ADAMEM: DISTORTION-GUIDED ADAPTIVE CONTEXT COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Soft context compression improves long-context inference efficiency by replacing raw contexts with compact continuous memory representations. Existing methods, however, either use fixed compression budgets or adapt them according to indirect signals such as relevance, redundancy, or information density, without explicitly modeling the quality degradation induced by different compression ratios. We formulate context compression as a quality-constrained memory allocation problem and propose AdaMem, a distortion-guided framework that adaptively allocates memory budgets to individual context chunks according to their predicted compression sensitivity. Specifically, AdaMem employs a shared multi-ratio compressor to provide multiple compression budgets. Based on prefix representations, it predicts the distortion of each candidate ratio and selects the most aggressive ratio that satisfies a local predicted-distortion threshold. AdaMem further enables ratio selection before memory construction while reusing the prefix KV cache. Experiments on MRQA show that AdaMem consistently improves the quality–compression trade-off over fixed-ratio and adaptive baselines, reducing exported memory representations by 73.18% in-domain and 62.48% out-of-domain relative to fixed $5\times$ compression, with consistent gains across 1B–8B model scales.

1 INTRODUCTION

Many applications require large language models (LLMs) to process substantial context. Examples include question answering over long documents, multi-turn dialogue, and retrieval-augmented generation. Context compression reduces the number of input representations by replacing the original text with shorter discrete sequences or compact continuous memories. The central challenge is to determine how aggressively each part of the context can be compressed while preserving the information needed to answer the query.

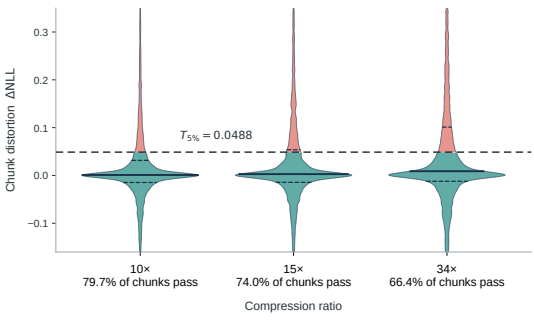

Figure 1: Chunks differ in compression sensitivity. Dashed line: 5% local perplexity tolerance.

Context compression methods differ in both representation and budget allocation. Hard compression selects, prunes, or rewrites text (Jiang et al., 2023; Pan et al., 2024), whereas soft compression produces continuous memory representations (Ge et al., 2023; Li et al., 2025; Zhao et al., 2024). Many soft compressors apply a fixed compression ratio across chunks. Query-conditioned methods adapt the encoded content to the current query, typically under a fixed output capacity (Cao et al., 2024; Louis et al., 2026). Other methods redistribute a prescribed global budget using relevance, redundancy, or information density (Chen et al., 2025; Tang et al., 2026). However, these allocation signals do not directly estimate the quality change caused by assigning a particular chunk each candidate compression ratio.

Content importance alone does not determine how strongly a chunk can be compressed. Figure 1 illustrates this distinction using 8,024 chunks from 5,000 held-out MRQA examples. We change

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.

one chunk's compression ratio at a time, keep all other chunks at $5\times$, and measure the change in token-averaged reference-answer negative log-likelihood (NLL). We use this NLL change as the local distortion signal because it captures how compression changes the likelihood of the target answer. At $34\times$, 66.4% of chunks remain below the 5% local tolerance, whereas 33.6% exceed it. A uniform ratio therefore assigns the same capacity to chunks with markedly different measured sensitivities. This observation motivates our central question: *can predicted compression-induced distortion guide the memory budget of each chunk?*

AdaMem answers this question by making compression sensitivity an explicit routing signal. A shared multi-ratio compressor supports candidate ratios of $5\times$, $10\times$, $15\times$, and $34\times$. The measured NLL changes provide supervision for a lightweight predictor, which estimates the distortion profile at inference time from query-conditioned prefix representations. Given a user-specified tolerance τ, Pathwise routing selects the largest ratio for which all candidate ratios up to that point remain below the corresponding local predicted-distortion threshold. Because the predictor estimates the complete distortion profile, τ can be adjusted without refitting the predictor, providing direct control over the quality–representation trade-off. AdaMem thus selects the output budget from a local quality proxy rather than requiring a fixed global budget in advance.

To resolve the circular dependency between ratio selection and ratio-specific memory construction, AdaMem separates prefix encoding from memory construction. The prefix stage supplies both the representations used by the predictor and a reusable KV cache. The memory stage encodes a shared token bank with the selected position layout and exports the corresponding subset to the downstream model. This design couples data-dependent allocation with one compressor shared across all ratios, without evaluating a separate compression forward pass for every candidate.

Experiments on MRQA assess the quality–representation trade-off in- and out-of-domain across 1B–8B backbones. At the default tolerance, AdaMem reaches nominal compression ratios of $18.65\times$ in-domain and $13.33\times$ out-of-domain, exporting 73.18% and 62.48% fewer representations than fixed $5\times$ while exceeding the strongest reproduced soft-compression baseline in F1 and EM. We additionally evaluate transfer to long-context QA on LongBench without benchmark-specific training or tuning.

Our contributions are summarized as follows:

1. We characterize chunk-level heterogeneity in compression sensitivity and formulate adaptive soft compression as per-chunk compression-ratio selection under a local predicted-distortion constraint.
2. We propose AdaMem, which combines a shared multi-ratio compressor, distortion-guided routing, and split execution to select the compression ratio before memory construction.
3. Experiments establish favorable quality–representation operating points across MRQA domains and backbone sizes, complemented by full-context QA transfer evaluation on LongBench.

2 Related Work

2.1 Hard Compression Methods

Hard compression selects, prunes, merges, or rewrites discrete text using information measures, attention scores, learned selectors, or reinforcement learning (Li et al., 2023; Zhao et al., 2025; Jung & Kim, 2024; Fei et al., 2026; Chuang et al., 2024; Xu et al., 2025; Hu et al., 2026; Liskavets et al., 2025a). LLMLingua (Jiang et al., 2023) uses budget-controlled coarse-to-fine compression; LLMLingua-2 (Pan et al., 2024) formulates task-agnostic compression as token classification.

Query-aware hard compression prioritizes content for the current question. LongLLMLingua (Jiang et al., 2024) uses question-aware context prioritization, while RECOMP (Xu et al., 2024) performs query-focused compression over retrieved documents. CompAct (Yoon et al., 2024), CPC (Liskavets et al., 2025b), Provence (Chirkova et al., 2025), Perception Compressor (Tang et al., 2025), and EXIT (Hwang et al., 2025) likewise use query relevance for sentence or context selection. AdaMem instead controls the capacity of continuous representations: it predicts how a candidate compression ratio changes an answer-conditioned loss, rather than selecting which text to retain.

2.2 Soft Compression Methods

Soft compression encodes context into continuous representations using learnable tokens, summary vectors, or memory slots (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2023; Wang et al., 2024; Tan et al., 2024). Later work develops context embeddings, high-ratio compression, position-aware memory layouts, and alternative encoder architectures (Rau et al., 2024; Dai et al., 2025; Li et al., 2025; Zhao et al., 2024; Zhang et al., 2025; Liao et al., 2025; Cheng et al., 2024; Gao et al., 2025). PISCO (Louis et al., 2025) and SAC (Liu et al., 2026) further study efficient training and autoencoding-free compression. These advances supply compact representations; AdaMem focuses on choosing their capacity per chunk.

Conditioning the representation and adapting its capacity are distinct choices. QGC (Cao et al., 2024) and OSCAR (Louis et al., 2026) construct query-dependent representations. COMI (Tang et al., 2026) redistributes a prescribed global budget using relevance and redundancy, while DAST (Chen et al., 2025) allocates soft tokens using information density. In contrast, AdaMem predicts the marginal answer-NLL change at explicit candidate ratios and selects each chunk's capacity under a local tolerance. Its aggregate budget emerges from these decisions rather than being fixed beforehand.

2.3 Representation Probing for Adaptive LLM Inference

Lightweight probes extract control signals from intermediate LLM representations for semantic routing, classification, and adaptive computation (Meyoyan & Del Corro, 2026; Ratnakar et al., 2026; Wu et al., 2025; Wang & Xu, 2025). AdaMem applies this principle to ratio selection: a distortion predictor uses learnable ratio queries to attend to the compressor's question-aware chunk states and estimates a distortion vector over candidate ratios. The predicted quantity is a continuous change in answer NLL, so the same predictor supports multiple tolerance settings without retraining. The split execution pathway makes these features available before the memory budget is instantiated.

3 Method

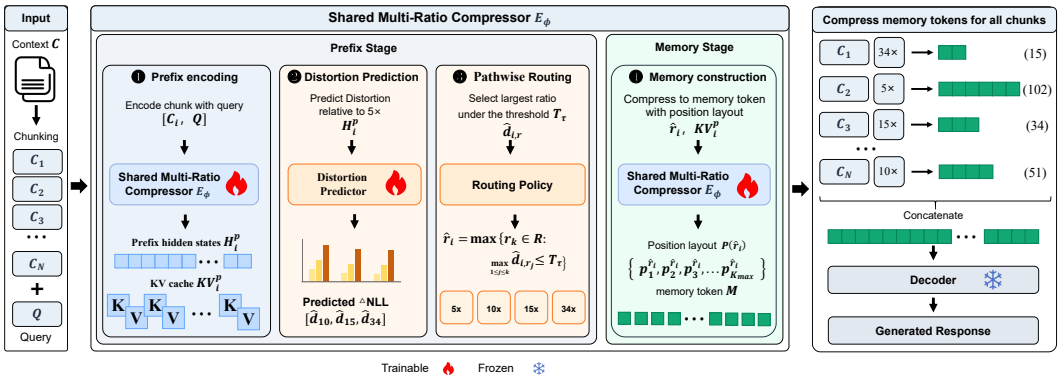

Figure 2: AdaMem predicts chunk distortion from query-conditioned prefix states, selects a ratio, and reuses the prefix KV cache to construct and export memory.

