LATENTWEAVE: RETHINKING CONTEXT COMPRESSION THROUGH RESIDUAL MEMORY ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference imposes growing demands on computation, memory capacity, and data movement. Soft-token compression reduces these costs, yet preserving relevant information does not guarantee its effective use by a frozen language model. We decompose excess predictive log-loss into compression-induced information loss and predictive mismatch, showing that lossy compression can improve prediction when the reduction in mismatch outweighs the information lost. Guided by this analysis, we introduce **LatentWeave**, a residual memory adaptation framework that jointly optimizes memory construction and representation adaptation. LatentWeave couples a compressor with a shared low-rank residual adapter that applies nonlinear, content-dependent corrections to the complete history-memory prefix, including embeddings from uncompressed segments. Target-derived key-value (KV) and attention supervision initialize the adapter, after which both components are jointly refined through the frozen target. On LongMemEval, LatentWeave achieves higher mean overall accuracy than the uncompressed Qwen3-8B target across all evaluated nominal assistant compression factors from $8\times$ to $256\times$, while reducing end-to-end inference latency by 39.3–49.5%. Across Qwen2.5-14B, Qwen3-8B, Qwen3.5-4B, and Qwen3.8-27B, LatentWeave also consistently outperforms the leading soft-token compression method LatentPress in mean overall and user-fact accuracy, with relative gains of up to 27.5% in mean overall accuracy. Component ablations further show that joint compressor–adapter refinement achieves lower reference-answer log-loss than updating either component alone. LatentWeave brings memory construction and representation adaptation into a unified framework, allowing context compression to improve predictive quality while reducing inference cost.

1 INTRODUCTION

Long contexts are increasingly central to large language models (LLMs) (Liu et al., 2025), supporting document understanding (Liu et al., 2024), conversational memory (Wu et al., 2025), and agentic workflows (Yao et al., 2023; Wang et al., 2024). However, processing longer contexts places growing demands on computation, memory capacity, and data movement (Zhou et al., 2024). Longer inputs increase computation and memory traffic during prefill, while larger key-value (KV) caches increase memory consumption and the computational and memory-access costs (Kwon et al., 2023; Dao et al., 2023). Together, these costs constrain inference efficiency and serving capacity, making context compression an important route to efficient and scalable long-context inference.

KV-cache eviction reduces cache storage and decoding costs by retaining a subset of historical KV states according to estimated importance (Zhang et al., 2023; Li et al., 2024; Cai et al., 2025). However, discarded states may contain information needed by later queries or generation steps. Attention sparsification reduces the computational and memory-access costs of attention by restricting query–key interactions during prefill or decoding (Jiang et al., 2024a; Lai et al., 2025; Tang et al., 2024). Yet attention sparsification alone leaves the projection and feed-forward computations over the full input unchanged during prefill, so attention-level speedups need not translate into comparable end-to-end gains. Compared with these approaches, soft-token compression shortens the target model's input sequence by encoding the original context into compact continuous memories (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024). This can reduce both target-model prefill computation and the KV-cache storage and attention costs of subsequent decoding.

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.

However, preserving information alone does not guarantee that compressed memory supports accurate predictions from a frozen target. Our analysis explains this gap by decomposing excess predictive log-loss into compression-induced information loss and predictive mismatch. The former measures answer-relevant information lost along the memory path; the latter measures the gap between the frozen target's predictions and the optimal predictor given its input. This decomposition shows that lossy compression can yield lower predictive log-loss than uncompressed inputs when the reduction in predictive mismatch outweighs the information lost. Effective compression therefore requires attention to both the information retained and the predictions induced by its representation. This perspective motivates jointly learning memory construction and representation adaptation, allowing both to be shaped by the frozen target's predictive feedback.

Building on this, we introduce **LatentWeave**, a residual memory adaptation framework that jointly learns memory construction and representation adaptation. The compressor constructs a history-memory prefix from pooled representations of compressed segments and token embeddings of uncompressed segments. A shared low-rank residual adapter then applies nonlinear, content-dependent corrections to every vector in this prefix. Training progresses from KV-guided adapter initialization to joint predictive refinement. Target-derived KV and attention supervision guide the initial representation correction, after which answer supervision and output-distribution distillation jointly optimize the compressor and the adapter. Predictive feedback can therefore reshape both the constructed memory and the residual transformation applied across the history prefix. At inference, the adapted prefix is processed through standard target-model prefill and autoregressive decoding.

We evaluate LatentWeave on LongMemEval (Wu et al., 2025). Across all evaluated nominal assistant compression factors from $8\times$ to $256\times$, LatentWeave exceeds the uncompressed Qwen3-8B target by 0.28–1.44 percentage points in mean overall accuracy while reducing end-to-end inference latency by 39.3–49.5%. Comparisons with the leading soft-token compression method LatentPress (Zhou & Sang, 2026) further demonstrate improvements across Qwen2.5-14B, Qwen3-8B, Qwen3.5-4B, and Qwen3.8-27B. LatentWeave consistently improves mean overall and user-fact accuracy, with up to 27.5% relative gains in overall accuracy. Component ablations show that adapting fixed compressed memories reduces reference-answer log-loss, while joint refinement achieves lower mean log-loss than updating either component alone. Together, these results support treating information preservation and representation adaptation as joint objectives of context compression: learning not only what to preserve, but also how to make compressed information usable.

2 Related Work

2.1 KV Cache Eviction

KV-cache eviction reduces memory consumption by retaining selected historical key–value states. H_2O combines attention-based heavy hitters with recent tokens (Zhang et al., 2023), SnapKV identifies important positions for each attention head using an observation window (Li et al., 2024), and PyramidKV allocates cache budgets across layers according to attention concentration (Cai et al., 2025). These methods manage the quality–memory trade-off by selecting which existing states to retain. However, states assigned low importance under observed attention may become relevant to later queries. Eviction performed after prefill also leaves the cost of encoding the full context unchanged. LatentWeave constructs compact memories before target-model prefill and learns their representations through joint optimization of a compressor and an adapter.

2.2 Attention Sparsification

Attention sparsification reduces the query–key interactions evaluated during inference. For prefill, MInference constructs head-specific sparse attention indices dynamically (Jiang et al., 2024a), while FlexPrefill adapts sparsity patterns and budgets to the input and attention head (Lai et al., 2025). For decoding, QUEST selects KV pages using query-dependent importance estimates (Tang et al., 2024). These approaches reduce attention computation or KV access without necessarily shortening the input sequence. Attention sparsification leaves token-wise projection and feed-forward computation intact during prefill, and its end-to-end benefits depend on the attention costs removed and the overhead of sparse selection. LatentWeave addresses a complementary source of cost by shortening the sequence processed by the target model while retaining its standard attention computation.

2.3 SOFT-TOKEN COMPRESSION

Soft-token compression learns compact continuous memories, as explored by Gisting, AutoCompressors, and ICAE (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024). Other compression routes include discrete token selection in LLMLingua series (Jiang et al., 2023; 2024b; Pan et al., 2024), and optical encoding for text reconstruction in DeepSeek-OCR (Wei et al., 2025). xRAG bridges retrieval embeddings to frozen language models (Cheng et al., 2024), while LatentPress learns reader-matched soft tokens through reconstruction and distribution distillation (Zhou & Sang, 2026). LatentWeave introduces an explicit residual adaptation stage after memory construction. A shared nonlinear low-rank adapter refines both pooled memory vectors and embeddings from uncompressed segments. Target-derived KV and attention supervision initialize this adapter, followed by joint compressor–adapter optimization through predictive feedback from the frozen target.

3 INFORMATION PRESERVATION AND PREDICTIVE MISMATCH

Context compression changes both the information available to a frozen target model and the representation through which that information is used. We analyze these effects through predictive log-loss. Let H denote the history, X the question and answer prefix, and W the next answer token. A deterministic map g, fixed before evaluation, produces memory $U = g(H)$ for the frozen target p_θ. All expectations are taken under a common evaluation distribution ν.

When can compressed memory improve prediction? Let $r_H = p_\theta(\cdot \mid H, X)$ and $s_U = p_\theta(\cdot \mid U, X)$ denote target's predictions from raw history and compressed memory. The corresponding true conditional token distributions are $\pi_H = \nu(W \mid H, X)$ and $\pi_U = \nu(W \mid U, X)$. Define the predictive risks $\mathcal{R}_{\mathrm{R}} = \mathbb{E}_\nu[-\log r_H(W)]$ and $\mathcal{R}_{\mathrm{L}} = \mathbb{E}_\nu[-\log s_U(W)]$, with mismatch terms

$$\mathcal{D}_{\mathrm{R}} = \mathbb{E}_\nu \,\mathrm{KL}(\pi_H \| r_H), \qquad \mathcal{D}_{\mathrm{L}} = \mathbb{E}_\nu \,\mathrm{KL}(\pi_U \| s_U). \tag{1}$$

Each mismatch measures the gap between the target's predictions and the optimal predictor given its input. Let $\delta = I_\nu(W; H \mid U, X)$ denote the answer-relevant information lost through compression.

Theorem 1 (Information loss and predictive mismatch)**.** *For a fixed deterministic memory map and finite predictive risks,*

$$\begin{aligned} \mathcal{R}_{\mathrm{R}} &= \mathsf{H}_\nu(W \mid H, X) + \mathcal{D}_{\mathrm{R}}, \\ \mathcal{R}_{\mathrm{L}} &= \mathsf{H}_\nu(W \mid H, X) + \delta + \mathcal{D}_{\mathrm{L}}. \end{aligned} \tag{2}$$

Consequently,

$$\mathcal{R}_{\mathrm{R}} - \mathcal{R}_{\mathrm{L}} = \mathcal{D}_{\mathrm{R}} - \mathcal{D}_{\mathrm{L}} - \delta. \tag{3}$$

Lossy compression therefore need not worsen a frozen model's predictions. Compressed memory yields lower predictive log-loss precisely when $\mathcal{D}_{\mathrm{R}} - \mathcal{D}_{\mathrm{L}} > \delta$: the reduction in predictive mismatch outweighs the information lost. This identifies two aspects of compression quality: the information retained in memory and how effectively its representation supports the target's predictions. The proof is provided in Appendix A.2.

What can memory adaptation change? To distinguish memory construction from representation adaptation, consider the factorization $Z = C_\phi(H)$ and $U = A_\psi(Z)$. The compressor C_ϕ constructs an intermediate memory, and the adapter A_ψ transforms its representation before it enters the target. Along $H \to Z \to U$, information loss decomposes as

$$\delta = I_\nu(W; H \mid Z, X) + I_\nu(W; Z \mid U, X). \tag{4}$$

The first term measures information discarded by the compressor, and the second measures any additional loss introduced by adaptation. With the compressor fixed, adaptation cannot change the first term or recover discarded information. It can still revise how retained information is presented to the target and thereby change predictive mismatch.

