DUAL-PATHWAY LATENT MEMORY INJECTION FOR FROZEN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Frozen large language models (LLMs) are usually augmented with external evidence by prepending the supporting passage to the prompt. This in-context injection is simple and effective, but prefill cost grows with evidence length, and the evidence competes with the generated history for attention. Recent latent injection methods instead write evidence into the KV cache, which integrates it more directly and at lower cost, yet these methods have not matched the accuracy of in-context injection. We explain this gap through the mechanics of factual recall: transformer LLMs rely on both MLP-encoded factual associations and attention-based attribute extraction, whereas KV injection acts only on attention, so under compression it preserves broad semantics but loses the fine-grained details needed for verbalization. We propose **LAMP** (**L**atent **A**ugmented **M**emory with **P**arallel Pathways), which compresses evidence into a fixed-size latent memory and injects it into a frozen LLM through two complementary pathways: a KV-cache pathway that broadcasts compressed semantics across layers, and a residual pathway in which the LLM's hidden states query token-level evidence features to recover missing details. On eight fictional, real-world, counterfactual, and long-tail question-answering benchmarks, LAMP outperforms in-context injection overall and also scores above an in-context baseline fine-tuned on the same data. Because each memory holds one passage, longer evidence is handled by composing memories, which keeps accuracy stable up to 256K tokens at constant memory per query. LAMP also supports mid-generation updates, adding new evidence without re-processing the preceding context and correcting earlier answers on the fly.

1 INTRODUCTION

Large language models (LLMs) store substantial factual knowledge in their parameters, but that knowledge is frozen at training time. One dominant way to keep LLMs current is retrieval-augmented generation (RAG) (Guu et al., 2020; Lewis et al., 2020), in which relevant evidence is first retrieved and prepended to the prompt, and the LLM then reads it through self-attention. Such in-context injection is simple and effective, but it is not an ideal memory mechanism. Long evidence is expensive to prefill and to keep in the KV cache (Vaswani et al., 2017), and relevant evidence can be under-attended when it is buried inside a long prompt or competes with the generated history (Liu et al., 2024). Adding more context can even lower performance (Jin et al., 2025; Li et al., 2025; Hsieh et al., 2024). The goal is thus to deliver the relevant evidence completely while avoiding these issues.

A direct way to pursue this goal is to inject evidence into the LLM's computation without changing its weights or extending its prompt. Larimar (Das et al., 2024) projects memory into the KV cache, MemoryLLM (Wang et al., 2024) maintains a persistent latent memory pool, KBLaM (Wang et al., 2025) encodes knowledge-base entries as continuous KV pairs, and Cartridges (Eyuboglu et al., 2026) trains a compact KV cache for each corpus. These methods show that latent injection is feasible and can be much cheaper, but their reported accuracy does not establish a clear advantage over in-context injection. KBLaM explicitly targets performance "comparable to in-context learning", and Cartridges reports matching in-context learning rather than exceeding it. This is puzzling. If latent injection avoids competition in text space and should be a more structured way to deliver evidence, why does placing evidence directly in the KV cache not clearly outperform pasting it into the prompt?

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

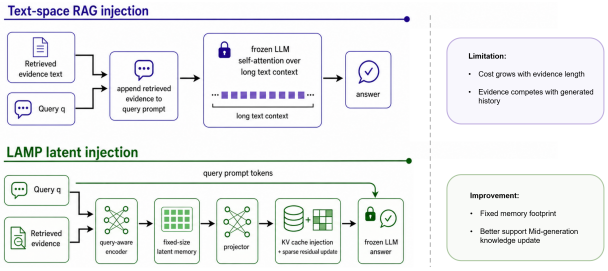

Figure 1: Conceptual illustration of text-space RAG injection and the proposed LAMP. In the standard RAG, retrieved evidence is appended to the query prompt. LAMP keeps the query as normal prompt tokens, but it encodes the retrieved evidence into a fixed-size latent memory for the KV cache and into token-level features read by sparse residual updates.