AdaMem selects a compression ratio for each chunk before memory construction (Figure 2). A shared multi-ratio compressor supports several candidate ratios, a distortion predictor estimates the local NLL change induced by each candidate, and split execution resolves the circular dependency between routing and ratio-specific memory construction.

3.1 Problem Formulation

Given a context–query pair $X = (C, Q)$, we partition the context into N chunks,

$$C = \{C_1, \ldots, C_N\}, \qquad |C_i| \le L.$$

Here, $L = 510$ is the fixed chunk size. AdaMem considers a discrete set of compression ratios $\mathcal{R} = \{5, 10, 15, 34\}$, where ratio r exports $K(r) = L/r$ memory representations. The four compression ratios correspond to $102, 51, 34$, and 15 memory tokens.

Let $d_i(r)$ denote the compression-induced change in answer NLL for chunk C_i relative to $5\times$, with $d_i(5) = 0$. For ordered candidates $5 = r_1 < r_2 < r_3 < r_4 = 34$ and a relative perplexity tolerance $\tau \geq 0$, Pathwise selection uses the NLL threshold $T_\tau = \ln(1 + \tau)$:

$$r_i^* = \max \left\{ r_k \in \mathcal{R} : \max_{1 \leq j \leq k} d_i(r_j) \leq T_\tau \right\}.$$

Every candidate up to the selected ratio must satisfy the threshold. Since $K(r)$ decreases with r, this rule minimizes $\sum_{i=1}^{N} K(r_i)$ under these independent per-chunk constraints. The deployed router substitutes predicted distortions. The constraints apply to a local NLL surrogate, not end-to-end answer quality; we evaluate the resulting allocations using downstream QA metrics.

3.2 Shared Multi-Ratio Compressor

AdaMem uses a single compressor E_ϕ with a shared bank of $K_{\max} = K(5) = 102$ learnable memory tokens,

$$M = \{m_1, \ldots, m_{K_{\max}}\}.$$

The compressor processes the same $K_{\max}$ memory tokens at every ratio; the ratio determines their logical position assignments and the number of exported representations. For ratio r, the last $K(r)$ memory tokens are assigned one complete set of logical position anchors uniformly distributed over the original chunk. Only the output representations of these tokens are exported to the frozen downstream LLM; the remaining memory tokens participate in compression but are not exported. Sharing parameters and memory-token embeddings enables a single compressor to operate under all four compression budgets, rather than requiring a separately trained model for each ratio. Appendix B.1 provides the complete position-assignment construction.

We use bidirectional attention within the context–query prefix and within the memory block, while prefix tokens do not attend to memory tokens. Memory tokens can therefore aggregate information from the complete prefix, whereas prefix representations can be computed and cached before the memory block is constructed. Detailed attention masks are provided in Appendix B.2.

To train the shared compressor across all candidate ratios, we use a curriculum that progressively expands the active ratio set from $\{5\}$ to $\mathcal{R}$. At each optimization step, one ratio is sampled from the active set and determines the logical position layout and export budget for that step. Appendix B.3 provides the complete curriculum schedule.

3.3 Distortion Prediction and Pathwise Routing

Rather than directly predicting a compression ratio, AdaMem estimates the distortion associated with each candidate ratio. For a multi-chunk QA example, let $\ell_{\text{all5}} = \mathcal{L}(5, \ldots, 5)$ be the teacher-forced answer NLL when all chunks use the $5\times$ reference ratio. The distortion label for chunk C_i at ratio $r \in \{10, 15, 34\}$ is

$$d_i(r) = \mathcal{L}(5, \ldots, r, \ldots, 5) - \ell_{\text{all5}},$$

where only C_i changes ratio and $\mathcal{L}$ averages natural-log NLL over reference-answer tokens. The target vector $d_i = [d_i(10), d_i(15), d_i(34)]$ is a measured distortion profile, not a ratio-class label. Negative entries indicate that the intervention lowers reference-answer NLL. Reference answers provide supervision for distortion-target construction and are used in offline evaluation and oracle analyses; the deployed router requires only the context and query.

At inference, the predictor reads $H^p_{i,C}$, the context-token subset of the compressor's final-layer prefix states H^p_i. Bidirectional prefix attention makes these states query-conditioned. Three learnable ratio queries, one for each non-reference candidate, attend to these states and output

$$\hat{d}_i = [\hat{d}_i(10), \hat{d}_i(15), \hat{d}_i(34)].$$

The predictor is trained directly against the measured distortion profiles, aligning its outputs with the ratio-specific quality changes used by the downstream routing policy. Additional details are provided in Appendix B.4.

Predicted distortions need not increase monotonically with the ratio. For the threshold $T_\tau = \ln(1+\tau)$, define $\hat{d}_i(r_1) = 0$ and the running maximum $g_i(r_k) = \max_{1\le j\le k} \hat{d}_i(r_j)$. Pathwise routing selects

$$\hat{r}_i = \max\{r_k \in \mathcal{R} : g_i(r_k) \le T_\tau\}.$$

Pathwise routing selects the largest admissible ratio for each chunk independently, requiring all preceding candidate ratios to satisfy the distortion threshold, while retaining $5\times$ as a fallback. The tolerance τ defines a local NLL-based quality constraint rather than a probabilistic guarantee on downstream degradation. Direct routing evaluates each candidate ratio independently; Appendix D.3 compares the two policies.

3.4 Decision-Before-Memory Execution

Adaptive routing creates an execution dependency because the selected ratio determines the memory-position layout, whereas routing itself depends on the current chunk representation. AdaMem resolves this through a split execution of the same compressor.

First, the context–query prefix is encoded without memory tokens,

$$(H_i^p, KV_i^p) = E_\phi^{\text{prefix}}([C_i; Q]),$$

and its context-token subset $H_{i,C}^p$ is used to predict $\hat{d}_i$ and select $\hat{r}_i$. After the ratio is determined, AdaMem constructs its logical position layout and reuses the prefix KV cache to encode the shared memory-token bank,

$$Z_i = E_\phi^{\text{memory}}(M; KV_i^p, P(\hat{r}_i)), \qquad \widetilde{Z}_i = Z_i[K_{\max} - K(\hat{r}_i) + 1 : K_{\max}].$$

The downstream model receives the concatenation of the exported chunk memories, whose total length is $\sum_i K(\hat{r}_i)$. The prefix and memory stages share all compressor parameters, and routing does not require trial compression at alternative ratios. Crucially, the selected budget determines the exported suffix, not the size of the shared bank processed by the encoder. We therefore quantify the resource reduction as fewer exported memory representations. Appendix B.5 details logical and physical cache positions.

4 Experiments

We evaluate three questions: whether AdaMem improves the quality–representation trade-off against existing methods, whether the MRQA-trained router transfers to full-context QA without adaptation, and which design choices explain its behavior. Controlled fixed-ratio comparisons isolate routing from compressor quality, while oracle analyses connect the sensitivity motivation to achievable allocations.

4.1 Experimental Setup

Datasets. Following SAC (Liu et al., 2026), we use SlimPajama-6B (Soboleva et al., 2023) for continued pretraining and MRQA (Fisch et al., 2019) for supervised finetuning and downstream evaluation. MRQA provides both in-domain (ID) and out-of-domain (OOD) test sets, enabling evaluation of in-distribution performance and cross-domain generalization. We use 320K SlimPajama samples for continued pretraining and 320K MRQA training examples for supervised finetuning. Each context is partitioned into chunks of up to 510 tokens.

We construct a separate distortion-predictor dataset from 30K examples in the MRQA training split, yielding 47,665 context chunks, with no overlap with the compressor-training data. Each chunk is annotated with three continuous ΔNLL targets for $10\times$, $15\times$, and $34\times$ compression relative to the fixed $5\times$ reference.

Implementation Details. We use Llama-3.2-1B as both the compression encoder and the target LLM. The encoder is augmented with trainable LoRA adapters of rank 128 and $\alpha = 256$, while the target LLM remains frozen. Compressor training consists of 20K continued-pretraining steps followed by 20K supervised-finetuning steps. We then freeze the model and train the distortion predictor from random initialization. Unless otherwise specified, AdaMem uses pathwise routing with a local distortion tolerance of $\tau = 5\%$. Key training settings are summarized in Appendix A.

Table 1: MRQA results (dataset-macro ROUGE-1 F1/EM). †: from SAC (Liu et al., 2026). Underlines: best fixed-5× soft baseline. AdaMem ratios are nominal; savings are relative to 5×.

