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. *arXiv preprint arXiv:2411.06037*, 2024.

Saurav Kadavath et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

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, 2018.

Ang Li, Sean McLeish, Haozhe Chen, Nimit Kalra, Zaiqian Chen, et al. End-to-end context compression at scale. *arXiv preprint arXiv:2606.09659*, 2026.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. *Conference on Empirical Methods in Natural Language Processing*, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 2024.

Viktor Moskvoretskii et al. Adaptive retrieval without self-knowledge? bringing uncertainty back home. *arXiv preprint arXiv:2501.12835*, 2025.

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

Tsendsuren Munkhdalai et al. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *International Conference on Learning Representations*, 2025.

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 *Annual Meeting of the Association for Computational Linguistics*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

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

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022.

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.

Roy Xie, Junlin Wang, Paul Rosu, Chunyuan Deng, Bolun Sun, Zihao Lin, and Bhuwan Dhingra. Knowing when to stop: Efficient context processing via latent sufficiency signals. In *Advances in Neural Information Processing Systems*, 2025.

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*, 2018.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, 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.

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: Human Language Technologies*, 2021.

A Compression and Restoration Protocol

The primary comparison uses a matched realized compression ratio of $4\times$; Appendix B reports $6\times$ and $8\times$ stress tests for LLMLingua-2. Documents are partitioned into fixed, non-overlapping context chunks shared by all policies. LCLM uses its native segment-to-latent correspondence. For AutoCompressor, we independently compress each chunk so that each summary-vector block has a unique context span. For LLMLingua-2, retained tokens inherit their context-chunk identifiers. Restoring chunk i removes its compressed representation and inserts the original $\mathbf{x}_i$; every subsequent answer is regenerated and charged to the trajectory cost. Because the released AutoCompressor checkpoint has a shorter raw-context window, its full-context reference and risk analysis use only examples whose uncompressed inputs fit that window. The Qwen3-4B-Instruct-2507 experiments form the primary full-length evaluation.

B Compression-Ratio Stress Test

We test whether RCPE remains useful as the initial compressed representation becomes increasingly lossy. The primary stress test uses Qwen3-4B-Instruct-2507 with LLMLingua-2 at realized

Table 3: Task performance and inference cost under increasing compression ratios for LLMLingua-2–Qwen3-4B-Instruct-2507 at $(\epsilon, \delta, \kappa) = (0.10, 0.20, 0.80)$ and $\alpha = 0.05$. All rows use the intersection of examples supported at every ratio. LongBench columns are task families, ZeroS is its official-metric aggregate, and AVG averages the six LongBench families and ZeroS. Each RCPE row is independently calibrated and certified at the stated operating point. "Fixed" uses the compressor without restoration; cost is normalized to full-context inference. The corresponding $4\times$ results are reported in Table 1.

Method	Ratio	SingleDoc	MultiDoc	Summ.	FewShot	Synth.	Code	ZeroS	AVG	Restored (%)	Norm. cost
Full context	$1\times$	40.8	39.5	27.1	68.4	39.6	55.8	33.1	43.5	–	1.00
Fixed	$6\times$	36.0	38.1	26.4	66.7	24.0	55.6	32.2	39.9	0	0.28
RCPE	$6\times$	**39.3**	**39.2**	**26.8**	**67.4**	**34.5**	**55.7**	**33.0**	**42.3**	15.2	0.43
Fixed	$8\times$	34.9	36.8	26.0	65.2	20.8	54.0	30.6	38.3	0	0.24
RCPE	$8\times$	**38.4**	**38.5**	**26.5**	**66.6**	**32.0**	**54.8**	**32.1**	**41.3**	21.8	0.49

compression ratios of $4\times$, $6\times$, and $8\times$. To avoid changes in dataset composition confounding the comparison, all ratios for a compression interface are evaluated on the intersection of examples supported at every reported ratio. Full-context outputs, prompts, decoding parameters, context chunks, and data splits are held fixed across ratios.