Updating the compressor as well allows predictive feedback to reshape the intermediate memory itself. Both components can affect total information loss and final predictive mismatch through the composed memory $U = A_\psi(C_\phi(H))$. This motivates jointly optimizing memory construction and representation adaptation, so that the information retained and its representation respond to the same predictive feedback.

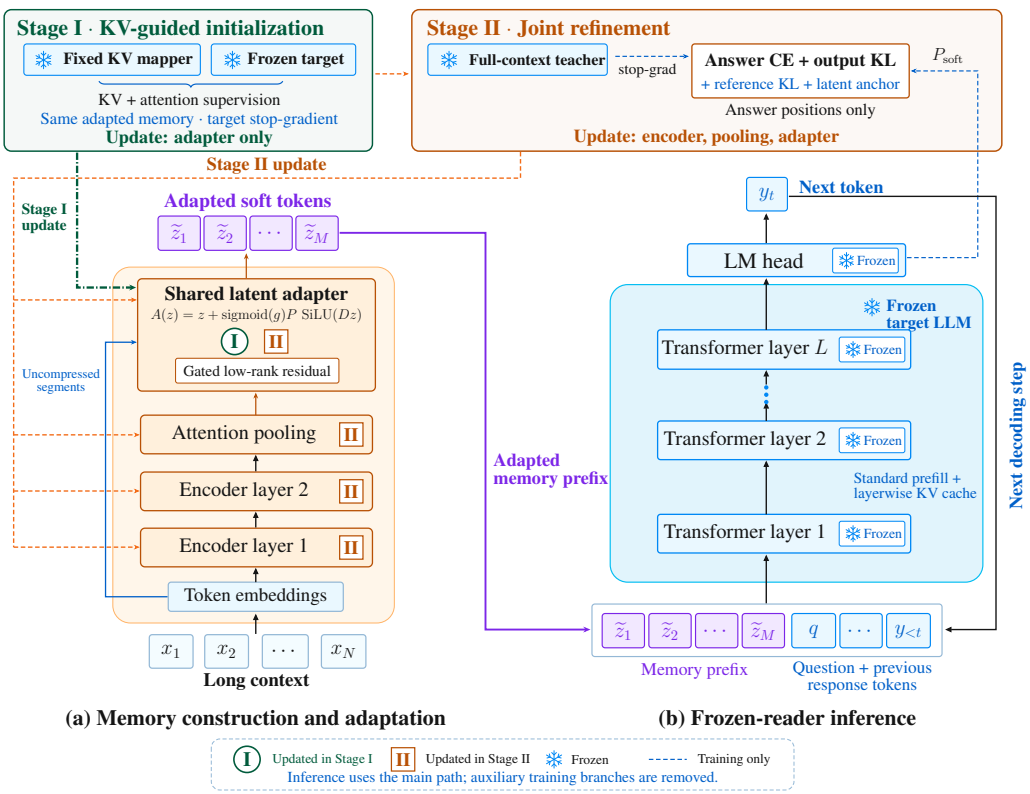

Figure 1: LatentWeave couples memory construction with residual representation adaptation, with $\widetilde{Z}$ corresponding to U in the text. Circular I and square II mark parameter updates during KV-guided adapter initialization and joint compressor–adapter refinement, respectively. Uncompressed segments bypass encoding and pooling but pass through the adapter. At inference, the frozen target builds its KV cache through standard prefill of the adapted memory.

From the analysis to residual memory adaptation. LatentWeave implements this perspective with a shared low-rank residual adapter that learns content-dependent corrections to the constructed memory. The residual form retains a direct path from the compressor's output while allowing its representation to be refined. Target-derived KV and attention supervision guide adapter initialization, after which answer supervision and output-distribution distillation jointly refine the compressor and adapter through the frozen target. Section 4 presents the architecture and training objectives.

4 LatentWeave: Residual Memory Adaptation

LatentWeave constructs a compact history-memory prefix and then refines its representations with a shared low-rank residual adapter. The adapter acts on the complete prefix, including embeddings from uncompressed segments, before it enters the frozen target. Training proceeds from KV-guided adapter initialization to joint predictive refinement of the compressor and adapter (Figure 1).

4.1 Memory Construction and Residual Adaptation

The memory-construction map produces $Z = C_\phi(H) \in \mathbb{R}^{M \times d}$, where M is the number of memory slots and d is the target embedding dimension. Here, Z denotes the complete history prefix. For compressed segments, a two-layer encoder produces contextual representations whose gated projections are combined with original token embeddings. Attention pooling then produces one memory vector per token group. Uncompressed segments bypass encoding and pooling and contribute their original token embeddings. The resulting vectors are assembled in history order to form Z.

We apply the same residual adapter to every vector z in Z:

$$A_\psi(z) = z + \gamma P\,\mathrm{SiLU}(Dz), \qquad \gamma = \sigma(g). \tag{5}$$

Here, $D \in \mathbb{R}^{k\times d}$ and $P \in \mathbb{R}^{d\times k}$ define a low-rank bottleneck with $k \ll d$, and the learned scalar gate γ controls the correction magnitude. The residual connection retains a direct path from the original memory vector, while the nonlinear branch learns a content-dependent correction. For each target model, adapter parameters are shared across memory positions and examples, and each correction depends only on its input vector z.

Applying the adapter to the complete prefix yields $U = A_\psi(Z)$, preserving the number and order of memory slots. Uncompressed segments therefore retain their token count but receive adapted embeddings. The frozen target consumes U together with the question to produce its answer.

4.2 Stage I: KV-guided adapter initialization

Stage I provides internal feature guidance for residual memory adaptation before joint predictive refinement. Only the adapter parameters ψ are updated. The compressor, target, and pretrained auxiliary KV mapper P_ω remain fixed. The mapper predicts KV features from the adapted memory U, while the target processes the same U to provide detached feature targets:

$$(\widehat{K}_l, \widehat{V}_l) = P_{\omega,l}(U, Q), \qquad (K_l^*, V_l^*) = \mathrm{sg}[F_{\theta,l}(U)]. \tag{6}$$

Here $F_{\theta,l}$ extracts layer-l target KV features, and sg denotes stop-gradient. These targets are recomputed from the current U at each step. The mapper also receives a pooled embedding of the question Q through a training-only routing branch. We match KV features at corresponding memory positions using normalized mean squared error:

$$\mathcal{L}_{\mathrm{KV}} = \frac{1}{L}\sum_l \left[\mathrm{NMSE}(\widehat{K}_l, K_l^*) + \mathrm{NMSE}(\widehat{V}_l, V_l^*)\right],$$
$$\mathrm{NMSE}(A, B) = \frac{\mathrm{mean}[(A-B)^2]}{\max\{\mathrm{mean}(B^2), \epsilon_{\mathrm{KV}}\}}. \tag{7}$$

Here L is the number of target layers and $\epsilon_{\mathrm{KV}} = 10^{-4}$. Keys are matched before key normalization and rotary positional transformations. To guide how answer positions attend to the adapted memory, we match attention distributions from normalized queries and keys after rotary transformations:

$$B_l^* = \mathrm{sg}\Big[\mathrm{softmax}_{\mathrm{hist}}(\bar{Q}_l^* \bar{K}_l^{*\top}/\sqrt{d_h})\Big],$$
$$\widehat{B}_l = \mathrm{softmax}_{\mathrm{hist}}(\widehat{\bar{Q}}_l \widehat{\bar{K}}_l^{\top}/\sqrt{d_h}),$$
$$\mathcal{L}_{\mathrm{att}} = \mathrm{mean}_{l,t,h}\,\mathrm{KL}(B_{l,t,h}^* \| \widehat{B}_{l,t,h}). \tag{8}$$

Here d_h is the head dimension, and the mean covers selected layers, answer positions, and heads. Both branches use the same prefixes, positions, and head assignments. The history-only normalization compares relative attention among memory slots. The Stage-I objective combines feature alignment with auxiliary predictive supervision and regularization:

$$\mathcal{L}_{\mathrm{I}} = 0.10\mathcal{L}_{\mathrm{KV}} + 0.02\mathcal{L}_{\mathrm{att}} + \mathcal{L}_{\mathrm{aux}}. \tag{9}$$

The auxiliary term includes query and attention-output matching, answer supervision, response distillation, and reference regularization. Appendix E details the auxiliary terms.

4.3 Stage II: joint memory construction and adaptation

Stage II jointly updates the compressor's two encoder layers and attention pooling together with the residual adapter, using predictive feedback through the frozen target without the auxiliary KV mapper. To regularize joint refinement, we fix a reference branch with the compressor's unchanged Stage-I parameters ϕ_{ref} and a copy of the adapter parameters ψ_{ref} at the start of the Stage-II run:

$$U_{\mathrm{ref}} = A_{\psi_{\mathrm{ref}}}(C_{\phi_{\mathrm{ref}}}(H)).$$

For runs initialized from a Stage-I checkpoint, this is the memory produced after Stage I. The reference parameters remain fixed while the current memory $U = A_\psi(C_\phi(H))$ evolves during training.

At answer position t, let $p_{\mathrm{raw},t}$, $p_{\mathrm{soft},t}$, and $p_{\mathrm{ref},t}$ denote target next-token distributions conditioned on H, U, and U_{ref}, respectively. All branches receive the same question and ground-truth answer prefix. The objective combines answer supervision, output-distribution distillation (Hinton et al., 2015), and reference regularization:

$$\mathcal{L}_{\mathrm{II}} = \lambda_y \mathcal{L}_{\mathrm{CE}} + \frac{\lambda_r}{T} \sum_t \mathrm{KL}(p_{\mathrm{raw},t} \| p_{\mathrm{soft},t}) + \frac{\lambda_b}{T} \sum_t \mathrm{KL}(p_{\mathrm{ref},t} \| p_{\mathrm{soft},t}) + \mathcal{L}_{\mathrm{anchor}}, \tag{10}$$

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{T} \sum_t \log p_{\mathrm{soft},t}(y_t),$$

$$\mathcal{L}_{\mathrm{anchor}} = \frac{\mathrm{mean}[(U - U_{\mathrm{ref}})^2]}{\max\{\mathrm{mean}(U_{\mathrm{ref}}^2), 10^{-4}\}}. \tag{11}$$

Here T counts supervised answer positions, y_t is the ground-truth token, and $(\lambda_r, \lambda_b, \lambda_y) = (1, 0.25, 0.05)$. Answer supervision and distillation from the uncompressed target guide predictive refinement. The reference KL and memory anchor regularize changes in predictions and memory representations, respectively. Teacher and reference quantities are detached. Gradients through the frozen target update the current compressor and adapter, allowing both memory construction and residual adaptation to respond to the same predictive feedback.