Method	Compression Ratio	ID F1	ID EM	OOD F1	OOD EM
Full-FT†	1×	71.51	56.85	52.51	36.87
LLMLingua-2†	5×	31.83	19.46	27.90	16.20
5× target output budget					
ICAE†	5×	47.53	33.82	31.22	20.51
500xCompressor†	5×	55.26	40.14	38.46	25.40
EPL	5×	59.43	43.18	42.94	28.73
SAC	5×	<u>63.57</u>	<u>46.95</u>	<u>47.90</u>	<u>32.13</u>
OSCAR	5×	56.05	42.09	30.52	18.93
COMI	5×	56.45	41.11	39.24	26.42
15× target output budget					
ICAE†	15×	44.50	31.28	30.81	20.18
500xCompressor†	15×	49.35	34.98	34.04	22.09
Activation Beacon†	15×	40.31	15.39	36.91	13.02
DAST†	15×	49.83	36.14	31.93	20.95
EPL	15×	50.57	35.81	36.97	24.44
SAC	15×	55.51	39.96	39.54	25.94
OSCAR	15×	51.85	37.89	29.37	18.67
COMI	15×	47.82	33.77	33.66	22.43
Adaptive compression (**Mem. saving vs. 5×: 73.18% ID / 62.48% OOD**)					
AdaMem (Ours)	**18.65**× (ID) **13.33**× (OOD)	**66.24**	**49.34**	**49.03**	**33.94**

Baselines. We compare AdaMem against both hard and soft context-compression methods. Full-FT denotes an uncompressed Llama-3.2-1B model finetuned on MRQA, and LLMLingua-2 (Pan et al., 2024) serves as the hard-compression baseline. Soft-compression baselines include ICAE (Ge et al., 2023), 500xCompressor (Li et al., 2025), Activation Beacon (Zhang et al., 2025), DAST (Chen et al., 2025), EPL (Zhao et al., 2024), SAC (Liu et al., 2026), OSCAR (Louis et al., 2026), and COMI (Tang et al., 2026). For Full-FT, LLMLingua-2, ICAE, 500xCompressor, Activation Beacon, and DAST, we report the results provided by SAC under its corresponding experimental setting. We independently reproduce EPL and SAC, and additionally evaluate OSCAR and COMI. OSCAR uses the same continued-pretraining and supervised-finetuning data as AdaMem, while COMI uses the same supervised-finetuning data. For reproduced baselines, we match the backbone, context chunking, data split, and evaluation protocol whenever compatible with the original method, while retaining method-specific training and compression procedures.

Evaluation Metrics. On MRQA, following SAC (Liu et al., 2026), we report **ROUGE-1 F1** (Lin, 2004) and **exact match (EM)** against the first reference answer. We adopt this evaluation protocol to ensure direct comparability with the baseline results reported by SAC. On LongBench, we instead use its official normalized multi-answer token F1 scorer.

We measure compression by the number of exported memory representations. Let M_5 and M_{selected} denote the corpus-level memory-token counts under fixed 5× compression and AdaMem routing, respectively. We report memory saving and the corresponding nominal compression ratio

$$S_{5\times} = 1 - \frac{M_{\text{selected}}}{M_5}, \quad r_{\text{nom}} = 5\frac{M_5}{M_{\text{selected}}}.$$

All token counts are aggregated over the evaluation set before computing these metrics.

4.2 Main Results

Table 1 summarizes the aggregate results on the in-domain (ID) and out-of-domain (OOD) MRQA test sets. Detailed per-dataset results are provided in Appendix C.

Table 2: Full-context LongBench QA with the same compressor. N: examples; F1: official multi-answer token F1; ratios: nominal. Bold marks the higher F1.

Dataset	N	Fixed-5× F1	AdaMem F1	Nominal ratio
2WikiMultiHopQA	200	22.01	**24.05**	27.802×
MultiFieldQA-en	150	19.52	**20.54**	24.895×
MuSiQue	200	**13.15**	10.42	31.251×
NarrativeQA	200	11.50	**12.38**	33.067×
Qasper	200	12.75	**15.82**	21.400×

In-domain performance. AdaMem achieves 66.24/49.34 ROUGE-1 F1/EM at $18.65\times$, exporting 73.18% fewer memories than fixed $5\times$. It exceeds SAC, the strongest reproduced fixed-$5\times$ soft-compression baseline, by 2.67 F1 and 2.39 EM points. It also outperforms SAC at $15\times$ and the reproduced OSCAR and COMI baselines at both reported budgets. Thus, its smaller output budget is accompanied by higher QA scores than these baseline operating points.

Out-of-domain generalization. AdaMem achieves 49.03/33.94 F1/EM at $13.33\times$ on OOD data, improving on fixed-$5\times$ SAC by 1.13/1.81 points with 62.48% fewer memories. The same $\tau = 5\%$ yields a more conservative ratio than on ID data ($13.33\times$ vs. $18.65\times$), consistent with routing according to distribution-dependent predicted distortion rather than enforcing a common output budget.

4.3 Transfer to Full-Context Question Answering

Evaluation protocol. We evaluate five English QA datasets from LongBench (Bai et al., 2024) that are not included in MRQA: 2WikiMultiHopQA, MultiFieldQA-en, MuSiQue, NarrativeQA, and Qasper. Their complete official test splits contain 950 examples. We chunk every original context without truncation, covering 916–65,274 Llama tokenizer tokens. The Llama-3.2-1B compressor, predictor, Pathwise policy, $\tau = 5\%$, and candidate ratios are unchanged from MRQA. No LongBench data are used for training, threshold selection, or tuning. Fixed-$5\times$ uses the same checkpoint, prompts, and generation settings; only routing differs.

Results. AdaMem transfers to substantially longer contexts while exporting fewer memories (Table 2). Across the five datasets, nominal ratios range from $21.400\times$ to $33.067\times$, corresponding to 76.64%–84.88% fewer exported representations than fixed $5\times$. F1 improves on four datasets: by 2.04 points on 2WikiMultiHopQA, 1.02 on MultiFieldQA-en, 0.88 on NarrativeQA, and 3.07 on Qasper. MuSiQue is the exception, declining from 13.15 to 10.42 F1 at $31.251\times$. The results thus establish a strong transfer operating point for four of the five tasks using the unchanged predictor and tolerance, with dataset-dependent quality outcomes.

4.4 Ablation Studies

Attention Topology. The 2×2 ablation in Table 10 shows that bidirectional prefix encoding provides the dominant gain: with causal memory, ratio-averaged ROUGE-1 F1/EM improves from $61.01/45.14$ to $65.97/50.33$. Adding bidirectional memory further improves F1 by 0.38, 0.43, and 0.72 points at $10\times$, $15\times$, and $34\times$. We therefore use bidirectional attention in both stages, which achieves the best ratio-averaged scores and strongest results at $10\times$–$34\times$. Appendix D.1 reports additional analyses.

Split vs. Joint Execution. With the adaptive router disabled, we compare Split and Joint execution using the same checkpoint and configurations. Maximum absolute differences are 0.095 percentage points in ROUGE-1 F1 and 0.052 in EM, and all paired-bootstrap 95% confidence intervals exceed the -0.5-point non-inferiority margin. Mean representation cosine similarity exceeds 0.9997. These results support making the routing decision before memory construction without materially altering the corresponding joint computation. The complete comparison and analyses appear in Appendix D.2.

Table 3: Fixed and adaptive routing with the same compressor. ROUGE-1 F1/EM are dataset-macro averages; adaptive nominal ratios are ID/OOD.

Routing Strategy	ID F1	ID EM	OOD F1	OOD EM	Nom. Ratio
Fixed 5×	67.53	51.09	50.14	34.45	5×
Fixed 10×	66.91	50.35	49.11	33.92	10×
Fixed 15×	66.31	49.64	48.69	34.06	15×
Fixed 34×	64.36	47.57	47.79	33.24	34×
Adaptive ($\tau = 5\%$)	**66.24**	**49.34**	**49.03**	**33.94**	**18.65× / 13.33×**

Table 4: Pathwise tolerance sweep (dataset-macro ROUGE-1 F1/EM). Ratios: ID/OOD nominal. Bold marks reference scores and the default $\tau = 5\%$ setting.

Setting	ID F1	ID EM	OOD F1	OOD EM	Nom. Ratio
Fixed 5×	**67.53**	**51.09**	**50.14**	**34.45**	5.00×
$\tau = 1\%$	67.23	50.77	49.96	34.38	8.54× / 5.92×
$\tau = 2\%$	67.10	50.54	49.67	34.04	11.62× / 7.33×
$\tau = 5\%$	**66.24**	**49.34**	**49.03**	**33.94**	**18.65× / 13.33×**
$\tau = 10\%$	65.50	48.47	48.04	33.53	24.89× / 25.22×

Fixed-Ratio vs. Adaptive Routing. Using the same shared compressor isolates the routing contribution from the baseline gains in Table 1. On ID data, AdaMem closely matches fixed $15\times$ in ROUGE-1 F1/EM ($66.24/49.34$ vs. $66.31/49.64$), while increasing the nominal ratio to $18.65\times$. On OOD data, it similarly matches fixed $10\times$ ($49.03/33.94$ vs. $49.11/33.92$) at $13.33\times$. Relative to those fixed-ratio budgets, AdaMem exports approximately 19.6% and 25.0% fewer memories, respectively. These paired operating points show the benefit of allocating different capacities to different chunks rather than selecting one uniform ratio.

Effect of Local Distortion Tolerance. The tolerance controls the quality–representation trade-off without changing the trained compressor or predictor (Table 4). Raising τ from 1% to 10% increases nominal compression from $8.54\times$ to $24.89\times$ on ID data and from $5.92\times$ to $25.22\times$ on OOD data, with gradual score reductions. The default $\tau = 5\%$ exports 73.18% and 62.48% fewer memories than the same compressor at fixed $5\times$, with observed F1 decreases of 1.29 and 1.11 points. The contrast with the baseline comparison is important: AdaMem improves on other methods, while routing within its own compressor trades a small amount of QA quality for a smaller output budget.

Pathwise routing is most consequential at strict tolerances. At $\tau = 1\%$, Direct routing changes 15.32% of ID routes and 10.27% of OOD routes by accepting rates beyond an earlier threshold violation. At $\tau = 5\%$, the two policies yield nearly identical task scores (Appendix D.3). Thus, the default results primarily reflect the learned distortion signal, while the Pathwise check specifies how non-monotonic predictions are handled.