For each ratio, we reconstruct the training trajectories and independently train and calibrate the controller. The policy family and operating targets are held fixed at $(\epsilon, \delta, \kappa) = (0.10, 0.20, 0.80)$ and $\alpha = 0.05$; the restoration budget remains the same fraction of the uncompressed context length. This protocol measures performance at a matched risk–coverage operating point rather than transferring a threshold tuned for one compression ratio to another. We report performance for every LongBench task family and ZeroS, together with the restored-token fraction and normalized trajectory cost. Table 3 uses "uncertified" in the RCPE row when no policy in the frozen family satisfies all three calibration constraints.

The results show that RCPE remains effective as compression becomes more aggressive. At $6\times$, RCPE improves the fixed-compression AVG from 39.9 to 42.3; at $8\times$, it improves AVG from 38.3 to 41.3. The gain therefore grows from 2.4 to 3.0 points as the initial representation becomes more lossy. The largest improvements occur on synthetic tasks, where RCPE adds 10.5 and 11.2 points at $6\times$ and $8\times$, respectively, while the gains on summarization and code completion are smaller. More aggressive compression nevertheless leaves a larger residual gap to full-context inference: RCPE is 1.2 AVG points below full context at $6\times$ and 2.2 points below at $8\times$. This additional recovery also requires more expansion, increasing the restored-token fraction from 15.2% to 21.8% and normalized trajectory cost from 0.43 to 0.49, both remaining below full-context cost.

C Data Sources, Codebases, Checkpoints, and RCPE Implementation

Controller-training data sources. We train the RCPE estimator heads using the official training partitions of QASPER (Dasigi et al., 2021), HotpotQA (Yang et al., 2018), QMSum (Zhong et al., 2021), and NarrativeQA (Kočiský et al., 2018). Together, these sources cover scientific-document QA with evidence annotations, multi-document and multi-hop QA with supporting facts, query-focused meeting summarization, and long-narrative QA. We convert every source example to the same chunk–compress–restore interface used at evaluation time and construct the reliability and counterfactual restoration labels with the frozen target backbone. Only the source training partitions are used to form the three-way controller data pool; their official validation and test partitions are excluded. Before trajectory construction, we remove examples whose normalized document hash matches a LongBench or ZeroSCROLLS evaluation document. All states and action labels derived from one source example remain in the same controller-training, controller-validation, or policy-validation split.

Baseline codebases. We build the textual-compression baselines from the official LLMLingua repository, which contains LLMLingua, LongLLMLingua, and LLMLingua-2: `microsoft/LLMLingua`. We use the official LCLM implementation and its model-

loading utilities from `LeonLixyz/LCLM`, and the official AutoCompressor implementation from `princeton-nlp/AutoCompressors`. Benchmark loading, prompts, and metric implementations are based on the official LongBench and ZeroSCROLLS repositories `THUDM/LongBench` and `tau-nlp/zero_scrolls`, respectively. Our BM25 baseline implements Okapi BM25 over exactly the same context chunks exposed to the other methods and restores chunks in descending retrieval-score order until reaching the matched restored-token budget; it does not use query expansion, reranking, or task labels. For every external baseline, our changes are confined to thin adapters that expose context-chunk identifiers, apply the common prompt and decoding interface, and record complete-trajectory costs. We do not retrain or modify the released compression models.

Hugging Face checkpoints. For full-context, BM25, and textual-compression runs in the Qwen3-4B-Instruct-2507 group, we load `Qwen/Qwen3-4B-Instruct-2507`. The released $4\times$ LCLM system is loaded as `latent-context/0.6b-4b-LCLM-4x` using the official LCLM loader; its packaged decoder is `Qwen/Qwen3-4B-Instruct-2507`, and we do not replace that decoder. For the Llama-2-7B group, the backbone is `meta-llama/Llama-2-7b-hf`, and the latent compressor is `princeton-nlp/AutoCompressor-Llama-2-7b-6k`. LLMLingua-2 uses the released token-classification checkpoint `microsoft/llmlingua-2-xlm-roberta-large-meetingbank`. The frozen RCPE state encoder is `Qwen/Qwen3-Embedding-0.6B`.