4.4 Inference

At inference, the compressor encodes the history into compact memory, and the shared adapter refines it to obtain U. The adapted memory is concatenated with question embeddings and passed to the frozen target through standard prefill. The target constructs its own KV cache and uses it for subsequent autoregressive decoding. Deployment requires only the trained compressor, adapter, and frozen target. The auxiliary KV mapper, teacher, and reference branches are removed.

5 Experiments and Discussion

We evaluate LatentWeave on LongMemEval, comparing answer accuracy with the uncompressed target and existing compression methods, and examining performance across target models of different generations and scales. We measure end-to-end latency including online memory construction and adaptation. Finally, component ablations assess the contributions of residual adaptation and joint refinement to reference-answer log-loss. Training details are provided in Appendix E.

5.1 Evaluation Protocol

We evaluate on LongMemEval (Wu et al., 2025), with each question paired with its evidence-bearing sessions to assess answering independently of retrieval. Qwen3-8B (Yang et al., 2025) serves as the primary target model, with additional evaluations on Qwen2.5-14B-Instruct (Yang et al., 2024; Qwen Team, 2024), Qwen3.5-4B (Qwen Team, 2026a), and Qwen3.8-27B (Qwen Team, 2026b). Inference uses BF16 and greedy decoding with a 64-token answer limit, with thinking disabled through the chat template where supported. We use DeepSeek-V4-Flash-0731 (DeepSeek-AI et al., 2026; DeepSeek-AI, 2026) as an automated correctness judge.

We report overall accuracy and single-session-user accuracy (user-fact), which measures recovery of explicitly stated user facts. Overall accuracy also covers multi-session aggregation, temporal reasoning, knowledge updates, and abstention. Baselines include Raw, LatentPress (Zhou & Sang, 2026), LLMLingua (Jiang et al., 2023), DeepSeek-OCR (Wei et al., 2025), and ICAE (Ge et al., 2024). Raw uses the original history with the same target model and prompt format.

Repeated experiments use five independent seeds. Trainable methods are retrained and evaluated for each seed under the same configuration. We report mean $\pm$ standard error (SE). LatentPress and LatentWeave leave user turns uncompressed and compress longer assistant turns. Their nominal factors therefore apply only to assistant content. Method suffixes denote configured factors, while *history ratio* denotes the original history length divided by the compressed history length.

Table 1: Answer accuracy on LongMemEval with Qwen3-8B. Higher accuracy is better.

Method	Factor	History ratio	Overall (%)	User-fact (%)
Raw	1×	1.00×	52.00 ± 0.49	80.62 ± 0.86
LatentPress	8×	4.56×	50.92 ± 0.64	92.19 ± 0.00
	16×	6.20×	50.72 ± 0.30	92.19 ± 0.00
	32×	7.60×	50.76 ± 0.36	90.62 ± 0.00
	64×	8.63×	50.96 ± 0.22	92.19 ± 0.00
	128×	9.28×	51.32 ± 0.48	93.75 ± 0.00
	256×	9.65×	50.96 ± 0.36	93.75 ± 0.00
LLMLingua	2×	1.97×	35.64 ± 0.43	50.00 ± 0.00
	4×	3.76×	25.56 ± 0.33	21.88 ± 0.00
	8×	6.68×	20.08 ± 0.82	12.50 ± 0.00
	16×	12.19×	14.16 ± 0.43	4.69 ± 0.00
	32×	17.87×	9.44 ± 0.38	2.81 ± 1.71
DeepSeek-OCR	–	4.48×	32.76 ± 0.17	46.88 ± 0.00
	–	10.08×	7.56 ± 0.17	0.00 ± 0.00
ICAE	2×	1.917×	18.68 ± 0.39	22.50 ± 0.86
	4×	3.834×	19.68 ± 0.69	27.19 ± 0.86
	8×	7.668×	17.20 ± 0.32	20.94 ± 1.78
	16×	15.336×	20.40 ± 0.75	26.56 ± 2.21
LatentWeave	8×	4.56×	**52.60 ± 0.68**	**93.75 ± 0.00**
	16×	6.20×	**53.44 ± 0.26**	**98.44 ± 0.00**
	32×	7.60×	**52.28 ± 0.36**	**96.88 ± 0.00**
	64×	8.63×	**53.00 ± 0.20**	**98.44 ± 0.00**
	128×	9.28×	**52.88 ± 0.18**	**96.88 ± 0.00**
	256×	9.65×	**52.28 ± 0.36**	**98.44 ± 0.00**

5.2 Answer Accuracy under Compression

To assess answer quality under context compression, we compare LatentWeave with Raw, LatentPress, LLMLingua, DeepSeek-OCR, and ICAE using Qwen3-8B on LongMemEval (Table 1). Across nominal assistant factors from $8\times$ to $256\times$, LatentWeave achieves 52.28–53.44% mean overall accuracy, exceeding Raw by 0.28–1.44 percentage points and LatentPress by 1.32–2.72 points. User-fact accuracy reaches 93.75–98.44%, compared with 80.62% for Raw and 90.62–93.75% for LatentPress. LatentWeave improves both metrics over LatentPress at every corresponding factor. These results show context compression need not compromise answer quality: LatentWeave produces compact memories support higher accuracy than original context across all evaluated settings.

LatentWeave also exceeds LLMLingua, DeepSeek-OCR, and ICAE on both accuracy metrics at their evaluated settings. LLMLingua and DeepSeek-OCR lose accuracy as compression increases, while ICAE remains at 17.20–20.40% overall accuracy. LatentWeave and LatentPress use the same realized history ratio at every matched nominal factor, ranging from $4.56\times$ to $9.65\times$ in this experiment. Their accuracy can therefore be compared under identical realized memory budgets.

5.3 End-to-End Inference Latency

We measure end-to-end latency on B200, including online compression, residual adaptation, target prefill, and decoding. Raw uses the same generation protocol with uncompressed history. LatentWeave requires 0.444–0.534 seconds per example, compared with 0.880 seconds for Raw, reducing mean latency by 39.3–49.5% across the evaluated factors (Table 2). These savings include the cost of constructing and adapting memory. Together with Table 1, the results show higher mean overall accuracy and lower end-to-end latency than Raw at every evaluated setting.

Table 2: End-to-end inference latency on B200, reported as mean ± SE over five runs.

Method	Factor	History ratio	Latency (s) ↓	Reduction (%)
Raw	1×	1.00×	0.880 ± 0.021	–
LatentWeave	8×	4.56×	**0.534 ± 0.004**	39.3
	16×	6.20×	**0.511 ± 0.003**	41.9
	32×	7.60×	**0.485 ± 0.002**	44.9
	64×	8.63×	**0.474 ± 0.006**	46.1
	128×	9.28×	**0.458 ± 0.005**	48.0
	256×	9.65×	**0.444 ± 0.002**	49.5

Table 3: Accuracy across frozen target models with separately trained memory interfaces.

Method	Factor	History ratio	Overall (%)	User-fact (%)
Qwen2.5-14B				
LatentPress	8×	4.62×	56.24 ± 0.50	95.31 ± 0.00
	16×	6.27×	56.40 ± 0.42	94.69 ± 0.86
	32×	7.70×	56.28 ± 0.46	95.94 ± 1.40
LatentWeave	8×	4.62×	**62.12 ± 0.44**	**96.88 ± 1.10**
	16×	6.27×	**62.88 ± 0.30**	**98.44 ± 0.00**
	32×	7.70×	**62.20 ± 0.62**	**97.19 ± 0.70**
Qwen3.5-4B				
LatentPress	8×	4.63×	30.20 ± 0.37	82.81 ± 0.00
	16×	6.29×	29.52 ± 0.30	82.81 ± 0.00
	32×	7.73×	27.36 ± 0.46	83.12 ± 0.70
LatentWeave	8×	4.63×	**31.16 ± 0.30**	**89.38 ± 0.70**
	16×	6.29×	**33.16 ± 0.33**	**92.50 ± 0.70**
	32×	7.73×	**34.88 ± 0.33**	**89.06 ± 0.00**
Qwen3.8-27B				
LatentPress	8×	4.63×	51.56 ± 0.38	89.06 ± 0.00
	16×	6.29×	51.88 ± 0.41	89.38 ± 0.70
	32×	7.73×	52.76 ± 0.52	87.50 ± 0.00
LatentWeave	8×	4.63×	**52.56 ± 0.59**	**90.31 ± 0.70**
	16×	6.29×	**55.68 ± 0.54**	**91.88 ± 0.70**
	32×	7.73×	**52.80 ± 0.79**	**88.12 ± 0.86**

5.4 Generalization across Target Models

To assess applicability beyond Qwen3-8B, we compare LatentWeave with LatentPress on Qwen2.5-14B-Instruct, Qwen3.5-4B, and Qwen3.8-27B at nominal assistant compression factors of 8×, 16×, and 32×. Each target model remains frozen and uses a separately trained compressor and adapter.

As shown in Table 3, LatentWeave achieves higher mean overall and user-fact accuracy than LatentPress in all nine reader–factor comparisons. Mean overall accuracy improves by 5.88–6.48 percentage points for Qwen2.5-14B, 0.96–7.52 points for Qwen3.5-4B, and 0.04–3.80 points for Qwen3.8-27B. These results support the applicability of LatentWeave across target models of different generations and sizes, with a separately trained compressor and adapter for each model.

5.5 Ablation Studies

We examine the contributions of residual adaptation, KV guidance, and joint refinement through controlled comparisons of memory interfaces. All conditions use the same frozen Qwen3-8B target, histories, questions, and teacher-forced answer prefixes. We compute log-loss over complete reference answers including EOS, averaging first within each answer and then equally across ques-

Table 4: Reference-answer log-loss under different memory interfaces and update conditions.

Configuration	Compressor	Adapter	Log-loss (nats/token) ↓
Raw	–	–	6.445
No adapter	Fixed	Absent	5.0806
Initialized interface	Fixed	Fixed	2.8616
Adapter-only refinement	Fixed	Updated	2.6145 ± 0.0293
Compressor-only refinement	Updated	Fixed	2.7964 ± 0.0151
Joint w/o KV guidance	Updated	Updated	2.6820 ± 0.0231
Joint refinement	Updated	Updated	$\mathbf{2.4871 \pm 0.0422}$

tions. This provides an empirical counterpart to the predictive log-loss analyzed in Section 3. All compressed-memory conditions use a nominal assistant factor of $16\times$.

Compressed memory versus raw context. We first compare the jointly refined memory with the original history. LatentWeave reduces mean reference-answer log-loss from 6.445 to 2.4871 ± 0.0422 nats/token (Table 4). This provides direct evidence that compressed memory can improve predictive log-loss for the frozen target, complementing the answer-accuracy results.

Residual adaptation with fixed memory. To examine representation adaptation independently of compressor updates, we keep the compressor fixed and vary the adapter. Using the compressor output directly yields a log-loss of 5.0806. Applying the initialized adapter reduces it to 2.8616, and adapter-only refinement further reduces it to 2.6145. These improvements show that the same compressor output can become more useful to the frozen target through changes to its representation.