4.5 Scalability Across Backbone Sizes

Table 5 evaluates Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B. AdaMem exceeds the corresponding fixed-$15\times$ SAC baseline in F1/EM at all three scales. With the 8B backbone, it reaches $74.86/59.31$ F1/EM at $22.64\times$ on ID data and $59.36/42.77$ at $18.96\times$ on OOD data. Both 3B and 8B retain these gains at nominal ratios above $15\times$ in both domains. At every scale, the same $\tau = 5\%$ induces more conservative routing on OOD than ID data. Appendix E provides per-dataset results and routing statistics.

4.6 From Chunk Sensitivity to Routing Quality

Figure 1 measures ratio-dependent sensitivity; oracle routing tests how that signal translates into an allocation. We use the same disjoint 5,000-example MRQA holdout set and compare the deployed predictor with two reference-answer oracles. The Chunk Distortion Oracle replaces predictions with measured single-chunk distortions while retaining Pathwise routing and $\tau = 5\%$. The Exact

Table 5: MRQA scaling results (dataset-macro ROUGE-1 F1/EM). SAC: fixed $15\times$; AdaMem: Pathwise routing at $\tau = 5\%$, with nominal ratios.

Backbone	Method	ID F1	ID EM	ID Nom. Ratio	OOD F1	OOD EM	OOD Nom. Ratio
Llama-3.2-1B	SAC	55.51	39.96	$15\times$	39.54	25.94	$15\times$
	AdaMem	**66.24**	**49.34**	$18.65\times$	**49.03**	**33.94**	$13.33\times$
Llama-3.2-3B	SAC	65.12	49.01	$15\times$	50.48	34.73	$15\times$
	AdaMem	**73.00**	**56.82**	$19.83\times$	**56.42**	**39.21**	$16.24\times$
Llama-3.1-8B	SAC	67.09	51.08	$15\times$	52.31	35.93	$15\times$
	AdaMem	**74.86**	**59.31**	$22.64\times$	**59.36**	**42.77**	$18.96\times$

Table 6: Oracle analysis on the disjoint 5K MRQA holdout set (example-averaged ROUGE-1 F1/EM). Oracles use reference answers; ratios are nominal; savings are relative to fixed $5\times$.

Method	F1	EM	Nom. Ratio	Mem. Saving
Fixed $5\times$	70.78	55.66	$5.00\times$	0.00%
Predicted Router	69.70	53.78	$17.75\times$	71.82%
Chunk Distortion Oracle	73.91	59.74	$13.38\times$	62.62%
Exact Task Oracle	**74.92**	**59.74**	**26.79**$\times$	**81.34%**

Task Oracle enumerates complete routes and minimizes exported memories subject to matching or exceeding each example's fixed-$5\times$ ROUGE-1 F1. Neither oracle is used for deployment or LongBench adaptation; Appendix F gives the construction and subset results.

Measured chunk distortions support effective nonuniform allocation (Table 6). The Chunk Distortion Oracle improves F1 from 70.78 under fixed $5\times$ to 73.91 while exporting 62.62% fewer memories. Relative to the predicted router, it gains 4.21 F1 points but selects a more conservative aggregate ratio ($13.38\times$ vs. $17.75\times$). This separates the usefulness of the distortion signal from the accuracy with which the lightweight predictor estimates it.

Allocation quality also depends on which chunks receive aggressive compression, not just how often it is selected. The predicted router assigns $34\times$ to 78.50% of chunks, compared with 59.88% for the Chunk Distortion Oracle. Yet the Exact Task Oracle selects $34\times$ for 90.30% of chunks and attains the highest F1 at $26.79\times$ (Appendix F). These results connect the heterogeneity in Figure 1 to concrete routing headroom: better prediction improves local decisions, while joint task-level selection can exploit allocations beyond independent chunk constraints.

5 Conclusion

AdaMem makes chunk-specific compression sensitivity the basis of memory allocation. A shared multi-ratio compressor, a lightweight distortion predictor, and a decision-before-memory pathway jointly select per-chunk budgets under a local predicted-distortion threshold. On MRQA, this design reduces exported memory representations while maintaining competitive QA quality across domains and 1B–8B backbones. Controlled ablations establish the contribution of adaptive allocation and support the split execution design. The resulting lesson is that the appropriate compression ratio depends on the chunk's response to compression, not only on a prescribed global budget.

Our evaluation focuses on question answering. Extending distortion-guided allocation to other generation tasks and learning predictors of complete-route quality are natural next steps, informed by the oracle analysis of allocation headroom.

6 AI Use Statement

In this work, we used generative AI tools to assist with the design and refinement of ablation experiments and with translation during manuscript preparation. We did not use generative AI tools for other required disclosure tasks, including generating synthetic datasets, developing theoretical or conceptual frameworks, formulating or proving mathematical claims, proposing or refining

hypotheses, implementing the core research methodology, dataset cleaning or reformatting, qualitative or thematic data analysis, or result interpretation; tasks not relevant to this work are not applicable. We additionally used generative AI tools to assist with editing software code, drafting parts of the manuscript, improving writing clarity and readability, and modifying scientific figures. All AI-assisted outputs were reviewed and revised by the authors before inclusion. We take full responsibility for the final content of this work, including all text, claims, code, experimental designs, translations, and artifacts produced with the aid of generative AI.

7 Reproducibility Statement

Our experiments use public Llama backbones and the SlimPajama and MRQA datasets. The predictor training data is disjoint from compressor training data. Appendices B.1–B.5 describe the compressor, distortion predictor, and execution procedure; Appendix A reports compressor and predictor settings and the hardware. The code will be made publicly available upon acceptance.

REFERENCES

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Zhiwei Cao, Qian Cao, Yu Lu, Ningxin Peng, Luyang Huang, Shanbo Cheng, and Jinsong Su. Retaining key information under high compression ratios: Query-guided compressor for llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12685–12695, 2024.

Shaoshen Chen, Yangning Li, Zishan Xu, Yongqin Zeng, Shunlong Wu, Xinshuo Hu, Zifei Shan, Xin Su, Jiwei Tang, Yinghui Li, et al. Dast: Context-aware compression in llms via dynamic allocation of soft tokens. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20544–20552, 2025.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516, 2024.

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.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 38082–38106, 2025.

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. Learning to compress prompt in natural language formats. 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. 7756–7767, 2024.

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28715–28732, 2025.

Weizhi Fei, Xueyan Niu, XIE GUOQING, Yingqing Liu, Bo Bai, and Wei Han. Efficient prompt compression with evaluator heads for long-context transformer inference. *Advances in Neural Information Processing Systems*, 38:154566–154593, 2026.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd workshop on machine reading for question answering*, pp. 1–13, 2019.

Jun Gao, Qi Lv, Zili Wang, Tianxiang Wu, Ziqiang Cao, and Wenjie Li. UniICL: An efficient ICL framework unifying compression, selection, and generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 500–510, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.24. URL https://aclanthology.org/2025.acl-long.24/.

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. *arXiv preprint arXiv:2307.06945*, 2023.

Jinwu Hu, Wei Zhang, Yufeng Wang, Yu Hu, Bin Xiao, Mingkui Tan, and Qing Du. Dynamic prompt compression for efficient inference of large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

Taeho Hwang, Sukmin Cho, Soyeong Jeong, Hoyun Song, SeungYoon Han, and Jong C Park. Exit: Context-aware extractive compression for enhancing retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4895–4924, 2025.

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.

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.

Hoyoun Jung and Kyung-Joong Kim. Discrete prompt compression with reinforcement learning. *IEEE Access*, 12:72578–72587, 2024.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 6342–6353, 2023.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xcompressor: Generalized prompt compression for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25081–25091, 2025.

Zihan Liao, Jun Wang, Hang Yu, Lingxiao Wei, Jianguo Li, and Wei Zhang. E2llm: Encoder elongated large language models for long-context understanding and reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 19212–19241, 2025.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Barys Liskavets, Shuvendu Roy, Maxim Ushakov, Mark Klibanov, Ali Etemad, and Shane Luke. Task-agnostic prompt compression with context-aware sentence embedding and reward-guided task descriptor. *arXiv preprint arXiv:2502.13374*, 2025a.

Barys Liskavets, Maxim Ushakov, Shuvendu Roy, Mark Klibanov, Ali Etemad, and Shane K Luke. Prompt compression with context-aware sentence encoding for fast and improved llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24595–24604, 2025b.

Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, Jingbo Zhu, et al. Autoencoding-free context compression for llms via contextual semantic anchors. In *International Conference on Learning Representations*, volume 2026, pp. 138643–138665, 2026.

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*, pp. 15506–15521, 2025.

Maxime Louis, Thibault Formal, Hervé Déjean, and Stéphane Clinchant. Oscar: Online soft compression for rag. In *International Conference on Learning Representations*, volume 2026, pp. 3102–3121, 2026.

Gonzalo Ariel Meyoyan and Luciano Del Corro. A bertology view of llm orchestrations: Token- and layer-selective probes for efficient single-pass classification. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42226–42239, 2026.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Shivam Ratnakar, Abhiroop Talasila, and Vinayak K Doifode. Latentgate: Low-latency semantic routing via frozen-backbone probing of small language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 2284–2294, 2026.

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. Context embeddings for efficient answer generation in rag. *arXiv preprint arXiv:2407.09252*, 2024.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. `https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama`, 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

Sijun Tan, Xiuyu Li, Shishir G Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E Gonzalez, and Raluca Ada Popa. Lloco: Learning long contexts offline. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17605–17621, 2024.