Our RCPE implementation. RCPE is implemented in PyTorch and Hugging Face Transformers as a separate controller around frozen generator and compressor adapters. A compressor adapter exposes `compress`, `restore`, and `serialize` operations while preserving a one-to-one mapping between each compressed block and its original context chunk. The offline trajectory builder caches full-context and fully compressed generations, constructs partial-restoration states, executes the add-one counterfactual rollouts used to label $\Delta_{t,i}$, and records generation, context-access, and restored-token costs. The estimator module contains the frozen shared state encoder, two reliability MLPs, and the candidate-conditioned utility head described in Appendix E; all available candidate utilities at a state are evaluated in one padded batch. The policy runner regenerates the answer after every selected restoration and passes the resulting state back to the controller. Finally, a separate calibration program replays complete validation trajectories for every member of the fixed policy grid, computes the simultaneous bounds in Appendix G, and selects the lowest-cost certified policy without consulting test outcomes. Dataset splits, cached generations, controller checkpoints, policy grids, seeds, and per-stage cost logs are stored as explicit artifacts so that each reported row can be reproduced from a single configuration.

D Counterfactual Trajectory Construction

All splits are formed at the example level before trajectory generation, so no two states derived from the same context occur in different splits. For each training example $(\mathbf{X}, \mathbf{q}, \mathbf{y}^*)$, we first cache the full-context and fully compressed predictions and their task scores. We then construct partial restoration states using annotated-evidence order when available, retriever order, reverse-retriever order, and random order. Online rollouts under periodically updated utility checkpoints add states resembling those visited by the learned policy.

Each retained state records $(\mathbf{q}, \mathbf{M}_t, \mathcal{S}_t, \mathbf{y}_t, a_t, a_{\text{full}})$. For an available chunk i, an add-one rollout regenerates the answer after restoring only that chunk and provides the target $\Delta_{t,i} = a^+_{t,i} - a_t$. The labeled action set therefore contains beneficial, neutral, and harmful restorations. When a high-scoring state is formed by deleting a restored chunk, the resulting state is treated as a new state and the corresponding add-back action is labeled by the same definition; deletion scores are not used with an unmatched sign convention.

Training-only hard cases remove a required chunk, replace a decisive entity or number, insert a plausible contradiction, or preserve query terms while breaking the relation needed to answer. These augmentations are never applied to calibration or test examples. Utility-head optimization samples at most 16 labeled available actions per state uniformly without replacement, as detailed in Appendix E.

E Estimator Architecture and Training

This section specifies the implementation of the learned estimators separately from the backbone that generates answers. All estimator architectures and optimization choices are fixed before policy calibration.

Shared State Encoder. We use the Hugging Face `Qwen/Qwen3-Embedding-0.6B` checkpoint (Zhang et al., 2025) as the shared state encoder E. All pretrained encoder parameters are frozen; only the prediction heads and input projections described below are trained. Text fields are serialized as the query $\mathbf{q}$, current answer $\mathbf{y}_t$, restored text $\mathbf{R}_t$, and the still-compressed memory $\mathbf{M}_t$, separated by field markers. For textual compressors, retained tokens appear in context order. For latent compressors, the vectors within each memory block $\mathbf{m}_i$ are mean-pooled and mapped to the Qwen encoder hidden dimension by a trainable linear projection; the projected blocks are inserted at their context positions as pseudo-token embeddings. The current restoration fraction, answer length, retrieval-score summary, and answer-uncertainty statistics are normalized on the training split and projected to the same hidden dimension as additional feature embeddings. We use the final-layer representation of the last non-padding token as the pooled state representation $\mathbf{r}_t$.

Relative- and Absolute-Reliability Heads. The two reliability estimators are separate multilayer perceptrons applied to the same $\mathbf{r}_t$. Each is a three-layer MLP with dimensions $1024 \to 512 \to 128 \to 1$. The first two layers use GELU activations, layer normalization, and dropout 0.1; the final layer is followed by a sigmoid. The relative head is trained toward $1 - [a_{\text{full}} - a_t]_+$, whereas the absolute head is trained toward a_t. Thus, the heads share the state encoder but do not share their final prediction layers. The latency of the shared encoder and reliability heads is included in state assessment, while the candidate utility pass is reported separately in Appendix H.2.

Restoration-Utility Head. For every available chunk $i \notin \mathcal{S}_t$, we construct a candidate representation from its compressed block $\mathbf{m}_i$ and metadata $\boldsymbol{\rho}_i$. The metadata comprise normalized context position, log-scaled chunk length, normalized retrieval score, normalized retrieval rank, and two binary indicators specifying whether the immediately preceding and following chunks have been restored. Each scalar feature is normalized using training-split statistics and projected to the model dimension.