We believe this puzzle comes from how Transformer LLMs recall facts. They store factual associations in multilayer perceptron (MLP) sub-layers that act as key-value memories (Geva et al., 2021; Meng et al., 2022), and recall proceeds in two stages: MLP sub-layers first enrich the residual stream with subject-related attributes, and upper attention heads then query this enriched representation to extract the attribute being predicted (Geva et al., 2023). Latent KV injection touches only part of this process. The injected evidence enters through self-attention, and its compressed content suffers the same dilution observed in long-context attention and KV-cache compression (Liu et al., 2024; Zhong et al., 2025). Broad semantics can survive, but fine-grained details such as names and dates are easily lost. As a result, KV injection can steer the model toward the correct semantic region without reliably supplying the exact details needed to verbalize the answer.

This explanation motivates us to pair KV injection with a second pathway that targets the upper-layer residual stream, where attribute extraction takes place. Rather than broadcast evidence there, we mirror the LLM's own recall mechanism: at each generation step, the LLM's hidden state queries a small cross-attention adapter that holds the evidence, and the retrieved features are written back into the residual stream. We refer to the resulting framework as **LAMP** (**L**atent **A**ugmented **M**emory with **P**arallel Pathways). A query-aware encoder compresses a supporting passage into a fixed number of latent tokens, and a dual-pathway projector delivers them to a frozen LLM. One pathway tells the model *where to head*, and the other tells it which details to use (Figure 1). Each LAMP memory holds one passage, and longer evidence is handled by composing memories through a retrieval pipeline (Section 3.5) rather than by enlarging one memory.

Since LAMP is trained and the frozen LLM is not, outperforming in-context injection with the frozen LLM does not establish much on its own. We therefore also compare against the strongest in-context baseline we could train on the same data, which we call In-context+LoRA: the same LLM fine-tuned with LoRA (Hu et al., 2022) on LAMP's 146K training triples, which still receives every passage uncompressed in its prompt. This comparison deliberately favours the baseline.

We evaluate on eight fictional, real-world, counterfactual, and long-tail QA benchmarks. On Qwen3-VL-8B (Bai et al., 2025), LAMP reaches 91.8% accuracy, compared with 84.7% for in-context injection. It also outperforms In-context+LoRA, with an F1 score of 93.2% against 91.7%, and it stays ahead on average when distractors are added to the evidence. When a correction arrives in the middle of generation, LAMP applies it within the same turn in 85.2% of cases, while In-context+LoRA never does. LAMP also outperforms the prior latent-memory methods that can run under our protocol, and controls show that the gain comes from the injection site rather than from extra capacity or supervision. Finally, when a reranker composes passage memories, LAMP keeps its accuracy between 86% and 91% from 16K to 256K tokens of evidence at constant memory per query, with higher F1 than the same retrieved passage given as text, whereas reading the whole corpus in context no longer fits on a 32 GB GPU beyond 32K tokens.

Our contributions are (i) a hypothesis for why latent KV injection has not outperformed in-context injection, together with causal tests of its predictions; (ii) LAMP, which delivers a passage to a frozen LLM through both a fixed-size KV memory and its upper-layer residual stream; (iii) an evaluation under matched supervision in which LAMP outperforms an in-context baseline fine-tuned on the same data; and (iv) a composition of passage memories that stays accurate up to 256K tokens.

2 RELATED WORK

Retrieval augmentation and long context. REALM (Guu et al., 2020), RAG (Lewis et al., 2020), and HippoRAG (Gutierrez et al., 2024) place retrieved evidence in the input as text, whereas FiD (Izacard & Grave, 2021) and Atlas (Izacard et al., 2023) encode each passage separately for the decoder to cross-attend to, and RETRO (Borgeaud et al., 2022) cross-attends to encoded neighbours retrieved for each chunk of the sequence. Long-context models instead enlarge the context window to 512K tokens for ProLong (Gao et al., 2025) and 1M tokens for Qwen2.5-1M (Yang et al., 2025), while Recursive Language Models (Zhang et al., 2026) let the model call itself over snippets of a long prompt. All of these approaches still represent evidence token by token. A larger window determines how much evidence fits but not its cost, since the KV cache still grows with the evidence, and accuracy degrades well within the window (Hsieh et al., 2024).