Jiwei Tang, Jin Xu, Tingwei Lu, Zhicheng Zhang, Yiming Zhao, Lin Hai, and Hai-Tao Zheng. Perception compressor: A training-free prompt compression framework in long context scenarios. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4093–4108, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.229. URL `https://aclanthology.org/2025.findings-naacl.229/`.

Jiwei Tang, Shilei Liu, Zhicheng Zhang, Yujin Yuan, Libin Zheng, Bo Zheng, et al. Comi: Coarse-to-fine context compression via marginal information gain. In *International Conference on Learning Representations*, volume 2026, pp. 46648–46666, 2026.

Xiangfeng Wang, Zaiyi Chen, Tong Xu, Zheyong Xie, Yongyi He, and Enhong Chen. In-context former: Lightning-fast compressing context for large language model. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2445–2460, 2024.

Zijian Wang and Chang Xu. Thoughtprobe: Classifier-guided llm thought space exploration via probing representations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6029–6050, 2025.

Menghua Wu, Cai Zhou, Stephen Bates, and Tommi Jaakkola. Thought calibration: Efficient and confident test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14302–14316, 2025.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Zhentao Xu, Fengyi Li, Albert C Chen, and Xiaofeng Wang. Procut: Llm prompt compression via attribution estimation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 285–309, 2025.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. Compact: Compressing retrieved documents actively for question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21424–21439, 2024.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *International Conference on Learning Representations*, volume 2025, pp. 101932–101948, 2025.

Runsong Zhao, Xin Liu, Xinyu Liu, Pengcheng Huang, Chunyang Xiao, Tong Xiao, and Jingbo Zhu. Position ids matter: An enhanced position layout for efficient context compression in large language models. *arXiv preprint arXiv:2409.14364*, 2024.

Yi Zhao, Zuchao Li, Hai Zhao, Baoyuan Qi, and Liu Guoming. DAC: A dynamic attention-aware approach for task-agnostic prompt compression. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19395–19407, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.952. URL https://aclanthology.org/2025.acl-long.952/.

A Training Configuration

Table 7: Key settings for compressor and distortion-predictor training. Shared compressor settings span both stage columns.

Compressor	**Pretraining**	**Finetuning**
Learning rate	1×10^{-4}	5×10^{-5}
Optimizer steps	20K per stage	
Global batch size	16	
Adam betas	$(0.9, 0.95)$	
Weight decay	0.1	
Warmup / LR schedule	300 steps / constant after warmup	
Max gradient norm	2.0	
LoRA rank / α	128 / 256	
Distortion predictor	**Setting**	
Projection dimension	128	
Ratio queries	3, for $r \in \{10, 15, 34\}$	
Auxiliary-loss weight	0.2	
Auxiliary-loss tolerances	$\tau \in \{1\%, 2\%, 5\%\}$	

Compressor training updates only the LoRA adapters on the query and value projections, memory-token embeddings, and two special-token embeddings; the pretrained parameters of the compression encoder and target LLM remain frozen. Both continued pretraining and supervised finetuning use AdamW, with the latter initialized from the former. Table 7 summarizes the key settings for the compressor and distortion predictor.

After compressor training, we freeze the compressor and train the distortion predictor from random initialization with AdamW. The predictor uses the architecture and normalized regression objective in Appendix B.4; the table lists its projection dimension, ratio queries, and auxiliary-loss settings.

All compressors are trained in bfloat16 precision. The 1B and 3B variants use four NVIDIA RTX 3090 GPUs (24GB), while the 8B variant uses four NVIDIA A100 GPUs (40GB).

B Additional Method Details

B.1 Ratio-Specific Position Assignment

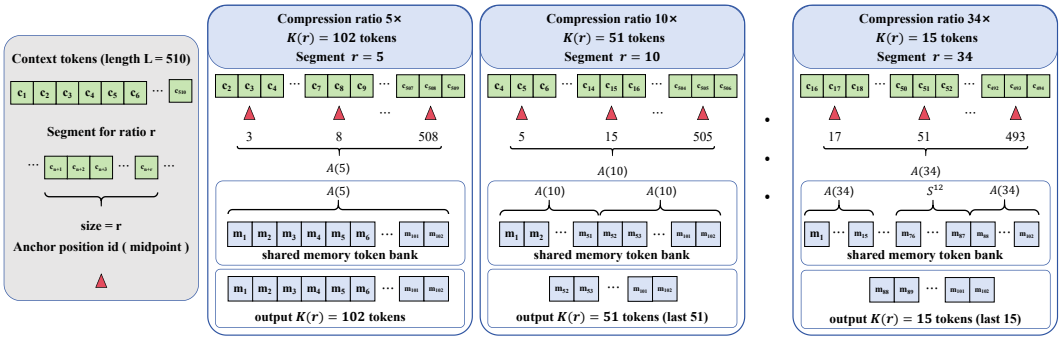

Figure 3: Ratio-specific logical positions in the shared memory bank. All 102 tokens participate in compression; only the final $K(r)$ representations, assigned a complete anchor sequence, are exported.

Following Section 3.2, we construct a ratio-specific position layout for the shared $K_{\max} = 102$ memory tokens (Figure 3). With fixed chunk size $L = 510$ and ratio $r \in \mathcal{R}$, let $k = K(r) = L/r$. Indexing context positions from 1 to L, we define the anchor of each length-r segment as its midpoint:

$$a_j^{(r)} = \left\lfloor \frac{(j-1)r + 1 + jr}{2} \right\rfloor, \qquad j = 1, \ldots, k.$$

We denote the ordered anchor sequence by $A(r) = (a_1^{(r)}, \ldots, a_k^{(r)})$. In Figure 3, $n = (j-1)r$ denotes a segment starting offset, so the segment spans $c_{n+1}, \ldots, c_{n+r}$; N denotes the number of context chunks. The final k memory tokens receive $A(r)$, and their output representations are exported to the downstream LLM. For the preceding $B = K_{\max} - k$ memory tokens, let

$$q = \left\lfloor \frac{B}{k} \right\rfloor, \qquad s = B - qk.$$

We assign q complete copies of the anchor sequence in order to the first qk memory tokens. If $s > 0$, the remaining tokens $m_{qk+1}, \ldots, m_B$ receive anchors $a_{u_1}^{(r)}, \ldots, a_{u_s}^{(r)}$, respectively, with indices

$$u_i = 1 + \left\lfloor \frac{(2i-1)k}{2s} \right\rfloor, \qquad i = 1, \ldots, s.$$

This subsampled sequence is omitted when $s = 0$. Together, these assignments define the ordered logical position layout $P(r) = (p_1^{(r)}, \ldots, p_{K_{\max}}^{(r)})$.

For example, at $r = 34$, we have $k = 15$, $B = 87$, $q = 5$, and $s = 12$: the first 75 memory tokens receive five complete anchor sequences, tokens $m_{76}, \ldots, m_{87}$ receive the subsampled sequence $S^{12} = (a_{u_1}^{(34)}, \ldots, a_{u_{12}}^{(34)})$ shown in Figure 3, and the final 15 receive the complete sequence $A(34)$ and are exported. Thus, the construction also covers ratios for which k does not divide $K_{\max}$.

B.2 Prefix and Memory Attention Topology

Memory tokens serve as latent compression slots rather than autoregressively generated language tokens. AdaMem therefore allows bidirectional interaction within both the context–query prefix and the memory-token block.

Let $P = [C_i; Q]$ denote the context–query prefix and M denote the memory-token block. The conceptual attention topology is

$$A = \begin{bmatrix} \mathbf{1}_{P \times P} & \mathbf{0}_{P \times M} \\ \mathbf{1}_{M \times P} & \mathbf{1}_{M \times M} \end{bmatrix},$$

where rows correspond to queries, columns to keys, and ones indicate permitted attention.

Bidirectional prefix attention makes the context-token states query-conditioned, although the predictor reads only context-token positions. Memory tokens attend to the entire prefix and to one another, while prefix tokens do not attend to memory tokens. This dependency allows the prefix to be computed and cached independently of the memory-position layout, enabling the split execution in Appendix B.5.

B.3 Multi-Ratio Curriculum Training

During both continued pretraining and supervised finetuning, we use a four-stage curriculum, with each stage lasting 5,000 optimizer steps. The active ratio set is progressively expanded as

$$\{5\} \to \{5, 10\} \to \{5, 10, 15\} \to \{5, 10, 15, 34\}.$$

At each optimizer step, one compression ratio is sampled uniformly from the currently active set. The same sampled ratio is used for all micro-batches within the corresponding gradient-accumulation cycle and is shared across all GPUs. The selected ratio determines the ratio-specific position layout and export budget for that optimizer step.

B.4 Distortion Predictor Architecture and Training

Architecture. The predictor projects the query-conditioned, final-layer context-token states $H_{i,C}^p$ to 128 dimensions. Three learnable ratio queries, corresponding to $r \in \{10, 15, 34\}$, attend to these projected states:

$$\begin{aligned} U_i &= \mathrm{Linear}\Big(\mathrm{LN}(H_{i,C}^p)\Big), \\ A_i &= \mathrm{MHA}(Q_{\mathrm{ratios}}, U_i, U_i; \mathrm{mask}_i), \\ \widehat{\widetilde{d}}_{i,r} &= \mathrm{Linear}(\mathrm{LN}(A_{i,r})). \end{aligned}$$

Here, $Q_{\text{ratios}} \in \mathbb{R}^{3\times 128}$. The same architecture is used across backbone sizes, with only the input projection dimension adjusted accordingly.