The utility module contains two trainable cross-attention layers with hidden dimension 1024, 16 attention heads, GELU feed-forward blocks, and dropout 0.1. The state representation supplies the query, while the projected candidate memory and metadata supply keys and values. The resulting candidate-conditioned state is mapped to a scalar by a linear projection. A $\tanh$ output produces $u_{t,i} \in [-1, 1]$. It is supervised by the measured counterfactual gain $\Delta_{t,i} = a^+_{t,i} - a_t$. Candidate scores within a state are computed in one batched forward pass. The reported estimator overhead includes this pass for all available chunks, not only the selected chunk.

Optimization and Model Selection. Training examples are grouped by example so that states derived from the same context never cross data splits. The global mini-batch contains eight states distributed across four GPUs. For each state, we sample at most 16 available candidate actions uniformly without replacement; all actions are used when fewer than 16 remain. We optimize the joint loss in Equation 15. The validation score is the negative joint loss, and we retain the checkpoint with the highest validation score. Training stops after ten epochs or after two consecutive epochs without improvement. Table 4 summarizes the configuration.

Three-way Split and Calibration Separation. The official source training partitions are repartitioned at the example level into mutually exclusive controller-training, controller-validation, and policy-validation subsets. The controller-training subset is used for gradient updates, while controller validation determines early stopping, selects the estimator checkpoint, and constructs the finite policy family Θ. After the estimators and Θ are frozen, the policy-validation subset serves as the independent calibration split used to construct simultaneous risk and coverage bounds and select $\hat{\theta}$. Reference answers, full-context outputs, a_t, a_{full}, and $\Delta_{t,i}$ are never supplied to the estimators at test time.

Table 4: RCPE estimator training configuration.

Configuration	Value
Pretrained encoder	`Qwen/Qwen3-Embedding-0.6B`
Tokenizer	Tokenizer from the same checkpoint
Encoder pooling / training	Last-token pooling; encoder frozen
Reliability heads	3-layer MLP, 1024–512–128–1
Utility head	2 cross-attention layers + linear projection
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$
Learning rate / scheduler	10^{-4} / linear decay
Warmup / weight decay	10% / 0.01
Global batch / gradient accumulation	8 / none
Epochs / early stopping	10 / patience 2
Gradient clipping / precision	1.0 / bfloat16
Loss weights (β, γ)	$(1, 1)$
Candidate-action sampling	Uniform, at most 16 per state
Random seed	42
Training hardware	4× NVIDIA A100 GPUs

Table 5: RCPE controller candidate grid, constructed on the controller-validation split and frozen before calibration.

Policy parameter	Candidates	Interpretation
τ_{rel}	$\{Q_{s^{\mathrm{rel}}}(0.50), Q_{s^{\mathrm{rel}}}(0.70), Q_{s^{\mathrm{rel}}}(0.85), Q_{s^{\mathrm{rel}}}(0.95)\}$	Relative-answer threshold
τ_{abs}	$\{Q_{s^{\mathrm{abs}}}(0.50), Q_{s^{\mathrm{abs}}}(0.70), Q_{s^{\mathrm{abs}}}(0.85), Q_{s^{\mathrm{abs}}}(0.95)\}$	Absolute-answer threshold
τ_u	$\{0, Q_{v\|v>0}(0.25), Q_{v\|v>0}(0.50), Q_{v\|v>0}(0.75)\}$	Restore threshold
b	$\{1, 2, 4\}$	Chunks per round
$B_{\max}$	$\{0.10, 0.25, 0.50, 1.00\}\,\lvert\mathbf{X}\rvert$	Restore budget

F Policy Grid and Execution

F.1 Policy Grid

Table 5 reports the policy grid used for RCPE. Let $Q_z(q)$ denote the empirical q-quantile of score z on the controller-validation split. For the utility threshold, quantiles are computed over positive cost-adjusted utilities $v_{t,i} > 0$. The maximum budget is expressed as a fraction of the original context-token count $|\mathbf{X}|$.