Latent memory for pretrained LLMs. External-memory LMs attach stored representations to generation (Khandelwal et al., 2020; Wu et al., 2022; Wang et al., 2023). Closer to our setting, KBLaM (Wang et al., 2025) encodes knowledge-base triples as KV pairs read through rectangular attention, Larimar (Das et al., 2024) couples an episodic memory with a decoder trained to read from it, and MemoryLLM (Wang et al., 2024) embeds a pool of memory tokens in every layer and trains the model to write new text into it. Cartridges (Eyuboglu et al., 2026) trains a small KV cache offline for each corpus with a context-distillation objective, and MAC (Tack et al., 2024) amortizes each document into a compact modulation and aggregates them, conditioned on the question, to adapt a frozen LM. All of them deliver evidence through a single interface read by attention, whereas LAMP adds a second pathway where attribute extraction occurs (Appendix Table 7).

Compression, parallel encoding, and parametric memory. AutoCompressors (Chevalier et al., 2023), Gist Tokens (Mu et al., 2023), ICAE (Ge et al., 2024), and xRAG (Cheng et al., 2024) compress context into soft tokens. CEPE (Yen et al., 2024) and a retrofitted RETRO (Borgeaud et al., 2022) let a frozen decoder cross-attend to separately encoded evidence, training only the encoder and the cross-attention layers. LAMP's residual pathway is a frozen-model cross-attention reader of this kind, confined to upper layers and gated; alone, it stays below in-context injection (Table 1), and LAMP's gain comes from pairing it with a compressed KV memory. Text-to-LoRA (Charakorn et al., 2025) generates adapters from task descriptions, and PERK (Chen et al., 2026) encodes context into adapters by test-time learning, so both need new weights for new evidence. Recurrent Memory Transformer (Bulatov et al., 2022) and Memory Layers (Berges et al., 2025) modify the architecture. Model editors (Mitchell et al., 2022a;b; Meng et al., 2022; 2023) change the model, whereas LAMP's edits are removable.

Mechanistic interpretability of factual recall. Interpretability work shows that MLP sub-layers act as key-value memories whose editing can install new facts (Geva et al., 2021; Meng et al., 2022; 2023), and that upper attention heads extract the required attribute from the enriched subject representation (Geva et al., 2023). LAMP's residual pathway is designed to mirror this second stage.

3 METHOD

The proposed LAMP augments a frozen LLM with external evidence through three components (Figure 2). Given a supporting passage p and a query q, the encoder compresses p into M query-conditioned latent memory tokens while retaining token-level evidence features. The projector converts the latent tokens into per-layer KV pairs and prepares the memory states read by sparse residual adapters. The injector then appends the KV pairs to the frozen LLM's cache and applies gated residual updates at selected upper layers. The parameters of the base LLM are never modified.

3.1 QUERY-AWARE MEMORY ENCODER

Backbone. We use a frozen bidirectional encoder, ModernBERT-large (Warner et al., 2025) (395M parameters), to produce per-token features $\mathbf{H} \in \mathbb{R}^{L \times d}$ with $d = 1024$ from a passage p of L tokens. Bidirectionality matters because each token representation can use both left and right context; for example, the representation of *France* can encode that the answer later in the sentence is *Paris*.