Training Objective. Distortion targets are standardized independently for each ratio using the training-set mean μ_r and standard deviation σ_r:

$$\tilde{d}_{i,r} = \frac{d_{i,r} - \mu_r}{\sigma_r}.$$

The predictor is optimized with

$$\mathcal{L}_{\text{pred}} = \text{SmoothL1}(\widehat{\tilde{\mathbf{d}}}, \tilde{\mathbf{d}}) + 0.2\,\mathcal{L}_{\text{acc}},$$

where $\mathcal{L}_{\text{acc}}$ is an auxiliary binary cross-entropy loss that supervises pathwise acceptability under $\tau \in \{1\%, 2\%, 5\%\}$. For ratio r, a sample is considered pathwise acceptable only if all candidate ratios up to r satisfy the corresponding NLL threshold. The auxiliary objective is derived directly from the regression outputs and introduces no separate classification head.

Training and Inference. The compressor is frozen while the predictor is trained with AdamW. At inference, the predicted standardized distortions are transformed back to the original ΔNLL scale as

$$\hat{d}_{i,r} = \sigma_r \hat{\tilde{d}}_{i,r} + \mu_r,$$

and passed to the routing policy.

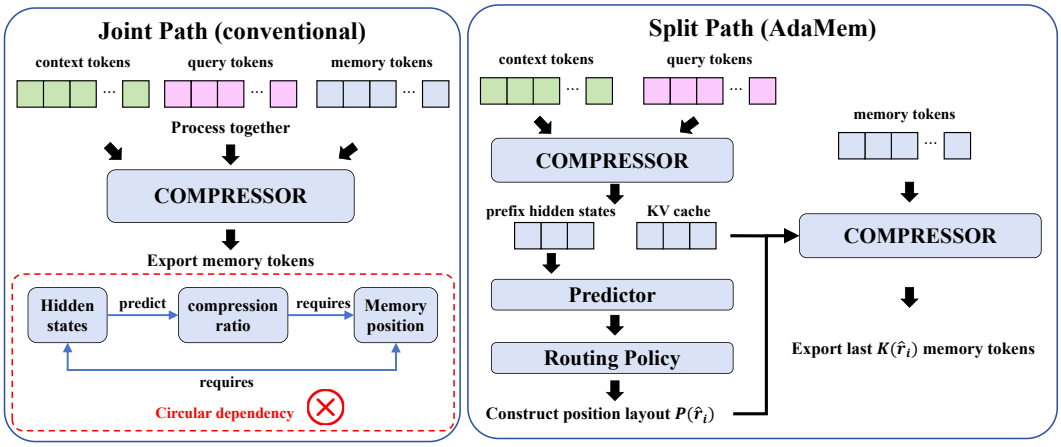

Figure 4: Joint and Split execution of the same compressor. Split encodes and caches the context–query prefix before routing, then reuses its KV cache to construct memory with the selected position layout.

B.5 Decision-Before-Memory Split Execution

Figure 4 illustrates the execution schedule. The attention topology in Appendix B.2 makes prefix states independent of memory tokens, allowing the same compressor to encode the prefix before constructing ratio-specific memory.

Prefix stage. The context–query prefix is first processed without memory tokens:

$$(H_i^p, KV_i^p) = E_\phi^{\text{prefix}}([C_i; Q]),$$

where H_i^p denotes the prefix hidden states and KV_i^p denotes the cached prefix keys and values.

The context-token portion of H_i^p is then used by the distortion predictor G_ψ to obtain

$$\hat{d}_i = G_\psi(H_{i,C}^p).$$

Memory stage. Reuse KV_i^p to encode all $K_{\max}$ memory tokens with layout $P(\hat{r}_i)$, and export only the final $K(\hat{r}_i)$ representations.

$$Z_i = E_\phi^{\text{memory}}(M; KV_i^p, P(\hat{r}_i)), \qquad \widetilde{Z}_i = Z_i[K_{\max} - K(\hat{r}_i) + 1 : K_{\max}].$$

Both stages share all compressor parameters. The ratio-specific layout specifies logical RoPE position IDs, which may repeat or decrease across anchor sequences; physical cache positions remain strictly increasing as memory KV states are appended. Logical positions therefore control positional alignment without changing the sequential storage order of the KV cache. Appendix D.2 evaluates Split against Joint execution using the same checkpoint.

C Full MRQA Results

This section provides the complete per-dataset results corresponding to the aggregate MRQA results reported in Table 1. We report ROUGE-1 F1 and exact match (EM) for all in-domain and out-of-domain subsets. The Average columns give equal weight to the six datasets in each split. The per-dataset breakdown shows how the aggregate quality–compression trade-off varies across datasets.

C.1 In-Domain Results

Table 8 reports the complete results on the six in-domain MRQA datasets: SQuAD, NewsQA, TriviaQA, SearchQA, HotpotQA, and Natural Questions.

AdaMem achieves average F1/EM of 66.24/49.34 at a nominal compression ratio of $18.65\times$, exceeding SAC at $5\times$ by 2.67/2.39 points while exporting 73.18% fewer memory tokens than fixed $5\times$ compression. Its F1 exceeds SAC at $5\times$ on five of the six datasets, with the largest gains on SQuAD (+7.28) and NewsQA (+6.63). The improvements are not uniform across metrics: TriviaQA F1/EM decreases by 0.99/0.82 points, and EM is slightly lower on SearchQA and HotpotQA. Against the reported $15\times$ baselines, AdaMem achieves higher F1 and EM on every dataset. Thus, the average improvement reflects gains across multiple datasets, with remaining dataset-specific trade-offs.

Table 8: Per-dataset in-domain MRQA results (ROUGE-1 F1/EM). Average is the mean over six datasets. †: results reported by SAC (Liu et al., 2026). AdaMem's ratio is nominal; memory savings are relative to fixed $5\times$.

	SQuAD		NewsQA		TriviaQA		SearchQA		HotpotQA		NQ		Average	
Methods	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Full-FT†	77.69	59.71	63.50	46.04	68.80	60.54	73.25	62.07	74.78	59.26	71.01	53.47	71.51	56.85
LLMLingua-2†	32.93	19.57	26.78	13.20	9.67	8.12	45.40	31.80	36.10	22.05	40.08	22.01	31.83	19.46
$5\times$ target output budget														
ICAE†	36.20	22.12	28.06	13.77	54.63	45.59	65.12	53.06	48.79	33.40	52.36	34.99	47.53	33.82
500xCompressor†	51.62	33.63	39.70	22.63	57.62	48.76	66.43	54.38	59.10	42.20	57.11	39.26	55.26	40.14
EPL	59.36	39.47	42.83	23.93	60.58	51.29	68.85	56.41	64.47	46.06	60.47	41.89	59.43	43.18
SAC	65.43	44.66	47.82	26.35	65.60	56.43	71.72	59.72	66.78	49.62	64.05	44.94	63.57	46.95
OSCAR	53.37	35.95	46.34	30.58	57.44	49.99	65.33	53.89	59.47	44.15	54.33	37.98	56.05	42.09
COMI	54.06	35.68	38.69	20.96	57.32	48.86	68.35	56.53	62.58	45.52	57.69	39.13	56.45	41.11
$15\times$ target output budget														
ICAE†	31.90	18.91	25.25	11.97	51.78	42.94	64.81	52.89	45.22	30.32	48.01	30.67	44.50	31.28
500xCompressor†	40.68	24.97	32.01	16.76	53.84	44.86	65.65	53.70	53.01	36.30	50.93	33.26	49.35	34.98
Activation Beacon†	37.53	14.31	31.10	7.72	48.85	4.47	39.06	22.67	45.01	27.35	40.29	15.82	40.31	15.39
DAST†	36.33	22.65	31.55	16.29	56.92	48.14	68.07	56.51	54.02	37.93	52.10	35.31	49.83	36.14
EPL	43.01	26.42	29.91	14.52	55.71	46.38	67.55	54.58	54.31	37.65	52.94	35.31	50.57	35.81
SAC	47.61	29.91	36.68	18.40	61.37	52.24	70.29	57.77	59.80	42.57	57.31	38.85	55.51	39.96
OSCAR	45.96	28.96	35.97	21.34	55.19	47.71	64.77	52.89	56.92	40.74	52.28	35.70	51.85	37.89
COMI	38.32	23.33	28.24	13.79	51.79	43.13	65.67	53.58	52.29	35.62	50.61	33.13	47.82	33.77
Adaptive compression ($18.65\times$ nominal ratio; 73.18% memory-token saving vs. $5\times$)														
AdaMem (Ours)	**72.71**	**52.30**	**54.45**	**33.14**	**64.61**	**55.61**	**72.30**	**58.99**	**67.70**	**49.55**	**65.68**	**46.42**	**66.24**	**49.34**

C.2 Out-of-Domain Results

Table 9 reports the complete results on the six out-of-domain MRQA datasets: BioASQ, DROP, DuoRC, RACE, Relation Extraction, and TextbookQA.

On the out-of-domain split, AdaMem achieves average F1/EM of 49.03/33.94 at a nominal ratio of $13.33\times$, improving on SAC at $5\times$ by 1.13/1.81 points while exporting 62.48% fewer memory tokens than fixed $5\times$ compression. Both metrics improve on five of the six datasets. TextbookQA shows the largest gains (+3.87 F1 and +3.80 EM), followed by BioASQ (+2.26 F1 and +3.20 EM). DROP is the exception, with decreases of 1.42 F1 and 0.26 EM. These results show that the aggregate gains extend to multiple unseen datasets, although they do not imply uniformly better performance under domain shift.

Table 9: Per-dataset out-of-domain MRQA results (ROUGE-1 F1/EM). Average is the mean over six datasets. †: results reported by SAC (Liu et al., 2026). AdaMem's ratio is nominal; memory savings are relative to fixed $5\times$.