Joint compressor–adapter refinement. We compare adapter-only, compressor-only, and joint refinement from the same initialized interface. Joint refinement achieves the lowest mean log-loss, outperforming adapter-only and compressor-only refinement by 0.1274 and 0.3093 nats/token, respectively. This comparison supports allowing predictive feedback to refine both the constructed memory and its residual adaptation.

Contribution of KV guidance. The jointly refined variant trained without KV guidance achieves a mean log-loss of 2.6820 ± 0.0231 nats/token, compared with 2.4871 ± 0.0422 for full LatentWeave. Incorporating KV guidance therefore reduces mean log-loss by 0.1949 nats/token, or 7.27%. This comparison supports the contribution of KV guidance to predictive log-loss reduction beyond residual adaptation and joint refinement without this guidance.

6 Conclusion

We presented LatentWeave, a residual memory adaptation framework that jointly optimizes memory construction and representation adaptation for a frozen language model. Our analysis shows that lossy compression can reduce predictive log-loss relative to uncompressed inputs when the reduction in predictive mismatch outweighs compression-induced information loss. LatentWeave implements this perspective with a shared low-rank residual adapter that learns content-dependent corrections to compressed memories. KV and attention supervision guide adapter initialization, followed by joint refinement of the compressor and adapter through the frozen target. In zero-shot evaluation on LongMemEval, LatentWeave achieves higher mean overall accuracy than the uncompressed Qwen3-8B target across all evaluated nominal assistant compression factors from $8\times$ to $256\times$, while reducing end-to-end inference latency by 39.3–49.5%. Its improvements over LatentPress extend across Qwen2.5-14B, Qwen3-8B, Qwen3.5-4B, and Qwen3.8-27B, with higher mean overall and user-fact accuracy at every evaluated nominal assistant factor from $8\times$ to $32\times$ and relative gains of up to 27.5% in mean overall accuracy. Component ablations further support the value of residual adaptation with a fixed compressor and show an additional reduction in mean reference-answer log-loss when the compressor and adapter are jointly refined. These findings support treating information preservation and representation adaptation as joint objectives of context compression, enabling compact memories to improve predictive quality while reducing inference cost.

AI USE STATEMENT

Generative AI tools assisted with conceptual framing, implementation, experiment organization, figure preparation, drafting, and English-language editing. Separately, DeepSeek-V4-Flash-0731 was used as the automated correctness judge following the evaluation protocol in Section 5.1. The authors are responsible for the method, experiments, references, and claims presented in the paper.

REFERENCES

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible Residual Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 573–582. PMLR, 2019. URL https://proceedings.mlr.press/v97/behrmann19a.html.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 109487–109516, 2024. doi: 10.52202/079017-3476. URL https://openreview.net/forum?id=6pTlXqrO0p.

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. Association for Computational Linguistics, December 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-Decoding for Long-Context Inference. Stanford Center for Research on Foundation Models blog, October 2023. URL https://crfm.stanford.edu/2023/10/12/flashdecoding.html.

DeepSeek-AI. DeepSeek-V4-Flash-0731. Hugging Face model release, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash-0731.

DeepSeek-AI et al. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. *arXiv preprint arXiv:2606.19348*, 2026. doi: 10.48550/arXiv.2606.19348. URL https://arxiv.org/abs/2606.19348.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. *arXiv preprint arXiv:2305.14233*, 2023. URL https://arxiv.org/abs/2305.14233.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-Context Autoencoder for Context Compression in a Large Language Model. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

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. Association for Computational Linguistics, December 2023. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52481–52515,

2024a. doi: 10.52202/079017-1663. URL https://papers.nips.cc/paper_files/paper/2024/hash/5dfbe6f5671e82c76841ba687a8a9ecb-Abstract-Conference.html.

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. Association for Computational Linguistics, August 2024b. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. Association for Computing Machinery, 2023. doi: 10.1145/3600006.3613165. URL https://arxiv.org/abs/2309.06180.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/03645743ea35690f30d795d6bac149a5-Abstract-Conference.html.

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*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722. URL https://openreview.net/forum?id=poE54GOq2l.

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*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Xiaoran Liu, Ruixiao Li, Mianqiu Huang, Zhigeng Liu, Yuerong Song, Qipeng Guo, Siyang He, Qiqi Wang, Linlin Li, Qun Liu, Ziwei He, Yaqian Zhou, Xuanjing Huang, and Xipeng Qiu. Thus Spake Long-Context Large Language Model. *arXiv preprint arXiv:2502.17129*, 2025. URL https://arxiv.org/abs/2502.17129.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to Compress Prompts with Gist Tokens. In *Advances in Neural Information Processing Systems*, volume 36, pp. 19327–19352, 2023. doi: 10.52202/075280-0848. URL https://openreview.net/forum?id=2DtxPCL3T5.

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

Qwen Team. Qwen2.5-14B-Instruct. Hugging Face model release, 2024. URL https://huggingface.co/Qwen/Qwen2.5-14B-Instruct. Model checkpoint accessed September 2026.

Qwen Team. Qwen3.5-4B. Hugging Face model release, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-4B. Model checkpoint accessed September 2026.

Qwen Team. Qwen3.8-27B. Hugging Face model release, 2026b. URL https://huggingface.co/Qwen/Qwen3.8-27B. Model checkpoint accessed September 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A Survey on Large Language Model Based Autonomous Agents. *Frontiers of Computer Science*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025. URL https://arxiv.org/abs/2510.18234.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

An Yang et al. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract.html.

Zhengze Zhou and Hejian Sang. LatentPress: Context Compression Beyond Text and Vision. *arXiv preprint arXiv:2609.01507*, 2026. URL https://arxiv.org/abs/2609.01507.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A Survey on Efficient Inference for Large Language Models. *arXiv preprint arXiv:2404.14294*, 2024. URL https://arxiv.org/abs/2404.14294.

A Foundations and Proofs of the Main Results

Context compression changes both the information available for prediction and the input presented to a frozen model. We formalize a common evaluation law and prove the risk decomposition in Section 3. We then state and prove the conditional distillation results, separating an ideal averaging gain from information loss and imperfect realization. Appendix B examines what residual memory adaptation can change within a factorized interface. Appendix C relates auxiliary and predictive objectives to LatentWeave's training, while Appendix D distinguishes token log-loss from sequence prediction and generated-answer accuracy.

A.1 A common law for comparing the two prediction paths

To isolate the effect of the memory interface, raw and compressed predictions are compared on the same examples and answer positions. Let μ be a population law over histories, questions, and answers (H, Q, Y). For each example, let $S(H, Q, Y)$ be a nonempty finite set of answer positions and $T_{\sup} = |S(H, Q, Y)|$. Define a law ν by drawing an example from μ, drawing J uniformly from S, and setting $X = (Q, J, Y_{<J})$ and $W = Y_J$. Equivalently,

$$\mathbb{E}_\nu f(H, X, W) = \mathbb{E}_{(H,Q,Y)\sim\mu} \frac{1}{T_{\sup}} \sum_{t \in S(H,Q,Y)} f\big(H, (Q, t, Y_{<t}), Y_t\big). \tag{12}$$

This construction averages over selected answer positions within each example and weights examples equally. It matches an example-averaged answer-token log-loss, including EOS when EOS belongs to S. The weighting is part of the prediction problem: changing the position law can change the conditional token distributions. In particular, this risk differs from an unnormalized sequence loss when answer lengths vary.

The memory maps $Z = C_\phi(H)$ and $U = A_\psi(Z)$ are measurable and deterministic after fixing their learned parameters. Memory variables include all input metadata, such as positions, masks, and sequence length. Histories and memories take values in standard Borel spaces, and the vocabulary $\mathcal{V}$ is finite. Conditional distributions therefore exist. Both prediction paths use the same history source, question state, and frozen target parameters:

$$r_H(v \mid X) = p_\theta(v \mid H, X), \qquad s_U(v \mid X) = p_\theta(v \mid U, X), \tag{13}$$

$$\pi_H(v) = \nu(W = v \mid H, X), \qquad \pi_U(v) = \nu(W = v \mid U, X). \tag{14}$$

The population risks are

$$\mathcal{R}_{\mathrm{R}} = \mathbb{E}_\nu[-\log r_H(W \mid X)], \qquad \mathcal{R}_{\mathrm{L}} = \mathbb{E}_\nu[-\log s_U(W \mid X)]. \tag{15}$$

We use natural logarithms and the convention $0 \log 0 = 0$. Model probabilities are positive, and the expectations in each statement are assumed finite. Uniform probability floors are imposed only where explicitly stated. All comparisons condition on the learned parameters before taking population expectations. Thus each result concerns a fixed memory map and a specified evaluation law. A change of dataset, prefix distribution, or learned map requires evaluating the corresponding quantities again. Within proofs, $\mathbb{E}$ abbreviates $\mathbb{E}_\nu$, and dependence on (U, X) is sometimes suppressed. We use the main-text definitions $\mathcal{D}_{\mathrm{R}} = \mathbb{E}\,\mathrm{KL}(\pi_H \| r_H)$, $\mathcal{D}_{\mathrm{L}} = \mathbb{E}\,\mathrm{KL}(\pi_U \| s_U)$, and $\delta = I_\nu(W; H \mid U, X)$.

A.2 Separating information loss from predictive mismatch

Theorem 1 separates two sources of predictive error: uncertainty that remains after observing an input, and the gap between the frozen target's predictions and the true conditional law. Comparing these terms across the two inputs gives the condition under which compressed memory can improve on Raw.

Proof of Theorem 1. Adding and subtracting $\log \pi_H(W)$ in the raw log-loss gives

$$\mathcal{R}_{\mathrm{R}} = \mathbb{E}[-\log \pi_H(W)] + \mathbb{E} \log \frac{\pi_H(W)}{r_H(W \mid X)} \tag{16}$$

$$= \mathsf{H}_\nu(W \mid H, X) + \mathbb{E}_{H,X} \sum_v \pi_H(v) \log \frac{\pi_H(v)}{r_H(v \mid X)} \tag{17}$$

$$= \mathsf{H}_\nu(W \mid H, X) + \mathcal{D}_{\mathrm{R}}. \tag{18}$$

Conditioning the same calculation on the deployed input (U, X) gives

$$\mathcal{R}_{\mathrm{L}} = \mathsf{H}_\nu(W \mid U, X) + \mathcal{D}_{\mathrm{L}}. \tag{19}$$

Since U is a deterministic function of H,

$$\mathsf{H}_\nu(W \mid H, U, X) = \mathsf{H}_\nu(W \mid H, X). \tag{20}$$

The difference between the two conditional entropies is the lost information:

$$\delta = \mathsf{H}_\nu(W \mid U, X) - \mathsf{H}_\nu(W \mid H, U, X) \tag{21}$$

$$= \mathsf{H}_\nu(W \mid U, X) - \mathsf{H}_\nu(W \mid H, X). \tag{22}$$

Substitution yields Equations (2) and (3). □

The risk gap compares the net effects of information loss and predictive mismatch. Both depend on the final memory; neither belongs exclusively to one component of a factorized interface. The split along $H \to Z \to U$ is derived in Appendix B.1.