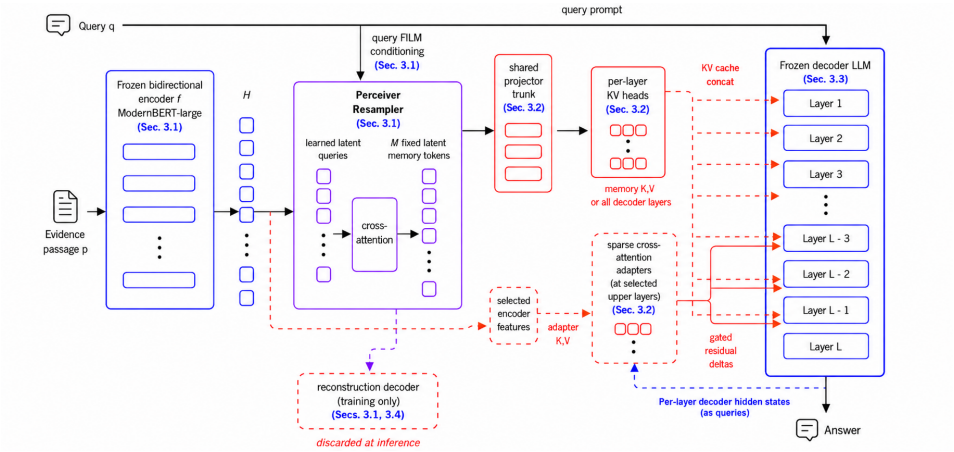

Figure 2: The proposed LAMP architecture. A frozen encoder extracts token-level evidence features, and a Perceiver resampler compresses them into fixed-size latent tokens. The KV branch produces memory key-value pairs for every LLM layer, and the residual branch uses sparse cross-attention adapters at selected upper layers, where LLM hidden states query the encoder features.

Perceiver resampler. The compressor is a Perceiver Resampler (Jaegle et al., 2021; Alayrac et al., 2022) with M learned query vectors that distill the backbone features into M latent tokens of dimension d through alternating cross-attention, self-attention, and feed-forward layers. To make the compression query-aware, the Perceiver queries are modulated by a pooled encoding of q via FiLM (Perez et al., 2018), and a lightweight block selector scores evidence blocks so that answer-bearing blocks are kept before compression.

Reconstruction decoder (training only). A 2-layer cross-attention decoder reconstructs the backbone features from the M latents under an MSE loss. This objective encourages the Perceiver to keep fine-grained details that compression would otherwise lose. The decoder is discarded at inference.

3.2 Dual-pathway projector

The projector has two pathways aligned with the two-stage view of factual recall: a KV pathway that operates through self-attention, and a residual pathway that targets attribute extraction.

Shared KV trunk and per-layer KV heads (first pathway). A shared trunk, $\text{Linear}(d, d) \to \text{GELU} \to \text{LayerNorm}$, pre-processes the latent tokens and reduces parameter redundancy across layers. For each of the N LLM layers, a linear head then maps the trunk features to $K_i, V_i \in \mathbb{R}^{n_{\text{kv}} \times M \times d_{\text{head}}}$ via $\text{Linear}(d, 2n_{\text{kv}}d_{\text{head}})$. Zero initialization keeps the learned memory contribution inactive at the start of training, and a learned sigmoid scale controls the strength of each layer.

Sparse residual adapters (second pathway). At selected upper LLM layers, cross-attention adapters use the current hidden states as queries over memory states built from token-level encoder features, which preserves details that may be lost in the compressed latents. The resulting token-wise update is gated, magnitude-clamped, and added back to the hidden states. A zero-initialized output projection and a learned gate keep the adapter selective.

3.3 Evidence injection

At inference time, the injector concatenates the projected pairs with the existing cache of every layer, giving $K_i' = \text{concat}(K_i^{\text{existing}}, K_i^{\text{memory}})$ and $V_i' = \text{concat}(V_i^{\text{existing}}, V_i^{\text{memory}})$. At each selected residual layer ℓ, an adapter computes Δh_ℓ from the hidden state and the encoder features and applies $h_\ell' = h_\ell + \Delta h_\ell$. Because M is fixed, the KV cache grows by exactly M entries per layer regardless of passage length. The adapters read the memory's encoder features, which grow with the passage,

up to the encoder's 8,192-token window, and stay outside the LLM's self-attention. Because injection adds no tokens, a memory can be added mid-decoding without re-processing earlier tokens.