	BioASQ		DROP		DuoRC		RACE		RE		TQA		Average	
Methods	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Full-FT†	49.37	36.77	44.67	34.46	48.82	35.51	35.57	9.64	83.34	72.46	53.32	32.40	52.51	36.87
LLMLingua-2†	27.76	19.48	27.28	18.83	27.07	18.32	17.54	4.15	39.30	20.59	28.42	15.83	27.90	16.20
$5\times$ target output budget														
ICAE†	36.08	26.06	28.95	21.09	16.67	10.79	15.65	3.12	54.73	41.01	35.24	20.96	31.22	20.51
500xCompressor†	40.30	28.99	35.40	25.55	29.43	19.32	21.57	4.90	65.43	50.88	38.62	22.75	38.46	25.40
EPL	41.70	28.99	39.72	28.01	32.84	22.58	26.14	5.19	75.26	61.70	41.97	25.88	42.94	28.73
SAC	45.23	30.78	42.22	31.40	40.00	27.25	31.05	6.68	78.18	65.06	50.74	31.60	47.90	32.13
OSCAR	24.98	15.96	32.11	23.62	10.68	5.60	19.47	4.01	58.65	44.88	37.23	19.49	30.52	18.93
COMI	36.68	26.06	33.00	23.75	29.93	20.32	21.94	4.15	73.22	60.14	40.68	24.09	39.24	26.42
$15\times$ target output budget														
ICAE†	35.51	24.47	30.39	21.96	13.78	9.06	15.21	3.71	55.24	40.33	34.75	21.56	30.81	20.18
500xCompressor†	36.30	25.93	33.46	23.55	20.53	12.72	18.49	3.41	54.37	41.11	41.09	25.82	34.04	22.09
Activation Beacon†	36.07	7.65	34.39	14.90	33.78	14.99	26.68	3.71	53.28	23.91	37.27	12.97	36.91	13.02
DAST†	36.57	27.06	31.90	22.42	21.56	14.26	16.31	3.26	48.54	35.92	36.69	22.75	31.93	20.95
EPL	40.12	28.46	32.51	23.35	23.88	15.06	20.96	4.60	64.07	49.56	40.26	25.62	36.97	24.44
SAC	41.09	28.59	35.98	25.55	29.47	20.12	23.33	4.75	62.71	48.61	44.63	28.01	39.54	25.94
OSCAR	18.75	11.97	30.10	21.69	24.57	16.72	15.68	3.12	54.01	40.88	33.10	17.63	29.37	18.67
COMI	36.10	26.53	29.24	20.23	21.54	13.52	17.06	4.01	61.68	49.05	36.35	21.22	33.66	22.43
Adaptive compression (13.33× nominal ratio; 62.48% memory-token saving vs. 5×)														
AdaMem (Ours)	**47.49**	**33.98**	**40.80**	**31.14**	**40.94**	**29.31**	**31.24**	**7.27**	**79.11**	**66.52**	**54.61**	**35.40**	**49.03**	**33.94**

D Additional Ablation Analyses

D.1 Additional Analysis of Attention Topology

Table 10: Prefix and memory attention ablation. Entries are ROUGE-1 F1/EM pooled over all 67,854 MRQA examples. Avg. is the mean over four ratios; bold marks the best scores in each column.

Prefix	**Memory**	**5×**	**10×**	**15×**	**34×**	**Avg.**
Causal	Causal	63.62 / 47.67	61.98 / 46.07	60.70 / 44.72	57.75 / 42.08	61.01 / 45.14
Causal	Bidirectional	64.27 / 48.45	62.92 / 46.99	61.99 / 46.04	59.80 / 44.07	62.24 / 46.39
Bidirectional	Causal	**67.38 / 51.78**	66.43 / 50.84	66.01 / 50.38	64.06 / 48.30	65.97 / 50.33
Bidirectional	Bidirectional	67.34 / 51.60	**66.81 / 51.03**	**66.44 / 50.65**	**64.78 / 48.89**	**66.34 / 50.54**

Table 11 reports the F1 main effects and interaction terms of the full 2×2 attention design. Bidirectional prefix encoding provides the dominant improvement, with a ratio-averaged F1 gain of $+4.53$; the corresponding EM gain computed from Table 10 is $+4.67$. Bidirectional memory yields smaller average gains of $+0.80$ F1 and $+0.73$ EM, but its effect increases with the compression ratio, suggesting that memory–memory interaction becomes more useful under a tighter representation

bottleneck. The interaction is negative at all four ratios for both metrics (-0.86 F1 and -1.04 EM on average), indicating diminishing returns between prefix and memory bidirectionality on the additive score scale.

Table 11: Main effects and interaction of prefix and memory bidirectionality, derived from Table 10. Values are ROUGE-1 F1 differences in percentage points. Main effects average over the other factor; Average is the mean over four ratios.

Ratio	Prefix Effect	Memory Effect	Interaction
$5\times$	+3.42	+0.31	-0.69
$10\times$	+4.17	+0.66	-0.56
$15\times$	+4.88	+0.86	-0.86
$34\times$	+5.64	+1.39	-1.33
Average	+4.53	+0.80	-0.86

The fully bidirectional configuration is also more robust to aggressive compression: its F1 decreases by only 2.56 points from $5\times$ to $34\times$, compared with 5.87 points for the fully causal variant.

D.2 Additional Analysis of Split vs. Joint Execution

Table 12: Joint versus Split execution on pooled MRQA examples with the same checkpoint and adaptive routing disabled. Δ is Split $-$ Joint in percentage points; brackets give paired-bootstrap 95% confidence intervals.

Ratio	ROUGE-1 F1			EM		
	Joint	Split	Δ [95% CI]	Joint	Split	Δ [95% CI]
$5\times$	67.349	67.342	-0.007 $[-0.063, 0.049]$	51.651	51.599	-0.052 $[-0.119, 0.013]$
$10\times$	66.812	66.809	-0.003 $[-0.057, 0.052]$	51.039	51.032	-0.007 $[-0.071, 0.058]$
$15\times$	66.419	66.438	$+0.020$ $[-0.035, 0.072]$	50.650	50.647	-0.003 $[-0.068, 0.062]$
$34\times$	64.682	64.777	$+0.095$ $[0.042, 0.151]$	48.842	48.886	$+0.044$ $[-0.019, 0.116]$

Table 12 reports the full task-level comparison. Joint processes prefix and memory tokens in one forward pass, whereas Split first encodes the prefix and reuses its KV cache to construct memory representations. Across rates, maximum absolute differences are 0.095 pp in ROUGE-1 F1 and 0.052 pp in EM; all paired-bootstrap 95% confidence intervals lie above the predefined -0.5-pp non-inferiority margin.

Representation and decoder consistency. Table 13 further compares the two execution paths at the representation and decoder levels. Although finite-precision GPU execution leads to small numerical differences, the compressed representations remain highly aligned across all compression ratios, with average cosine similarity above 0.9997. Decoder predictions are also stable, with token-level top-1 agreement around 98% and KL divergence below 0.0025.

Table 13: Joint–Split consistency, averaged over sampled evaluation examples. Rep. Cosine is memory-representation cosine similarity; Top-1 Agreement and KL measure token-level prediction agreement and KL divergence between decoder output distributions, respectively.

Ratio	Rep. Cosine	Top-1 Agreement	KL
$5\times$	0.999862	97.9301%	0.002177
$10\times$	0.999846	97.7276%	0.002135
$15\times$	0.999781	97.9619%	0.002273
$34\times$	0.999723	97.9258%	0.002447

Domain-level non-inferiority. The non-inferiority conclusion also holds separately on the ID and OOD subsets: for both ROUGE-1 F1 and EM, all four compression ratios satisfy the predefined 0.5-pp non-inferiority criterion in the overall, ID, and OOD evaluations. Confidence intervals are

estimated from 2,000 paired-bootstrap resamples over strictly paired QA examples. This analysis is based on a single fixed trained checkpoint and therefore establishes inference-path non-inferiority rather than equivalence between independently trained Joint and Split models.

D.3 Direct vs. Pathwise Routing

Table 14: Pathwise versus Direct routing using the same predicted distortions and threshold $T_\tau = \log(1+\tau)$. ROUGE-1 F1/EM are dataset-macro averages. Nominal ratios and route-change rates are reported as ID/OOD; route change is the fraction of chunks assigned different ratios. Bold marks the default setting.

τ	Routing	ID F1	ID EM	OOD F1	OOD EM	Nom. Ratio	Route Change
1%	Pathwise	67.23	50.77	49.96	34.38	$8.54\times$ /$5.92\times$	15.32%/10.27%
	Direct	67.22	50.72	50.01	34.36	$10.01\times$ /$6.39\times$	
2%	Pathwise	67.10	50.54	49.67	34.04	$11.62\times$ /$7.33\times$	6.25%/3.81%
	Direct	67.08	50.48	49.74	34.13	$12.53\times$ /$7.56\times$	
5%	**Pathwise**	**66.24**	**49.34**	**49.03**	**33.94**	$18.65\times$ /$13.33\times$	0.16%/0.21%
	Direct	66.23	49.32	49.02	33.94	$18.69\times$ /$13.37\times$	
10%	Pathwise	65.50	48.47	48.04	33.53	$24.89\times$ /$25.22\times$	0.01%/0.00%
	Direct	65.50	48.47	48.04	33.53	$24.89\times$ /$25.22\times$	

Table 14 provides the full comparison between Direct and Pathwise routing. Both strategies operate on the same predicted distortion vectors and local distortion thresholds; Direct evaluates each candidate ratio independently, whereas Pathwise requires all preceding ratios to satisfy the constraint before selecting a more aggressive operating point.