A.3 The conditional prediction target of distillation

The decomposition gives a condition for improvement, rather than an optimization procedure. To analyze one predictive objective, fix the memory map and let s_U range over unrestricted conditional predictors based on (U, X), rather than only predictions realizable by the frozen target. Define $\mathcal{R}(s) = \mathbb{E}_\nu[-\log s_U(W)]$ and the forward-KL objective

$$\mathcal{K}(s) = \mathbb{E}_\nu \, \mathrm{KL}(r_H \| s_U), \qquad \bar{r}_U = \mathbb{E}_\nu[r_H \mid U, X]. \tag{23}$$

Histories sharing (U, X) must use the same student prediction, so their teacher distributions enter the optimum through this conditional average.

Lemma 2 (Conditional projection of the teacher)**.** *For any predictor depending only on (U, X),*

$$\mathcal{K}(s) = \mathcal{K}(\bar{r}) + \mathbb{E}_\nu \, \mathrm{KL}(\bar{r}_U \| s_U). \tag{24}$$

Thus $\bar{r}_U$ minimizes the population distillation objective over unrestricted conditional predictors.

Proof of Lemma 2. Suppress explicit (U, X) arguments for brevity. Expanding around the conditional average by adding and subtracting $\log \bar{r}(v)$ gives

$$\mathcal{K}(s) = \mathbb{E} \sum_v r_H(v) \log \frac{r_H(v)}{\bar{r}(v)} + \mathbb{E} \sum_v r_H(v) \log \frac{\bar{r}(v)}{s(v)} \tag{25}$$

$$= \mathcal{K}(\bar{r}) + \mathbb{E}_{U,X} \sum_v \mathbb{E}[r_H(v) \mid U, X] \log \frac{\bar{r}(v)}{s(v)} \tag{26}$$

$$= \mathcal{K}(\bar{r}) + \mathbb{E}_{U,X} \mathrm{KL}(\bar{r} \| s). \tag{27}$$

Because $\bar{r}$ and s are measurable functions of (U, X), the inner teacher probability in the second line can be replaced by its conditional expectation. KL nonnegativity then establishes optimality, with equality only when $s = \bar{r}$ almost surely on the conditional supports. □

The frozen target and its parameterized memory interface may not realize this unrestricted optimum. Updating the memory also changes $\bar{r}_U$ itself. Whether the conditional target improves on Raw, and whether the interface realizes that target, are separate questions.

A.4 When conditional averaging improves on Raw

The following result states when conditional averaging cannot worsen raw predictive log-loss and when the inequality is strict.

Theorem 3 (Improvement under answer-sufficient compression)**.** *Suppose the memory is answer-sufficient, i.e., $W \perp H \mid U, X$. Then the conditional-average predictor $\bar{r}_U$ satisfies*

$$\mathcal{R}(\bar{r}) = \mathbb{E}_\nu[-\log \bar{r}_U(W)] \le \mathcal{R}_{\mathrm{R}}. \tag{28}$$

The inequality is strict if, on a positive-measure set of (U, X), there exists a token v with $\pi_U(v) > 0$ whose teacher probability $r_H(v \mid X)$ is nonconstant under $\nu(H \mid U, X)$.

Under answer sufficiency, histories sharing (U, X) have the same true conditional token law. Convexity then turns variation in their teacher predictions into an averaging gain. To quantify it, define, for a general law ν,

$$\mathcal{J} = \mathbb{E}_{U,X} \sum_{v \in \mathcal{V}} \pi_U(v) \left\{ \mathbb{E}[-\log r_H(v \mid X) \mid U, X] + \log \bar{r}_U(v \mid X) \right\}. \tag{29}$$

The quantity $\mathcal{J}$ is the answer-weighted conditional Jensen gap. Tokens with $\pi_U(v) = 0$ contribute zero, and convexity makes each positive-weight term nonnegative whenever the expectations exist. Under answer sufficiency, this gain equals $\mathcal{R}_{\mathrm{R}} - \mathcal{R}(\bar{r})$. It also admits the variance lower bound

$$\mathcal{J} \ge \frac{1}{2} \mathbb{E}_{U,X} \sum_{v} \pi_U(v) \operatorname{Var}(r_H(v \mid X) \mid U, X). \tag{30}$$

Proof of Theorem 3 and the variance bound. Answer sufficiency gives $\pi_H = \pi_U$ almost surely. Both risks therefore use the same conditional token weights:

$$\mathcal{R}_{\mathrm{R}} = \mathbb{E}_{U,X} \sum_{v} \pi_U(v) \mathbb{E}[-\log r_H(v \mid X) \mid U, X], \tag{31}$$

$$\mathcal{R}(\bar{r}) = \mathbb{E}_{U,X} \sum_{v} \pi_U(v) [-\log \bar{r}_U(v \mid X)]. \tag{32}$$

Subtracting these expressions yields $\mathcal{J}$. For each conditional state and token, Jensen's inequality for $-\log$ gives

$$\mathbb{E}[-\log r_H(v \mid X) \mid U, X] \ge -\log \mathbb{E}[r_H(v \mid X) \mid U, X]. \tag{33}$$

Nonnegativity follows. Strict convexity makes the inequality strict under the theorem's stated variation condition.

For the variance bound, let $R = r_H(v \mid X) \in (0, 1]$ under a fixed conditional law and $m = \mathbb{E}R$. Since $f(x) = -\log x$ satisfies $f''(x) = x^{-2} \ge 1$ on this interval,

$$f(R) \ge f(m) + f'(m)(R - m) + \tfrac{1}{2}(R - m)^2. \tag{34}$$

Expectation removes the linear term. Weighting by $\pi_U(v)$, summing over v, and averaging over (U, X) proves equation 30. Integrability of the losses suffices for this step; a uniform probability floor is unnecessary. □

A.5 Approximate sufficiency and realization error

Two departures from the ideal result matter for a learned memory: compression can discard answer-relevant information, and the frozen target may not realize $\bar{r}_U$ from the adapted input. The next bound separates these two effects from the averaging gain under an explicit probability-floor condition.

Theorem 4 (Information loss and conditional-average realization error)**.** *Let $\kappa = \mathbb{E}_\nu \mathrm{KL}(\bar{r}_U \| s_U)$. Suppose that $r_H(v \mid X)$ and $s_U(v \mid X)$ are at least $\varepsilon > 0$ for every token and almost every relevant state. Then*

$$\boxed{\mathcal{R}_{\mathrm{R}} - \mathcal{R}_{\mathrm{L}} \ge \mathcal{J} - \log(1/\varepsilon)\sqrt{\delta/2} - \frac{1}{\varepsilon}\sqrt{\kappa/2}.} \tag{35}$$

Hence a strictly positive right-hand side is sufficient for improvement over Raw.

Here δ measures information loss and κ measures departure from the conditional teacher average. They describe different obstacles to realizing the ideal gain. The probability floor applies to every token, so $\varepsilon \le 1/|\mathcal{V}|$; the resulting constants can make the bound uninformative for a large vocabulary. We therefore use the theorem as a sufficient condition for lower predictive log-loss, without claiming that a trained checkpoint satisfies a useful numerical bound.

Proof. Adjoin U to ν and form a comparison law that makes the answer token conditionally independent of history given the memory and state:

$$\nu_0(dh, du, dx, w) = \nu(dh, du, dx)\pi_U(w). \tag{36}$$

This construction preserves both the (H, U, X) and (U, X, W) marginals of ν. Its KL distance from the original law is

$$\mathrm{KL}(\nu\|\nu_0) = \mathbb{E}_\nu \log \frac{\pi_H(W)}{\pi_U(W)} = \delta. \tag{37}$$

Conditional sufficiency holds under ν_0, and hence

$$\mathbb{E}_{\nu_0}[-\log r_H(W \mid X)] - \mathcal{R}(\bar{r}) = \mathcal{J}. \tag{38}$$

Because $\mathcal{R}(\bar{r})$ depends only on (U, X, W), it is unchanged by this replacement of the law.

Let $b = \log(1/\varepsilon)$. Since the raw negative log-probability lies in $[0, b]$, bounded-loss stability and the total-variation–KL inequality imply

$$|\mathbb{E}_\nu[-\log r_H(W \mid X)] - \mathbb{E}_{\nu_0}[-\log r_H(W \mid X)]| \le b\,\mathrm{TV}(\nu, \nu_0) \tag{39}$$

$$\le b\sqrt{\delta/2}. \tag{40}$$

Consequently,

$$\mathcal{R}_{\mathrm{R}} - \mathcal{R}(\bar{r}) \ge \mathcal{J} - b\sqrt{\delta/2}. \tag{41}$$

For the realization error, both s and $\bar{r}$ are at least ε. At each conditional state,

$$\left|\sum_v \pi_U(v) \log \frac{\bar{r}(v)}{s(v)}\right| \le \max_v |\log \bar{r}(v) - \log s(v)| \tag{42}$$

$$\le \frac{1}{\varepsilon} \max_v |\bar{r}(v) - s(v)| \tag{43}$$

$$\le \frac{1}{\varepsilon}\mathrm{TV}(\bar{r}, s). \tag{44}$$

Each coordinate difference between two probability vectors is at most their total variation, giving the last inequality. Average over conditional states, apply the total-variation–KL inequality in the direction $\mathrm{KL}(\bar{r}\|s)$, and use concavity of the square root to obtain

$$|\mathcal{R}_{\mathrm{L}} - \mathcal{R}(\bar{r})| \le \frac{1}{\varepsilon}\mathbb{E}\sqrt{\mathrm{KL}(\bar{r}\|s)/2} \le \frac{1}{\varepsilon}\sqrt{\kappa/2}. \tag{45}$$

Subtracting this upper bound from equation 41 proves the theorem. □

For completeness, the probability inequality used above is $\mathrm{TV}(P, Q) \le \sqrt{\mathrm{KL}(P\|Q)/2}$ for natural logarithms. Map the measures to the indicator of a set attaining their total variation (or use an approximating sequence of sets). The log-sum inequality gives a lower bound by the resulting Bernoulli KL. For a fixed Bernoulli parameter q, its KL as a function of p has value and first derivative zero at $p = q$, and second derivative $1/[p(1-p)] \ge 4$. Thus the Bernoulli KL is at least $2(p-q)^2$. Boundary cases follow by continuity or an infinite KL. The bounded-loss inequality follows by integrating event-probability differences for a function in $[0, b]$.