3.4 Training protocol

We train the resampler, projector, and adapters jointly through context distillation (Snell et al., 2023). A *teacher* forward pass feeds the passage text directly into the frozen LLM's context to produce a reference output distribution, and a *student* forward pass removes the passage and injects the encoded latent memory instead. The training objective is

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{CE}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{HSM}}\mathcal{L}_{\text{HSM}} + \lambda_{\text{nh}}\mathcal{L}_{\text{no-harm}}}_{\text{behavioural terms}} + \underbrace{\lambda_{\text{kp}}\mathcal{L}_{\text{known-preserve}} + \lambda_{\text{sel}}\mathcal{L}_{\text{selector}} + \lambda_{\text{probe}}\mathcal{L}_{\text{probe}} + \lambda_{E}\mathcal{L}_{E} + \lambda_{\text{comp}}\mathcal{L}_{\text{comp}}}_{\text{regularizers and auxiliary heads}}, \tag{1}$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy on answer tokens, $\mathcal{L}_{\text{KL}}$ is the top-K filtered KL divergence between the student and teacher distributions (K=16), $\mathcal{L}_{\text{recon}}$ is the reconstruction MSE, $\mathcal{L}_{\text{HSM}}$ matches final-layer hidden states by cosine distance, and $\mathcal{L}_{\text{no-harm}}$ penalizes injection that makes the LLM worse than it is without injection. The remaining five terms protect answers the base model already knows, supervise the block selector, train a confidence probe, limit the energy of the residual update, and prevent the residual pathway from overwriting a KV pathway that has already solved an example. All ten terms are defined in Appendix E. The KL weight and temperature are annealed during training. We train on 146K (passage, question, answer) triples from the eight benchmarks for 5 epochs with an effective batch size of 24. For Qwen3-VL-8B, 154M parameters (1.9% of the target) are trained.

3.5 Composing passage memories

Each LAMP memory encodes a single passage. When the evidence is a collection of passages, a retrieval pipeline encodes each passage into its own memory, retrieves candidate passages with an embedding model, reranks them with an off-the-shelf reranker from the Qwen3 Embedding series (Zhang et al., 2025), and injects the top k memories through both pathways, without changing any weights. The injected KV state is then $k \times M$ entries per layer, set by the budget rather than by the collection or the passage length.

We view the passage as the natural unit of a latent memory rather than as a workaround. A fixed-size memory cannot preserve an arbitrarily long text, so compressing a whole collection into M latents would lose fidelity as the collection grows, while per-passage encoding keeps the compression ratio bounded. Deciding which passage matters is a ranking problem that rerankers already handle well. A compressor that also had to select evidence would need to learn to reject passages, which a compressor trained only on relevant passages never learns (Section 4.5). Independent memories can also be added or removed one at a time, as mid-generation injection requires (Section 4.6). Other systems scale evidence in the same way: FiD encodes passages independently (Izacard & Grave, 2021), MAC keeps one modulation per document (Tack et al., 2024), separately trained Cartridges can be composed at inference (Eyuboglu et al., 2026), and Recursive Language Models decompose long prompts (Zhang et al., 2026). The frozen encoder also reads at most 8,192 tokens per pass (Warner et al., 2025), so a single-memory design would have to chunk longer inputs anyway.

4 Experiments

4.1 Training and evaluation settings

Benchmarks and targets. We train and evaluate on eight benchmarks covering fictional QA (RepLiQA (Monteiro et al., 2024)), real-world QA (QA-ZRE (Levy et al., 2017), TriviaQA (Joshi et al., 2017), NQ (Kwiatkowski et al., 2019), REBEL (Huguet Cabot & Navigli, 2021)), counterfactual editing (CounterFact (Meng et al., 2022), WikiBigEdit (Thede et al., 2025)), and long-tail recall (PopQA (Mallen et al., 2023)) (Appendix F). Their mean evidence lengths range from 13 to 184 tokens, so this suite measures how well a single relevant passage is delivered, and Section 4.5 turns to noisy and long evidence. All results use fixed held-out examples that are disjoint from the 146K

training triples. The target LLMs are Qwen3-VL-8B (Bai et al., 2025) (8.2B parameters, 36 layers, native 256K context) and Phi-4-Mini (Abouelenin et al., 2025) (3.8B parameters, 32 layers). Both stay frozen, and the same recipe trains on both without modification.