The difference is most pronounced under tight local distortion tolerances. At $\tau = 1\%$, Direct changes 15.32% of ID routes and 10.27% of OOD routes relative to Pathwise, increasing the nominal compression ratio from $8.54\times$ to $10.01\times$ on ID data and from $5.92\times$ to $6.39\times$ on OOD data. The corresponding differences in macro-averaged F1 and EM remain small and do not consistently favor either strategy. As τ increases, the two routing rules rapidly converge. At the default $\tau = 5\%$, only 0.16% of ID and 0.21% of OOD routes differ, with nearly identical quality and nominal compression ratios. We therefore use Pathwise routing as the default conservative rule: it explicitly prevents skipping an intermediate compression level that violates the local predicted-distortion threshold, while incurring negligible loss in exported-representation reduction at the operating point used in the main experiments.

E Additional Scalability Results

Per-dataset results. Tables 15 and 16 report the subset-level results of AdaMem across the three backbone sizes. The gains are observed across most subsets, suggesting that the scaling trend reported in the main text is not concentrated in only a few datasets.

Table 15: Per-dataset in-domain MRQA results for AdaMem across backbone sizes at $\tau = 5\%$. Scores are ROUGE-1 F1/EM; Average is the mean over six datasets.

	SQuAD		NewsQA		TriviaQA		SearchQA		HotpotQA		NQ		Average	
Backbone	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Llama-3.2-1B	72.71	52.30	54.45	33.14	64.61	55.61	72.30	58.99	67.70	49.55	65.68	46.42	66.24	49.34
Llama-3.2-3B	80.28	61.08	62.51	42.09	71.76	62.80	76.99	64.00	74.33	57.30	72.13	53.66	73.00	56.82
Llama-3.1-8B	82.04	63.48	66.34	46.87	74.15	65.34	76.56	64.91	76.63	59.97	73.46	55.30	74.86	59.31

Routing distribution. Table 17 reports the chunk-level routing distribution under the same local distortion tolerance of $\tau = 5\%$. The percentages are computed over all context chunks in each split (*chunk-micro*), rather than by first computing a routing distribution for each QA example and then

Table 16: Per-dataset out-of-domain MRQA results for AdaMem across backbone sizes at $\tau = 5\%$. Scores are ROUGE-1 F1/EM; Average is the mean over six datasets.

	BioASQ		DROP		DuoRC		RACE		RE		TQA		Average	
Backbone	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Llama-3.2-1B	47.49	33.98	40.80	31.14	40.94	29.31	31.24	7.27	79.11	66.52	54.61	35.40	49.03	33.94
Llama-3.2-3B	54.23	40.03	55.81	43.91	48.16	34.84	40.06	9.94	84.61	74.02	55.64	32.53	56.42	39.21
Llama-3.1-8B	57.01	43.42	62.96	52.03	50.15	36.38	41.15	10.53	86.76	78.63	58.12	35.66	59.36	42.77

averaging across examples. We additionally report the corresponding nominal compression ratio and memory-token saving relative to fixed $5\times$ compression.

Table 17: Chunk-level routing across backbone sizes under Pathwise routing at $\tau = 5\%$. Percentages pool all chunks in each ID/OOD split. Ratio is nominal; Saving is the reduction in exported memories relative to fixed $5\times$.

Backbone	Split	#Chunks	$5\times$	$10\times$	$15\times$	$34\times$	Ratio	Saving
1B	ID	96,496	11.95%	3.00%	4.62%	80.43%	18.65×	73.18%
1B	OOD	12,405	20.32%	8.49%	13.34%	57.85%	13.33×	62.48%
3B	ID	96,496	10.06%	3.41%	3.87%	82.65%	19.83×	74.79%
3B	OOD	12,405	13.23%	8.06%	10.50%	68.21%	16.24×	69.21%
8B	ID	96,496	5.04%	5.50%	6.10%	83.37%	22.64×	77.92%
8B	OOD	12,405	5.14%	12.49%	15.45%	66.92%	18.96×	73.63%

Backbone-specific predictor construction. We regenerate final-layer features and ΔNLL labels for each backbone using the same 30K MRQA training data. Predictors share the cross-attention architecture, with input dimensions adapted to each backbone. All scales use Pathwise routing at $\tau = 5\%$.

F Additional Oracle Routing Analysis

We detail the oracle evaluation in Section 4.6, the construction of Figure 1, and the resulting routing distributions and per-subset performance.

F.1 Oracle Construction

The oracle analysis uses 5,000 MRQA training examples held out from both the 320K examples used for compressor finetuning and the data used to train the distortion predictor. The Predicted Router is the deployed AdaMem predictor with Pathwise routing at $\tau = 5\%$; both oracles below require reference answers and are used only for analysis.

The *Chunk Distortion Oracle* replaces predicted distortions with measured single-chunk ΔNLL while retaining the same routing rule and tolerance. Each measurement changes one chunk to $r \in \{10, 15, 34\}$ and keeps all others at $5\times$. This requires $5{,}000 + 3 \times 8{,}024 = 29{,}072$ NLL evaluations, including one all-$5\times$ reference per example.

The *Exact Task Oracle* exhaustively evaluates routes $R = (r_1, \ldots, r_N) \in \mathcal{R}^N$ for each N-chunk example and selects

$$R^* \in \arg\min_{R \in \mathcal{R}^N} \sum_{i=1}^{N} K(r_i) \quad \text{subject to} \quad \mathrm{F1}(R) \geq \mathrm{F1}(R_{5\times}),$$

where $K(r_i) = 510/r_i$, $R_{5\times} = (5, \ldots, 5)$, and F1 denotes ROUGE-1 F1 against the reference answer. Answer scores break ties in memory cost. The search covers 93,044 routes across the holdout set and gives the maximum attainable compression within $\mathcal{R}$ under this per-example quality constraint. Aggregate memory savings and nominal ratios follow Section 4.1.

F.2 Construction of the Motivation Figure

Figure 1 uses the 24,072 measured single-chunk distortions from the same oracle holdout set. Distortion is the change in teacher-forced reference-answer NLL, averaged over answer tokens using natural logarithms. Thus, $\exp(d_i(r))$ is the answer-perplexity ratio relative to the all-$5\times$ reference, and the 5% tolerance corresponds to $T_{5\%} = \ln(1.05) \approx 0.0488$. Negative distortion indicates lower NLL, without implying higher generated-answer F1.

The violin shapes are Gaussian-smoothed histograms of labels within $[-0.16, 0.35]$, obtained by rounding the pooled 5th and 92.5th percentiles outward. Labels outside this interval are excluded only from the violin shapes; medians, quartiles, and threshold-pass rates use all 8,024 labels per ratio. Solid lines mark medians and dashed lines mark the 25th and 75th percentiles. The pass rates are 79.7%, 74.0%, and 66.4% at $10\times$, $15\times$, and $34\times$, respectively. No label equals the threshold, so strict and non-strict comparisons give identical rates.

F.3 Routing Distribution

Table 18: Chunk-level routing distributions over the 8,024 chunks in the 5K MRQA oracle holdout set. Percentages pool all chunks; the Predicted Router and Chunk Distortion Oracle use Pathwise routing at $\tau = 5\%$.

Method	$5\times$	$10\times$	$15\times$	$34\times$
Predicted Router	13.38%	3.25%	4.86%	78.50%
Chunk Distortion Oracle	20.25%	10.21%	9.66%	59.88%
Exact Task Oracle	2.74%	1.93%	5.02%	90.30%

Table 18 reports the selected ratios. The Chunk Distortion Oracle assigns $34\times$ to 59.88% of chunks, below the 66.4% threshold-pass rate in Figure 1, because Pathwise routing also requires the lower ratios to pass. The Predicted Router and Exact Task Oracle select $34\times$ more often (78.50% and 90.30%, respectively), yet achieve different quality–compression trade-offs (Table 6); aggressive-ratio frequency alone does not determine routing quality.

F.4 Subset-level Analysis

Table 19: Per-subset oracle results on the 5K holdout set. QA and Chunks report sample counts; scores are ROUGE-1 F1/EM averaged within each subset. Compression ratios are nominal.

			Fixed $5\times$		Predicted Router			Chunk Distortion Oracle			Exact Task Oracle		
Dataset	QA	Chunks	F1	EM	F1	EM	Ratio	F1	EM	Ratio	F1	EM	Ratio
HotpotQA	723	727	83.56	73.31	82.61	71.78	$21.00\times$	85.49	76.63	$13.78\times$	86.71	77.73	$25.98\times$
NaturalQuestionsShort	1,028	1,240	68.40	49.12	66.43	46.30	$11.54\times$	70.77	52.82	$10.51\times$	72.55	53.11	$22.85\times$
NewsQA	637	1,083	58.20	36.11	58.18	33.60	$9.88\times$	62.03	39.87	$10.19\times$	64.94	40.66	$22.85\times$
SQuAD	796	798	75.88	57.66	75.26	55.90	$10.59\times$	79.63	62.56	$10.08\times$	80.92	61.68	$20.99\times$
SearchQA	1,195	2,849	70.53	58.33	70.26	58.24	$33.88\times$	74.35	63.43	$18.36\times$	73.92	62.76	$32.07\times$
TriviaQA	621	1,327	66.67	58.29	63.67	54.59	$32.33\times$	69.61	61.19	$14.99\times$	69.62	61.03	$30.88\times$

The Chunk Distortion Oracle improves ROUGE-1 F1 over fixed $5\times$ on all six subsets (Table 19). The Exact Task Oracle attains nominal ratios of approximately 21–$26\times$ on HotpotQA, NaturalQuestionsShort, NewsQA, and SQuAD, and above $30\times$ on SearchQA and TriviaQA, indicating substantial variation in attainable compression across datasets under the same per-example F1 constraint.