The raw Cartesian product contains $4 \times 4 \times 4 \times 3 \times 4 = 768$ candidate policies. Before accessing the policy-validation split, we execute these policies on controller-validation trajectories and remove behaviorally identical candidates and candidates that are dominated in both cost and empirical constraint satisfaction. The resulting finite family Θ is then frozen. Policy validation evaluates every remaining policy, retains only those whose simultaneous bounds satisfy Equation 17, and selects the feasible policy with the lowest estimated cost. The requested operating targets $(\epsilon, \delta, \kappa)$ and the total failure probability α are fixed separately and are not elements of Θ.

F.2 Execution Algorithm

Algorithm 1 gives the complete test-time controller for a selected policy $\hat{\theta} = (\tau_{\mathrm{rel}}, \tau_{\mathrm{abs}}, \tau_u, b, B_{\max})$.

Algorithm 1 Risk-Controlled Progressive Expansion at test time

Require: Query $\mathbf{q}$, compressed memory $\mathbf{M}_0$, context chunks $\{\mathbf{x}_i\}_{i=1}^{k}$, calibrated policy $\hat{\theta}$
Ensure: An answer or ABSTAIN

$t \leftarrow 0, \mathcal{S}_0 \leftarrow \varnothing, \mathbf{R}_0 \leftarrow \varnothing, B_0 \leftarrow 0$
loop
  $\mathbf{y}_t \leftarrow F(\mathbf{q}, \mathbf{M}_t, \mathbf{R}_t)$
  $\mathbf{r}_t \leftarrow E(\mathbf{q}, \mathbf{M}_t, \mathbf{R}_t, \mathbf{y}_t)$
  $(s_t^{\mathrm{rel}}, s_t^{\mathrm{abs}}) \leftarrow (g_\psi^{\mathrm{rel}}(\mathbf{r}_t), g_\eta^{\mathrm{abs}}(\mathbf{r}_t))$
  $\mathcal{U}_t \leftarrow [k] \setminus \mathcal{S}_t$
  Compute $u_{t,i}$ and $v_{t,i} = u_{t,i}/c_{t,i}$ for all $i \in \mathcal{U}_t$ in one batched utility pass
  if $s_t^{\mathrm{rel}} \geq \tau_{\mathrm{rel}}$ **and** $s_t^{\mathrm{abs}} \geq \tau_{\mathrm{abs}}$ **then**
    return $\mathbf{y}_t$
  else if $s_t^{\mathrm{rel}} \geq \tau_{\mathrm{rel}}$ **then**
    return ABSTAIN ▷ Low absolute quality without a compression deficit
  end if
  $\mathcal{A}_t \leftarrow \{i \in \mathcal{U}_t : v_{t,i} \geq \tau_u,\ B_t + c_{t,i} \leq B_{\max}\}$
  Sort $\mathcal{A}_t$ by decreasing $v_{t,i}$, increasing retrieval rank, then increasing context position
  $\mathcal{H}_t \leftarrow \varnothing, C_t \leftarrow 0$
  for all $i \in \mathcal{A}_t$ in sorted order **do**
    if $|\mathcal{H}_t| < b$ **and** $B_t + C_t + c_{t,i} \leq B_{\max}$ **then**
      $\mathcal{H}_t \leftarrow \mathcal{H}_t \cup \{i\}$
      $C_t \leftarrow C_t + c_{t,i}$
    end if
  end for
  if $\mathcal{H}_t = \varnothing$ **then**
    return ABSTAIN
  end if
  $(\mathbf{M}_{t+1}, \mathbf{R}_{t+1}) \leftarrow \mathrm{RESTORE}(\mathbf{M}_t, \mathbf{R}_t, \{\mathbf{x}_i : i \in \mathcal{H}_t\})$
  $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \mathcal{H}_t, B_{t+1} \leftarrow B_t + C_t, t \leftarrow t + 1$
end loop

RESTORE removes the selected compressed blocks and inserts their original chunks in context order. The deterministic secondary ordering by retrieval rank and context position resolves utility ties. Full-context predictions, references, and task-score labels are used only offline and never enter Algorithm 1.