Baselines and controls. The *closed-book* baseline uses the frozen LLM without any evidence. *In-context injection* prepends the passage as text to the prompt of the frozen LLM. *In-context+LoRA* applies in-context injection after fine-tuning the same LLM with LoRA on the identical 146K triples, using LAMP's epochs and prompt template. The *input-layer injection* control keeps LAMP's encoder, resampler, $M{=}256$, data, schedule, and loss terms (except those tied to the missing pathways) and changes only the delivery: the latents are projected once into the input embedding space and prepended to the prompt as soft tokens. Supervision is comparable, as In-context+LoRA sees each answer span in full context. Because these controls share LAMP's data, schedule, and losses, any gap to LAMP reflects where evidence enters.

Metrics. *Accuracy*, the default metric in the per-benchmark tables, counts a prediction as correct when its normalized form contains the primary gold answer. *F1* is the token-level F1 score, maximized over all accepted answer aliases. Because accuracy only checks whether the gold answer appears in the output, it tends to reward verbose answers. For example, it scores the WikiBigEdit predictions of In-context+LoRA at 100.00%, while F1 gives them 83.74%. We therefore compare LAMP with In-context+LoRA on F1 for better fairness, and still report both metrics in Appendix J. Evaluation is deterministic (fixed splits, one checkpoint per configuration, greedy decoding) and paired.

4.2 Main results

Table 1 compares LAMP with both baselines. LAMP improves over the closed-book baseline by 61.5 points overall on Qwen3-VL-8B and by 76.2 points on Phi-4-Mini, and it improves over in-context injection by 7.1 and 9.7 points, respectively. The largest gain appears on CounterFact on Qwen3-VL-8B (100.0% vs. 54.1%), where direct injection overrides parametric knowledge more effectively than prompt text. To confirm that this result does not come from edits seen in training, we retrained LAMP with the entire CounterFact family removed, and it still reaches 72.97% on held-out CounterFact, above in-context injection with the passage in the prompt. In the ablation block, each pathway alone falls short of in-context injection (Section 4.4). With only 16 latents, LAMP matches in-context injection (84.8% vs. 84.7%), losing most on TriviaQA and NQ, whose passages are longest. On Qwen3-VL-8B, LAMP trails in-context injection on two benchmarks, QA-ZRE by 2.5 points and WikiBigEdit (80.9% vs. 91.5%). The WikiBigEdit gap is an artifact of the accuracy metric. Its gold answers are long and admit several correct phrasings, and under F1, LAMP (83.03%), in-context injection (82.98%), and In-context+LoRA (83.74%) lie within 0.76 points (Appendix J).

Prior latent-memory methods. MAC and KBLaM, the two prior methods that keep the target frozen, are trained on our 146K triples with Qwen3-VL-8B, with each passage converted into triples for KBLaM, which accepts only structured knowledge (Appendix G). LAMP outperforms both on every benchmark and overall (91.8% against 54.0% and 42.2%), and apart from MAC on CounterFact, both also trail in-context injection (Table 1). KBLaM is weakest where answers must be read from free text: on RepLiQA, whose fictional entities cannot come from parametric knowledge, it reaches only 8.3%, close to the closed-book 3.7%, and on TriviaQA it adds only 1.9 points over the closed-book model, consistent with its own observation that a fixed-length vector per triple may fail to reproduce exact names or numbers (Wang et al., 2025). Its largest gains come on already relational evidence, CounterFact and REBEL (+37.8 and +25.0 points). MAC, the stronger baseline, handles short counterfactual evidence well and exceeds in-context injection on CounterFact (75.7% vs. 54.1%), but it degrades on long passages such as RepLiQA and NQ, where a 24-token modulation must hold the whole passage, and trails our KV-only variant on six of eight benchmarks. Both patterns support our motivation that a single compressed interface keeps broad semantics but loses the details needed for verbalization. Larimar and MemoryLLM retrain the LLM itself, and Cartridges trains a KV cache per corpus offline, so none of them fits a frozen target with evidence that arrives per query (Appendix B).