An exact separation of averaging gain and error. Whenever the expectations exist, define

$$\mathcal{B} = \mathbb{E}_{\nu_0}[-\log r_H(W \mid X)] - \mathcal{R}_{\mathrm{R}}, \tag{46}$$
$$\mathcal{A} = \mathcal{R}_{\mathrm{L}} - \mathcal{R}(\bar{r}). \tag{47}$$

Then

$$\boxed{\mathcal{R}_{\mathrm{R}} - \mathcal{R}_{\mathrm{L}} = \mathcal{J} - \mathcal{B} - \mathcal{A}.} \tag{48}$$

The signed term $\mathcal{B}$ measures the effect of removing residual answer dependence on history, while $\mathcal{A}$ measures the change in predictive log-loss from realizing s_U instead of $\bar{r}_U$. Neither is itself a mutual information or a KL divergence, and either can be negative. Theorem 4 controls their magnitudes using δ and κ under its probability floor. Without a suitable support or tail condition, small distillation KL alone need not control predictive log-loss: a student can assign very small probabilities to labels that retain nonnegligible true probability. These results concern ideal conditional predictors and their realization error. Appendix B turns to the properties of the memory interface itself.

B Residual Memory Adaptation

The preceding results characterize the final memory U. LatentWeave constructs it through $Z = C_\phi(H)$ and $U = A_\psi(Z)$. This appendix first separates correction of a fixed memory from joint refinement of its construction and representation. It then examines local predictive improvement with the shared low-rank residual adapter and a sufficient condition for adaptation to preserve information.

B.1 Correcting the representation of a fixed memory

Hold the compressor fixed and compare the frozen target with the identity and adapted interfaces. Let $s_Z = p_\theta(\cdot \mid Z, X)$ and $\pi_Z = \nu(W = \cdot \mid Z, X)$, and define $\mathcal{D}_Z = \mathbb{E}\,\mathrm{KL}(\pi_Z \| s_Z)$. Write $\mathcal{R}(C, A)$ for the risk of the composed interface. Applying the cross-entropy decomposition to Z and $U = A(Z)$ gives

$$\begin{aligned}\mathcal{R}(C, \mathrm{Id}) &= \mathsf{H}_\nu(W \mid Z, X) + \mathcal{D}_Z,\\ \mathcal{R}(C, A) &= \mathsf{H}_\nu(W \mid U, X) + \mathcal{D}_{\mathrm{L}}.\end{aligned} \tag{49}$$

Since U is determined by Z, $\mathsf{H}_\nu(W \mid U, X) - \mathsf{H}_\nu(W \mid Z, X) = I_\nu(W; Z \mid U, X)$. Consequently,

$$\mathcal{R}(C, \mathrm{Id}) - \mathcal{R}(C, A) = \mathcal{D}_Z - \mathcal{D}_{\mathrm{L}} - I_\nu(W; Z \mid U, X). \tag{50}$$

At fixed Z, adaptation improves predictive risk when its reduction in mismatch exceeds the information loss it introduces. For an adapter family $\mathcal{A}$ containing the identity map,

$$\mathcal{G}_{\mathcal{A}}(C, \theta) = \mathcal{R}(C, \mathrm{Id}) - \inf_{A \in \mathcal{A}} \mathcal{R}(C, A) \geq 0. \tag{51}$$

The identity is admissible, so this gap measures nonnegative available improvement within the adapter family. It describes the family rather than a guarantee that training attains the infimum or improves risk.

Joint memory construction and residual adaptation. The decomposition in Equation equation 4 separates the information lost at each stage. Inserting $\mathsf{H}_\nu(W \mid Z, X)$ into Equation equation 22 gives

$$\begin{aligned}\delta &= \big[\mathsf{H}_\nu(W \mid Z, X) - \mathsf{H}_\nu(W \mid H, X)\big]\\ &\quad + \big[\mathsf{H}_\nu(W \mid U, X) - \mathsf{H}_\nu(W \mid Z, X)\big]\\ &= I_\nu(W; H \mid Z, X) + I_\nu(W; Z \mid U, X).\end{aligned} \tag{52}$$

The deterministic maps make both conditional mutual information terms nonnegative. With C_ϕ fixed, adaptation cannot change $I_\nu(W; H \mid Z, X)$ or recover distinctions discarded before Z. Updating C_ϕ lets predictive feedback revise the intermediate memory as well as its adapted representation. Both components can affect total information loss and final predictive mismatch $\mathcal{D}_{\mathrm{L}}$. This added flexibility motivates joint training without guaranteeing an improvement over adapter-only training.

B.2 Local predictive improvement through shared low-rank adaptation

Consider the shared residual adapter

$$A(z) = z + \gamma P \,\mathrm{SiLU}(Dz), \qquad D \in \mathbb{R}^{k\times d}, \quad P \in \mathbb{R}^{d\times k}, \quad \gamma \in [0,1]. \tag{53}$$

Shared parameters give each slot a correction determined by its content. To connect this architecture to predictive risk, we examine a local update from the identity interface.

Proposition 5 (Local predictive improvement through shared adaptation)**.** *Fix the compressor, D, and γ. Set $\varphi_i = \gamma\,\mathrm{SiLU}(Dz_i)$ and $u_i(P) = z_i + P\varphi_i$. Let $F(P)$ be the population answer-position log-loss of the frozen reader on $U(P)$. Assume differentiability, interchange of derivative and expectation, and L-Lipschitz continuity of ∇F on a neighborhood containing the update segment. Take $L > 0$. At the identity setting $P = 0$, define*

$$G = \mathbb{E}_\nu \sum_i g_i \varphi_i^\top, \qquad g_i = \nabla_{u_i}[-\log p_\theta(W \mid U, X)]|_{U=Z}\,. \tag{54}$$

Then, for $0 < \eta < 2/L$ within that neighborhood,

$$F(-\eta G) \le F(0) - \eta(1 - L\eta/2)\|G\|_F^2. \tag{55}$$

In particular, this update outperforms Raw whenever $\eta(1 - L\eta/2)\|G\|_F^2 > F(0) - \mathcal{R}_{\mathrm{R}}$.

The shared matrix G aggregates predictive gradients across slots and examples. If $G \neq 0$, the update strictly reduces risk relative to the identity interface. Improvement over Raw requires the additional inequality in the proposition. This local population-risk result fixes the compressor, down-projection, and gate and does not establish convergence of the training procedure.

Proof. Holding D, γ, and Z fixed makes φ_i independent of P. For one example and perturbation dP,

$$d[-\log p_\theta(W \mid U(P), X)] = \sum_i g_i^\top (dP)\varphi_i = \left\langle \sum_i g_i \varphi_i^\top, dP \right\rangle_F \tag{56}$$

at $P = 0$. Exchanging expectation and differentiation gives $\nabla F(0) = G$. Apply smoothness along the update segment:

$$F(P) \le F(0) + \langle G, P\rangle_F + \frac{L}{2}\|P\|_F^2. \tag{57}$$

Set $P = -\eta G$ to obtain equation 55. Comparing the resulting upper bound with $\mathcal{R}_{\mathrm{R}}$ gives the sufficient condition for outperforming Raw. □

Rank of the representation correction. For row-stacked memory $Z \in \mathbb{R}^{M\times d}$,

$$\Delta Z = \gamma\,\mathrm{SiLU}(ZD^\top)P^\top, \qquad \mathrm{rank}(\Delta Z) \le k. \tag{58}$$

The columns of P span a correction space of dimension at most k. The content-dependent coefficients φ_i produce different corrections across slots and examples within that shared space. The same bottleneck gives

$$J_A(z) - I = \gamma P\,\mathrm{diag}(\mathrm{SiLU}'(Dz))D, \tag{59}$$

whose rank is at most k. These bounds concern the representation correction and its Jacobian. Nonlinear target computation can produce KV changes of higher rank, and the bounds do not imply that predictive mismatch itself has low-rank structure.

B.3 Invertibility and information preservation

An invertible adapter changes the representation while preserving the information in Z. In this case, Equation equation 50 attributes any risk improvement entirely to reduced predictive mismatch. Contractive residual maps provide a sufficient condition for invertibility (Behrmann et al., 2019).

Proposition 6 (An information-preserving adapter)**.** *Let $L_{\mathrm{SiLU}} = \sup_x |\,\mathrm{SiLU}'(x)| < \infty$. If*

$$\rho = \gamma\|P\|_2\|D\|_2 L_{\mathrm{SiLU}} < 1, \tag{60}$$

then A is a bijection on $\mathbb{R}^d$ with a Lipschitz inverse. Applied slotwise with unchanged metadata, it satisfies

$$I_\nu(W; Z \mid U, X) = 0. \tag{61}$$

Proof. The residual $f(z) = \gamma P \,\mathrm{SiLU}(Dz)$ is ρ-Lipschitz, so

$$(1-\rho)\|u-v\| \le \|A(u)-A(v)\| \le (1+\rho)\|u-v\|. \tag{62}$$

Thus A is injective with inverse-Lipschitz constant at most $(1-\rho)^{-1}$. For each y, the contraction $z \mapsto y - f(z)$ has a unique fixed point, proving surjectivity. Slotwise inversion recovers Z from U, so they determine the same conditional law of W given X. □

If $G \neq 0$, a sufficiently small step in Proposition 5 also satisfies Equation equation 60. It therefore reduces predictive risk relative to the identity without additional information loss. Equation equation 50 then identifies the improvement as a reduction in predictive mismatch. This exact-arithmetic existence result does not establish that LatentWeave's trained adapter satisfies the sufficient norm condition.

C Training Signals for Residual Memory Adaptation

LatentWeave trains the residual memory interface in two stages. Stage I combines KV and attention guidance with auxiliary predictive supervision and regularization to initialize the adapter. Stage II jointly refines the compressor and adapter through the frozen target. We examine how the internal feature signals act on the adapted memory, then characterize the conditional target of the predictive objective.

C.1 Stage I: KV guidance through detached target features

Stage I updates the adapter while keeping the compressor, target, and auxiliary KV mapper fixed. Its feature targets are recomputed from the current adapted memory and detached during differentiation. These dynamic targets differ from the initial-memory references held fixed for auxiliary regularization. Let $\mathcal{R}(\psi)$ denote predictive risk at a fixed compressor, and consider a representative squared feature discrepancy with

$$a_\psi = P_\omega(A_\psi(Z), Q), \qquad t_\psi = F_\theta(A_\psi(Z)), \tag{63}$$

where the auxiliary mapper P_ω and target feature map F_θ process the same adapted memory. Write $J_a = \partial a_\psi/\partial\psi$ and $J_t = \partial t_\psi/\partial\psi$. Holding the target feature value constant during differentiation gives

$$g_{\mathrm{KV}} = J_a^\top (a_\psi - t_\psi). \tag{64}$$

The ordinary derivative retaining both input dependencies is

$$\nabla_\psi \tfrac{1}{2}\|a_\psi - t_\psi\|^2 = (J_a - J_t)^\top (a_\psi - t_\psi). \tag{65}$$

Although the target parameters are frozen, its features depend on the adapted input, so J_t need not vanish. Detaching the recomputed targets therefore produces a semi-gradient of the two-path discrepancy. For the normalized KV loss, detaching the target also fixes the normalization during each gradient calculation.