G CALIBRATION BOUNDS AND PROOF

Let the calibration set contain n i.i.d. examples from the target test distribution, and let $M = |\Theta|$. For each fixed $\theta \in \Theta$ and calibration example j, define

$$Z_{j,\theta}^{\mathrm{comp}} = D_{j,\theta}[a_{j,\mathrm{full}} - a_{j,T_\theta}]_+, \qquad W_{j,\theta}^{\mathrm{comp}} = D_{j,\theta} a_{j,\mathrm{full}}, \tag{19}$$

$$Z_{j,\theta}^{\mathrm{abs}} = D_{j,\theta}(1 - a_{j,T_\theta}), \qquad W_{j,\theta}^{\mathrm{abs}} = D_{j,\theta}. \tag{20}$$

All four variables lie in $[0, 1]$. Denote their empirical means by $\widehat{\mu}_{Z_c}$, $\widehat{\mu}_{W_c}$, $\widehat{\mu}_{Z_a}$, and $\widehat{\mu}_{W_a}$, with the dependence on θ suppressed. We use simultaneous one-sided Hoeffding bounds with

$$r_{n,M,\alpha} = \sqrt{\frac{\log(4M/\alpha)}{2n}}. \tag{21}$$

For policies whose denominator lower bounds are positive, the reported bounds are

$$U_{\mathrm{comp}}(\theta) = \min\left\{1, \frac{\widehat{\mu}_{Z_c} + r_{n,M,\alpha}}{\widehat{\mu}_{W_c} - r_{n,M,\alpha}}\right\}, \tag{22}$$

$$U_{\mathrm{abs}}(\theta) = \min\left\{1, \frac{\widehat{\mu}_{Z_a} + r_{n,M,\alpha}}{\widehat{\mu}_{W_a} - r_{n,M,\alpha}}\right\}, \tag{23}$$

$$L_{\mathrm{cov}}(\theta) = \max\{0, \widehat{\mu}_{W_a} - r_{n,M,\alpha}\}. \tag{24}$$

Table 6: Repeated-calibration coverage audit for $4\times$ LCLM–Qwen3-4B-Instruct-2507 on LongBench with $(\delta, \kappa) = (0.20, 0.80)$, $\alpha = 0.05$, and $B = 100$ calibration resamples. Violation rates are computed among repetitions for which a policy is certified; parentheses contain 95% intervals. "Any" denotes the union in Equation 25.

Target ϵ	Certified (%)	Any viol. (%)	Comp. viol. (%)	Abs. viol. (%)	Cov. viol. (%)	Test R_{comp}	Test coverage	Norm. cost
0.05	78.0 (68.9–85.0)	3.8 (1.3–10.7)	2.6 (0.7–8.9)	0.0 (0.0–4.7)	1.3 (0.2–6.9)	0.041 (0.036–0.046)	0.821 (0.812–0.830)	0.62 (0.57–0.68)
0.10	93.0 (86.1–96.5)	3.2 (1.1–9.0)	2.2 (0.6–7.5)	1.1 (0.2–5.8)	0.0 (0.0–4.0)	0.081 (0.074–0.088)	0.839 (0.830–0.848)	0.47 (0.43–0.52)
0.15	99.0 (94.6–99.8)	2.0 (0.6–7.1)	1.0 (0.2–5.5)	1.0 (0.2–5.5)	0.0 (0.0–3.7)	0.126 (0.118–0.134)	0.856 (0.848–0.865)	0.36 (0.33–0.40)

Audit metadata: calibration size $n = 2{,}000$, held-out size $n_{\mathrm{test}} = 2{,}000$, and frozen policy-family size $M = 192$.

A policy is declared uncertifiable if either denominator lower bound is nonpositive. Clipping an upper risk bound at one is valid because both risks are themselves in $[0, 1]$.

For each policy, Hoeffding's inequality bounds the failure probability of each required one-sided mean inequality by $\alpha/(4M)$. A union bound over the four mean inequalities and all M policies gives simultaneous validity with probability at least $1 - \alpha$. On this event, the numerator upper bounds and positive denominator lower bounds imply $R_{\mathrm{comp}}(\theta) \le U_{\mathrm{comp}}(\theta)$ and $R_{\mathrm{abs}}(\theta) \le U_{\mathrm{abs}}(\theta)$, while $\mathrm{Cov}(\theta) \ge L_{\mathrm{cov}}(\theta)$ for every candidate. Consequently every policy satisfying Equation 17 satisfies the population constraints, and selecting the lowest-cost feasible policy preserves the guarantee in Proposition 1.