Table 1: Benchmark accuracy (%). LAMP uses M=256 latents unless noted; KBLaM and MAC are trained on our data with the frozen Qwen3-VL-8B ([‡]triples extracted from passages). *Ctx.*: mean evidence tokens.

	Fictional	**Real-world**				**Counterfactual**		**Long-tail**	**All**
	RepLi QA	QA-ZRE	Trivia QA	NQ	REBEL	Counter Fact	WikiBig Edit	PopQA	
Ctx.	*100.2*	*40.6*	*183.6*	*122.9*	*33.8*	*13.6*	*17.9*	*13.2*	
Phi-4-Mini									
Closed-book	1.9	13.8	39.8	22.5	31.9	0.0	14.9	5.4	18.8
In-context	89.8	87.5	84.3	78.4	87.5	78.4	80.9	**100.0**	85.3
LAMP	**100.0**	**95.0**	**93.5**	**92.8**	**95.8**	**100.0**	**83.0**	**100.0**	**95.0**
Qwen3-VL-8B									
Closed-book	3.7	40.0	69.4	30.6	31.9	8.1	17.0	8.1	30.3
In-context	**88.9**	**86.3**	90.7	79.3	81.9	54.1	**91.5**	94.6	84.7
KBLaM[‡]	8.3	51.3	71.3	39.6	56.9	45.9	31.9	24.3	42.2
MAC	29.6	65.0	75.9	42.3	50.0	75.7	51.1	62.2	54.0
LAMP	**88.9**	83.8	**97.2**	**93.7**	**93.1**	**100.0**	80.9	**100.0**	**91.8**
Ablations on Qwen3-VL-8B									
KV pathway only	55.6	58.8	71.3	46.8	69.4	91.9	63.8	94.6	64.2
Residual pathway only	55.6	61.3	77.8	47.7	76.4	100.0	68.1	97.3	67.7
LAMP, M=16	82.4	81.3	86.1	79.3	91.7	89.2	80.9	100.0	84.8
LAMP, M=64	87.0	83.8	88.9	82.0	91.7	91.9	83.0	100.0	87.3
LAMP, 4-layer resampler	87.0	83.8	95.4	89.2	93.1	100.0	80.9	100.0	90.3

Table 2: Matched supervision on Qwen3-VL-8B (%). F1 is token-level F1 of the systems trained on the same triples; the last two columns give the rate at which a mid-generation update is applied.

		Correction applied	
System	**F1**	**Same turn**	**New turn**
In-context	–	2.5	24.6
In-context+LoRA	91.66	0.0	22.5
Input-layer injection	54.36	0.0	18.1
LAMP	**93.18**	**85.2**	**87.6**

Table 3: Mechanism controls on Qwen3-VL-8B (accuracy, %). The closed-book accuracy is 30.3.

Variant	**Acc.**
Full LAMP	**91.83**
KV disabled at inference	51.83
Residual disabled at inference	25.33
Trained without KV	67.67
Trained without residual	64.17
KV-only, widened to 154M	69.83
Input-layer injection	46.33

4.3 MATCHED SUPERVISION

Table 2 compares LAMP with the systems trained on the same 146K triples. LAMP reaches an F1 score of 93.18% against 91.66% for In-context+LoRA, even though In-context+LoRA sees every token that LAMP compresses. On this metric, compressing the evidence into a fixed-size memory therefore has a minimal cost to LAMP. The lead is modest in distribution but grows where In-context+LoRA was not trained. When distractors are added, LAMP is ahead in three of the four conditions (Section 4.5), and it applies a mid-generation correction within the same turn in 85.2% of cases, whereas In-context+LoRA never does (Section 4.6). Only the accuracy metric puts In-context+LoRA ahead, by 0.84 points, because it rewards verbose answers (Appendix J). Learned compression alone does not explain this result, since input-layer injection, identical to LAMP except for where the memory enters, reaches only 54.36% F1.

To test whether the trained systems answer from memorized training pairs, we gave each question an empty passage or another question's passage (a random derangement). LAMP falls to 15.83% and 7.33% accuracy, and In-context+LoRA to 13.50% and 7.00%, all below the closed-book accuracy of 30.3%, so both depend on the evidence to the point that a mismatched passage misleads them.