Suppose $\nabla\mathcal{R}$ is L_ψ-Lipschitz, with $L_\psi > 0$, on a neighborhood containing a proposed update segment. For any auxiliary direction g_{I}, smoothness gives

$$\mathcal{R}(\psi - \eta g_{\mathrm{I}}) \le \mathcal{R}(\psi) - \eta\langle\nabla\mathcal{R}, g_{\mathrm{I}}\rangle + \frac{L_\psi \eta^2}{2}\|g_{\mathrm{I}}\|^2. \tag{66}$$

Hence strict predictive descent is guaranteed if

$$\langle\nabla\mathcal{R}, g_{\mathrm{I}}\rangle > 0, \qquad 0 < \eta < \frac{2\langle\nabla\mathcal{R}, g_{\mathrm{I}}\rangle}{L_\psi\|g_{\mathrm{I}}\|^2}, \tag{67}$$

and the segment remains in that neighborhood. The bound follows by making the last two terms in equation 66 strictly negative. This condition depends on alignment with the predictive gradient, which is not established by a smaller feature discrepancy alone. KV supervision supplies initialization guidance, while its predictive effect is determined by the resulting memory interface.

C.2 THE SCOPE OF HISTORICAL ATTENTION GUIDANCE

The attention component of Stage I constrains how the auxiliary and target paths distribute attention within historical memory. Its scope can be seen directly from the normalization. For historical logits s_i, let

$$B_i = \frac{e^{s_i}}{\sum_{j\in\mathrm{hist}} e^{s_j}}, \qquad S_{\mathrm{hist}} = \sum_{j\in\mathrm{hist}} e^{s_j}. \tag{68}$$

Adding the same constant c to every historical logit preserves B. If S_{other} is the unnormalized mass of the other keys, the total attention mass allocated to history changes from

$$\frac{S_{\mathrm{hist}}}{S_{\mathrm{hist}} + S_{\mathrm{other}}} \quad \text{to} \quad \frac{e^c S_{\mathrm{hist}}}{e^c S_{\mathrm{hist}} + S_{\mathrm{other}}}. \tag{69}$$

Agreement over historical slots therefore constrains relative access within memory, while leaving its competition with question and generated-token keys undetermined. The attention output also depends on values, and vocabulary predictions require further target computation. Historical attention guidance supplies a signal for organizing access to memory. The predictive losses used during training act directly on the resulting output distributions.

C.3 STAGE II: COMBINING TEACHER DISTRIBUTIONS WITH ANSWER SUPERVISION

Stage II combines answer supervision with distillation from the raw target and a fixed reference teacher. We extend the conditional projection result to these predictive terms. Fix the memory map U and teacher functions r_H and b_H, where b_H denotes the reference teacher distribution at the same prediction state. Let $\lambda_r, \lambda_b, \lambda_y \geq 0$ with $\Lambda = \lambda_r + \lambda_b + \lambda_y > 0$. Terms with zero coefficients are omitted. Consider

$$\mathcal{L}_{\mathrm{pred}}(s) = \mathbb{E}_\nu \left[\lambda_r \,\mathrm{KL}(r_H \| s_U) + \lambda_b \,\mathrm{KL}(b_H \| s_U) - \lambda_y \log s_U(W)\right]. \tag{70}$$

With $\bar{b}_U = \mathbb{E}[b_H \mid U, X]$, define

$$m_U = \frac{\lambda_r \bar{r}_U + \lambda_b \bar{b}_U + \lambda_y \pi_U}{\Lambda}. \tag{71}$$

Proposition 7 (Mixed-supervision population target)**.** *For every conditional predictor s with finite objective,*

$$\mathcal{L}_{\mathrm{pred}}(s) = \mathcal{L}_{\mathrm{pred}}(m) + \Lambda\, \mathbb{E}_\nu \,\mathrm{KL}(m_U \| s_U). \tag{72}$$

Moreover,

$$\mathbb{E}\,\mathrm{KL}(\pi_U \| m_U) \leq \frac{\lambda_r \mathbb{E}\,\mathrm{KL}(\pi_U \| \bar{r}_U) + \lambda_b \mathbb{E}\,\mathrm{KL}(\pi_U \| \bar{b}_U)}{\Lambda}. \tag{73}$$

Proof. Condition on (U, X). Terms depending on s are

$$\begin{aligned} &-\sum_v \left[\lambda_r \bar{r}_U(v) + \lambda_b \bar{b}_U(v) + \lambda_y \pi_U(v)\right] \log s_U(v) \\ &\qquad = -\Lambda \sum_v m_U(v) \log s_U(v). \end{aligned} \tag{74}$$

The cross-entropy identity then gives Equation equation 72. Convexity of $\mathrm{KL}(\pi_U \| \cdot)$ gives

$$\mathrm{KL}(\pi_U \| m_U) \leq \frac{\lambda_r \,\mathrm{KL}(\pi_U \| \bar{r}_U) + \lambda_b \,\mathrm{KL}(\pi_U \| \bar{b}_U) + \lambda_y \,\mathrm{KL}(\pi_U \| \pi_U)}{\Lambda}. \tag{75}$$

The last term is zero. Taking expectations proves Equation equation 73. □

Answer labels contribute the true conditional law π_U through population averaging, so the learner need not have explicit access to that law. The two distillation terms contribute teacher distributions averaged over the same (U, X). This identifies the conditional target of the predictive part of Stage II and explains how answer supervision complements teacher imitation.

The proposition characterizes the predictive terms over unrestricted conditional predictors at a fixed memory map. In LatentWeave, joint updates change that map and its conditional mixture target, while the memory anchor further constrains the parameterized interface. The mixture therefore describes the predictive supervision target, not the optimum of the complete training objective.

D Predictive Log-Loss and Generated-Answer Quality

Section 3 compares memory interfaces using example-averaged reference-token log-loss. We now extend the decomposition to sequence prediction and relate it to decision risk. These results distinguish the probability assigned to reference answers from the quality of answers selected by a decoder.

D.1 Sequence risks and generated-prefix distributions

To compare complete sequences, fix a finite horizon and a common absorbing termination rule, including EOS or forced termination at the horizon. Raw and compressed-input prediction then define normalized distributions on the same finite sequence space. For an answer Y in that space, the risk without length normalization is

$$\mathcal{R}_{\mathrm{seq}}(s) = \mathbb{E}_{\mu}[-\log s(Y \mid U, Q)] = \mathbb{E}_{\mu} \sum_{t=1}^{T(Y)} -\log s(Y_t \mid U, Q, Y_{<t}). \tag{76}$$

The proof of Theorem 1 then gives

$$\mathcal{R}_{\mathrm{R},seq} - \mathcal{R}_{\mathrm{L},seq} = \mathcal{D}_{\mathrm{R},seq} - \mathcal{D}_{\mathrm{L},seq} - I_{\mu}(Y; H \mid U, Q), \tag{77}$$

Here the mismatch terms compare true and predicted *sequence* distributions. The information term obeys the corresponding token chain rule under the common termination convention. This sequence identity uses its own weighting. Replacing variable $T(Y)$ with its mean in Equation equation 12 does not recover it.

Likewise, the chain rule for teacher–student sequence KL is

$$\mathrm{KL}(r(\cdot \mid H, Q) \| s(\cdot \mid U, Q)) = \mathbb{E}_{Y \sim r(\cdot \mid H, Q)} \sum_{t} \mathrm{KL}\big(r_t(\cdot \mid H, Q, Y_{<t}) \| s_t(\cdot \mid U, Q, Y_{<t})\big). \tag{78}$$

This expectation uses teacher-generated prefixes, which generally differ from the reference prefixes in the token-risk law ν. Furthermore, multiplying the tokenwise conditional averages in Lemma 2 need not produce the conditional average of the teacher's full-sequence probability. The reference-prefix analysis and student rollouts consequently involve different prefix distributions. Relating their risks requires controlling that distribution change.

D.2 Predictive log loss and generated-answer correctness

A lower reference-token log-loss indicates better probabilistic prediction under the specified weighting, but correctness depends on the answer selected by the decoder. For a specified bounded answer loss $\ell(a, Y, Q)$, define the Bayes risks for decision rules with access to (H, Q) and (U, Q) as

$$\mathcal{E}^*_H = \inf_{a(H,Q)} \mathbb{E}\ell(a(H, Q), Y, Q), \qquad \mathcal{E}^*_U = \inf_{b(U,Q)} \mathbb{E}\ell(b(U, Q), Y, Q). \tag{79}$$

Every U-based rule can be simulated from H, so $\mathcal{E}^*_U \geq \mathcal{E}^*_H$. Let $\epsilon_{\mathrm{R}} = \mathcal{E}_{\mathrm{R}} - \mathcal{E}^*_H$ and $\epsilon_{\mathrm{L}} = \mathcal{E}_{\mathrm{L}} - \mathcal{E}^*_U$ for the actual decoders. Adding and subtracting the Bayes risks gives the exact identity

$$\boxed{\mathcal{E}_{\mathrm{R}} - \mathcal{E}_{\mathrm{L}} = \epsilon_{\mathrm{R}} - \epsilon_{\mathrm{L}} - (\mathcal{E}^*_U - \mathcal{E}^*_H).} \tag{80}$$

This is the decision-level counterpart of the predictive trade-off: improvement requires the reduction in decoder excess risk to outweigh the increase in Bayes decision risk caused by the restricted input. Reference-answer log-loss measurements alone do not identify these terms and therefore do not establish the accuracy ordering of two decoders.