G.1 Repeated-Calibration Coverage Audit

The single-split audit in Table 2 shows the behavior of the policy used for the main evaluation, but it does not reveal how often the calibration procedure succeeds across alternative calibration samples. We therefore conduct a repeated-split audit with the estimator, trajectory generator, and candidate policy family Θ frozen throughout. For each target $\epsilon \in \{0.05, 0.10, 0.15\}$, we draw $B = 100$ example-level calibration samples of the same size n as the main calibration split. The remaining examples in the audit pool are used only to estimate the selected policy's held-out risk and coverage. States derived from the same document are always assigned together, and the same set of split seeds is used for all three targets.

In repetition b, calibration either returns the minimum-cost certified policy $\hat{\theta}_b$ or declares the target uncertifiable. Among certified repetitions, we record violations of each held-out constraint and their union,

$$V_b = \mathbf{1}\Big\{ R_{\mathrm{comp}}(\hat{\theta}_b) > \epsilon \ \vee \ R_{\mathrm{abs}}(\hat{\theta}_b) > \delta \ \vee \ \mathrm{Cov}(\hat{\theta}_b) < \kappa \Big\}. \tag{25}$$

We report the certification rate, the empirical rates of simultaneous and component-wise violations, and the mean normalized cost and coverage of the selected policies. Binomial proportions are accompanied by two-sided 95% Wilson intervals across repetitions; cost and coverage use percentile bootstrap intervals over repetitions. Because held-out risks are themselves finite-test estimates, this experiment is an empirical audit of repeated-sample behavior, not an additional finite-sample guarantee. We additionally report n, the held-out audit size, and the number of policies M so that the effective Hoeffding radius in Equation 21 is explicit.

The repeated-calibration audit provides empirical evidence on behavior across alternative calibration samples. As the compression-risk target is relaxed from 0.05 to 0.15, the certification rate rises from 78% to 99%, while mean normalized cost falls from 0.62 to 0.36 and coverage increases from 0.821 to 0.856. The selected policies use the additional risk allowance—mean test compression risk rises from 0.041 to 0.126—but remains below the corresponding target. Conditional simultaneous violation rates range from 2.0% to 3.8%. These rates are descriptive: Proposition 1 controls the unconditional probability over calibration samples, whereas the table reports violations only among certified repetitions. Because only 100 resamples are used, the Wilson intervals for rare violations remain wide and should be considered alongside the point estimates.

H Evaluation and Cost-Accounting Details

H.1 Evaluation Details

Evaluation is performed at the level of complete policy trajectories. The cost of an example includes initial compression, the first answer generation, every state-encoder and batched utility-head evaluation, context access and processing for restored chunks, and every subsequent regeneration. Unsuccessful restorations and trajectories that ultimately abstain are therefore not free. Normalized trajectory cost uses full-context inference for the same backbone and example set as the reference value 1.00.

Comparisons at a matched operating point use the same example split, chunk boundaries, decoding parameters, restoration budget, and risk and coverage targets. Full-context outputs and reference answers are used to score completed trajectories and to construct offline labels, but are excluded from test-time controller inputs. Compressor-specific context-window filtering is applied before comparing methods.

H.2 Latency and Memory Measurement Protocol

Unless otherwise stated, inference is measured on a single NVIDIA A100 80GB GPU with batch size one and bfloat16 model weights. We disable inter-example batching so that adaptive trajectories with different numbers of restoration rounds are timed independently. Each model is loaded once, followed by 20 unmeasured warm-up trajectories. We then run the evaluation set three times. For each example, stage latency is aggregated over all invocations of that stage (and is zero when the stage is not executed), and the three repeated measurements are averaged. Complete-trajectory latency is formed by summing stages within each example; its mean, median, and 95th percentile are then computed directly across examples rather than by adding component quantiles. CUDA is synchronized immediately before and after every timed region. Dataset loading is excluded, whereas host-to-device transfer, initial compression, context access, controller evaluation, restoration, and all generation rounds are included.