For a finite-label decision problem on (H, X, W), zero–one loss and a student MAP decision permit a more explicit bound. In the remainder of this subsection, the $\mathcal{E}$ quantities refer to this finite-label problem, with decisions based on (H, X) or (U, X), rather than to the general complete-answer loss above. The Bayes errors satisfy

$$0 \leq \mathcal{E}^*_U - \mathcal{E}^*_H = \mathbb{E}[\max_{v} \pi_H(v) - \max_{v} \pi_U(v)] \tag{81}$$

$$\leq \mathbb{E}\mathrm{TV}(\pi_H, \pi_U) \leq \sqrt{\delta/2}. \tag{82}$$

The equality uses the tower property to express both Bayes errors over the same history marginal. For $a^* = \arg\max_v \pi_U(v)$ and $a_s = \arg\max_v s_U(v)$,

$$\pi_U(a^*) - \pi_U(a_s) \le |\pi_U(a^*) - s_U(a^*)| + |\pi_U(a_s) - s_U(a_s)| \tag{83}$$

$$\le \|\pi_U - s_U\|_1. \tag{84}$$

Averaging and applying the total-variation–KL inequality gives

$$\epsilon_{\mathrm{L}} \le \sqrt{2\mathcal{D}_{\mathrm{L}}}, \qquad \mathcal{E}_{\mathrm{L}} \le \mathcal{E}_H^* + \sqrt{\delta/2} + \sqrt{2\mathcal{D}_{\mathrm{L}}}. \tag{85}$$

The bound applies to the stated finite-label MAP decision. Tokenwise greedy generation generally differs from complete-answer MAP decoding, so its accuracy cannot be inferred by applying this bound to each token without further argument. Accordingly, LatentWeave's reference-answer NLL results provide predictive log-loss evidence, while generated-answer accuracy must be evaluated directly under the chosen decoding protocol.

E Implementation Details

This appendix details memory construction, the auxiliary losses used for KV-guided adapter initialization, and the reference branches used for joint refinement. The numerical configuration below describes the Qwen3-8B implementation. Reader-dependent dimensions and settings are not assumed to be shared by the separately trained readers.

E.1 Training data and target models

Training uses the original UltraChat conversations (Ding et al., 2023), the corpus used by LatentPress (Zhou & Sang, 2026), without LLM-generated augmentation or model-based verification. The original dialogue turns are used directly as training supervision. Evaluation uses the original LongMemEval question–answer examples and their evidence-bearing sessions, following the protocol in Section 5.1.

We train separate compressor–adapter interfaces for Qwen2.5-14B-Instruct, Qwen3-8B, Qwen3.5-4B, and Qwen3.8-27B. Each interface operates in its target model's embedding space, with the target parameters frozen throughout training.

E.2 Memory construction and residual adaptation

The compressor uses two copied target-model encoder layers. For a token embedding e_i and its encoder representation h_i, its fusion head computes

$$x_i = \alpha_i W_{\mathrm{f}} h_i + (1 - \alpha_i) e_i, \qquad \alpha_i = \sigma(w_{\mathrm{g}}^\top h_i + b_{\mathrm{g}}). \tag{86}$$

The fusion projection and gate belong to the compressor and are distinct from the shared residual adapter applied after pooling. Attention pooling uses four independent key projections and learned queries. Within a non-overlapping group G of r fused vectors, it produces

$$z_G = \frac{1}{4} \sum_{j=1}^{4} \sum_{i \in G} a_{j,i} x_i, \qquad a_{j,i} = \frac{\exp(q_j^\top K_j(x_i)/\sqrt{d})}{\sum_{v \in G} \exp(q_j^\top K_j(x_v)/\sqrt{d})}. \tag{87}$$

The four pooled vectors are averaged, so each group emits one memory slot. An incomplete final group is zero-padded to r entries before pooling. The recorded Stage-I initialization uses uniform factor-four pooling. In the role-segmented Stage-II and inference path, each segment is pooled independently. Factor-one segments instead supply their original token embeddings. The shared residual adapter acts on the complete history-memory prefix, including both pooled memory vectors and original embeddings from uncompressed segments.

For Qwen3-8B, the shared adapter has $d = 4096$ and bottleneck rank $k = 128$. Both linear maps are bias-free, and the gate is a learned scalar, giving $2dk+1 = 1{,}048{,}577$ parameters. Stage II initializes the compressor and adapter from the Stage-I checkpoint. The adapter and the compressor's fusion and pooling heads compute in FP32 before supplying BF16 memory embeddings to the frozen target.

E.3 KV-GUIDED INITIALIZATION AND AUXILIARY LOSSES

Let $U_0 = C_{\phi_0}(H)$ denote the unadapted memory from the fixed compressor and $U = A_\psi(C_{\phi_0}(H))$ the current adapted memory. Stage I compares three prediction paths. The auxiliary path supplies mapped historical KV states to the frozen target. The soft path uses standard target prefill on U, while the fixed-reference path uses standard target prefill on U_0. Their answer-position distributions are denoted by $p_{\mathrm{aux},t}$, $p_{\mathrm{soft},t}$, and $p_{\mathrm{ref},t}$, respectively.

The soft path provides feature and response targets that are recomputed from the current U and detached. The fixed-reference path instead uses unchanged memory U_0 to regularize the soft path. Corresponding branches receive the same question and answer-prefix trajectory. Fixing the reference memory and model parameters does not make its predictions independent of that trajectory.

Feature alignment. Auxiliary queries are obtained from the frozen target while it processes the answer prefix with mapped historical KV states. Teacher queries come from the target's standard prefill path on U. Both are captured at the positions predicting answer tokens, then undergo the target's query normalization and rotary transformation. Query alignment uses mean cosine dissimilarity. For historical-attention alignment, KV heads are repeated to match the target's query-head assignment, and the softmax is taken only over historical memory slots. Attention-output alignment compares the outputs of the target's self-attention modules at the same prediction positions, again using mean cosine dissimilarity. These query and attention losses use zero-based layers $\{0, 8, 16, 24, 35\}$. The KV loss averages over all target layers.

Auxiliary objective. In addition to the KV and historical-attention losses in Equation 9, the auxiliary objective combines feature alignment, predictive supervision, and reference regularization:

$$\begin{aligned}\mathcal{L}_{\mathrm{aux}} &= 0.02\mathcal{L}_{\mathrm{Q}} + 0.10\mathcal{L}_{\mathrm{AO}} + \mathcal{L}_{\mathrm{resp}} + \mathcal{L}_{\mathrm{ref}} + 10\,\mathrm{NMSE}(U, U_0) \\ &\quad + \eta\big(0.02\mathcal{L}_{\mathrm{CE,aux}} + 0.05\mathcal{L}_{\mathrm{first,aux}} + 0.02\mathcal{L}_{\mathrm{CE,soft}}\big),\end{aligned} \tag{88}$$

$$\mathcal{L}_{\mathrm{resp}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{KL}\big(\mathrm{sg}[p_{\mathrm{soft},t}] \| p_{\mathrm{aux},t}\big), \tag{89}$$

$$\mathcal{L}_{\mathrm{ref}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{KL}\big(\mathrm{sg}[p_{\mathrm{ref},t}] \| p_{\mathrm{soft},t}\big). \tag{90}$$

Here $\mathcal{L}_{\mathrm{Q}}$ and $\mathcal{L}_{\mathrm{AO}}$ average $1 - \cos(\cdot, \cdot)$ over corresponding vectors and selected layers. Response distillation matches the auxiliary path to the detached soft path. Reference KL and the memory penalty constrain deviations from the fixed-reference predictions and U_0. The quantity T counts compared answer-prediction positions on the current trajectory. On ground-truth-prefix steps,

$$\mathcal{L}_{\mathrm{CE},b} = -\frac{1}{T}\sum_{t=1}^{T} \log p_{b,t}(y_t), \qquad \mathcal{L}_{\mathrm{first,aux}} = -\log p_{\mathrm{aux},1}(y_1), \quad b \in \{\mathrm{aux}, \mathrm{soft}\}. \tag{91}$$

Prefix sampling and optimization. The Stage-I run alternates ground-truth and on-policy prefixes, with an on-policy step every two updates. On-policy prefixes are generated by the auxiliary mapped-KV path and detached. The switch η is one on ground-truth-prefix steps and zero on on-policy steps, disabling all three ground-truth CE terms on the latter. The other loss terms remain active and compare the corresponding trajectory positions. Both KV and latent NMSE use a denominator floor of 10^{-4}. Together with the main-text coefficients 0.10 for KV and 0.02 for historical attention, Equation 88 specifies all ten loss weights.

Optimization and numerical precision. For Qwen3-8B, Stage I uses AdamW with an adapter learning rate of 2×10^{-5}, zero weight decay, and gradient-norm clipping at 1.0. Only the shared residual adapter is optimized. The compressor, auxiliary KV mapper, and target model retain their weights, while their computations remain differentiable with respect to the adapted memory where required by the objective. Teacher features and reference predictions are detached. The adapter and the compressor's fusion and pooling heads compute in FP32, and memory embeddings are cast to BF16 before entering the frozen target.

E.4 AUXILIARY KV MAPPER

The Qwen3-8B auxiliary mapper is restored from previously trained KV-reconstruction weights. Its base construction inherits a rank-32 layerwise projector and a bottleneck residual of width 128. Its causal context decoder projects memory to width 512 and uses four attention heads. A group spans four target layers: a causal block produces the first state and three causal transition blocks produce the remaining states. Nine such groups provide states for the target's 36 layers, with layer-specific projections to eight KV heads of dimension 128. The question input to the mapper is the mean of frozen question-token embeddings, computed without gradient. This question dependence is confined to the training auxiliary path. The deployed shared adapter takes only the individual memory vector as its input. The mapper is an inherited checkpoint dependency whose pretraining precedes both refinement stages.

Pretrained dependencies. Adapter initialization starts from a pretrained compressor and an auxiliary mapper restored from KV-reconstruction training. These inherited weights precede Stage I. The compressor provides the initial memory, while the frozen mapper supplies the differentiable auxiliary path for KV-guided adaptation. Stage II inherits the compressor and adapter from Stage I. Stage-local update counts therefore exclude the earlier training used to obtain the compressor and mapper.

E.5 JOINT REFINEMENT AND REFERENCE SNAPSHOTS

Stage II jointly optimizes the compressor's two encoder layers, fusion projection and gate, pooling projections and queries, and shared residual adapter. The target model remains frozen, and the auxiliary mapper is not used. Gradients propagate through the target's soft-token prediction path to the compressor and adapter.

The objective in Equation 10 uses weights of 0.05 for answer cross-entropy, 1.0 for distillation from the uncompressed target, 0.25 for reference-distribution KL, and 1.0 for the normalized memory anchor. Stage II uses AdamW for both the compressor and the adapter. It runs for 768 updates with a batch size of 1 and gradient accumulation of 1, so each example triggers an optimizer step. The compressor uses a learning rate of 2×10^{-7} and weight decay of 1×10^{-4}, while the adapter uses a learning rate of 1×10^{-5} and zero weight decay. Both learning rates remain constant, without a scheduler or warmup. Gradients from the compressor and adapter parameters are jointly clipped to a global norm of 1.0. Training rows are shuffled independently for each run. Teacher and reference predictions are detached. The anchor uses the denominator floor of 10^{-4} specified in Equation 11.

The reference compressor is restored from the inherited Stage-I checkpoint. The reference adapter is copied from the adapter loaded at the beginning of the current training invocation. For a continuation run, the reference compressor therefore retains its Stage-I weights, while the reference adapter takes the resumed adapter weights. Both reference modules remain fixed within that invocation and determine the reference predictions and memory anchor.