Memory is measured in a profiling pass separate from latency measurement. The complete-trajectory peak is measured during one uninterrupted execution after resetting allocator peak statistics immediately before the trajectory. For a stage-local profile, we replay the prefix needed to reach that stage, release unused cached blocks, reset peak statistics immediately before the target stage, and record the maximum over all invocations and examples. Stage-local peaks can therefore decrease between rows and are not additive; they are not used to derive the complete-trajectory peak. We report both allocated and reserved memory, together with the resident parameter footprint measured after model loading. For RCPE, the complete peak includes the compressor and backbone states, the shared estimator encoder, both reliability heads, the batched candidate utility pass, restored context tokens, and the active generation KV cache. The same measurement boundaries are used for every baseline.

Table 7 reports the measured latency and GPU-memory breakdown for the $4\times$ LCLM–Qwen3-4B-Instruct-2507 RCPE configuration.

The breakdown attributes most of RCPE's latency to initial and post-restoration generation. The controller itself—state assessment plus the batched utility pass—accounts for 0.16 seconds on average. The complete trajectory takes 1.97 seconds on average (1.88-second median and 2.98-second 95th percentile), with peak allocated and reserved memory of 17.6 GB and 19.4 GB, respectively.

I Prompt, Decoding, and Diagnostic Specification

All methods used the same task instruction and decoding configuration; only the context representation differed. We used greedy decoding with one beam, temperature zero, top-p equal to one, and no sampling. Maximum generation length and stop strings followed the official configuration for each LongBench or ZeroSCROLLS task. Restored chunks replaced their compressed blocks at the corresponding context positions rather than being appended after the query.

The exact question-answering and retrieval prompt used in our experiments was:

Table 7: Latency and GPU-memory breakdown for $4\times$ LCLM–Qwen3-4B-Instruct-2507 with RCPE. Repeated stages aggregate all invocations. Complete-trajectory quantiles are computed from per-example total times, not sums of component quantiles. Memory columns report separately measured stage-local peaks and are not additive.

Component	Mean (s)	Median (s)	P95 (s)	Peak alloc. (GB)	Peak reserved (GB)
Compression	0.31	0.29	0.48	11.6	12.8
Initial generation	0.72	0.67	1.08	15.4	16.9
State assessment	0.11	0.10	0.17	12.4	13.6
Utility scoring	0.05	0.05	0.08	12.9	14.1
Restoration	0.03	0.03	0.05	13.8	15.0
Regeneration	0.75	0.70	1.35	17.3	19.0
Complete trajectory	**1.97**	**1.88**	**2.98**	**17.6**	**19.4**

Resident parameter footprint: 10.8 GB. The complete trajectory includes unsuccessful restorations and trajectories that ultimately abstain.

```
Read the provided context and answer the query.  Use a
concise answer and do not add explanations unless the task
requests them.
<CONTEXT>
{compressed context with restored chunks inserted in context
order}
</CONTEXT>
<QUERY> {query} </QUERY>
<ANSWER>
```

The exact summarization prompt used in our experiments was:

```
Summarize the provided document according to the task
instruction.  Preserve the central facts and do not
introduce information absent from the document.
<DOCUMENT>
{compressed document with restored chunks inserted in
context order}
</DOCUMENT>
<TASK> {dataset-specific summarization instruction} </TASK>
<SUMMARY>
```

For classification, synthetic, few-shot, and code-completion tasks, we used the official benchmark prompt and replaced only its context field with the current RCPE state. Classification prompts requested only a valid label, while code-completion prompts preserved the original prefix and stop sequence. The state encoder used a separate serialization consisting of the query, current answer, restored text, and still-compressed memory, each preceded by a fixed field marker. Latent memory blocks were injected as projected pseudo-token embeddings at the positions occupied by their context chunks.

The failure-stage analysis assigns each unsuccessful trajectory to the earliest stage identified by controlled reruns: trigger, selection, stopping, evidence utilization, or backbone failure. This ordering prevents a later consequence from being counted as the initiating error. Estimator latency is reported for the shared state encoder, the two reliability heads, and the batched utility pass, while end-to-end latency additionally includes compression, context restoration, and all generation rounds.