4.4 WHAT THE TWO PATHWAYS DO

Trained separately, each pathway falls below in-context injection (64.17% and 67.67%), yet together they reach 91.83% (Tables 1 and 3). We first rule out extra capacity and extra supervision as the source of this gain, and then ask how the two pathways divide the work.

Capacity and supervision. Widening the KV-only variant to the full model's 154M trainable parameters recovers only 5.66 of the 27.66 points between KV-only and LAMP, and input-layer injection, which has LAMP's supervision and memory size, reaches only 46.33%.

Knockouts. To see how the jointly trained model relies on each pathway, we disable one at inference time (Table 3). Removing a pathway hurts more than never training with it, by 15.84 points for the KV pathway and by 38.84 points for the residual pathway, whose removal drops accuracy below the closed-book level. The pathways are thus co-adapted, and the residual one carries more of the answer.

When each pathway is used. Knockouts remove a pathway throughout generation, so we next remove each one only while the question is read or only at steps that emit tokens. Removing the residual pathway at emission drops accuracy from 91.83% to 48.50%, against 7.2 points during reading; without the KV memory, it loses 17.0 points during reading and 12.3 points during emission. The residual pathway is thus consumed at emission, with its attention on the gold answer span 3.37$\times$ higher there, while the KV memory mainly shapes how the question is read.

What the KV memory carries. If the KV memory mainly shapes how the question is read, it should carry answer-relevant content, not the answer itself. With the residual pathway disabled, we give each question its own evidence or another question's. Where both answers are wrong, its own evidence brings them closer to the gold (token F1 0.089 vs. 0.051, cosine 0.735 vs. 0.701), yet another question's evidence almost never yields that question's answer (0.2%, against 17.8% for the full model; Appendix H). These results match the two-stage view of recall: the KV memory steers the reading of the question, and the residual pathway supplies the answer at emission.

4.5 LONG EVIDENCE: FROM ONE MEMORY TO A RETRIEVAL PIPELINE

The benchmarks above give each question one relevant passage. Real evidence is noisier and longer, and we study it at two levels: a single memory that holds the whole collection (the component), and the retrieval pipeline of Section 3.5 (the system). A held-out suite from TriviaQA, NQ, and RepLiQA pairs each question's passage with four distractors, drawn at random, mined with BM25 (Robertson & Zaragoza, 2009) from the same corpus, or taken from the passage's nearest neighbours, and a fourth condition extends the evidence to 8,192 tokens (Table 4).

The component. With all candidates concatenated into one 256-latent memory, the checkpoint of Table 1 drops to between 23.55% and 42.17%. The cause is the training data: every training triple had exactly one relevant passage, so the compressor learned to find the answer inside a passage but never to reject one. The limit is trainable: 1,000 steps of fine-tuning with BM25 hard negatives raise all four conditions by 7.3 to 11.3 points (a trend, not a converged result).

The system. The pipeline instead encodes each candidate as its own memory and injects all of them in reranked order, without changing weights. The in-context baselines read the same passages in the same order, so only the delivery differs. LAMP recovers to between 80.83% and 89.83%, level with in-context injection on average (86.2% against 86.8%) and ahead of In-context+LoRA (84.5%), which falls below in-context injection in three of the four conditions.

Scale. We then grow the evidence into shared corpora of 16K to 262,144 tokens (269 to 4,610 passages), up to the native context limit of Qwen3-VL-8B, with the supporting passage at a random position. Qwen3-Reranker-4B reranks the 48 passages nearest to the question in embedding space, and the pipeline injects the top one. As the matching baseline, the frozen LLM reads that same passage as text (Table 5). The reranker ranks the supporting passage first (R@1) for 95% to 100% of questions, and LAMP keeps its F1 between 88.5 and 93.0 (86% to 91% accuracy) across this 16$\times$ increase. Given the same passage, LAMP's F1 is 3.4 to 5.7 points higher at every size. The

Table 4: Distractors and long evidence (accuracy, %).

	LAMP			In-context	
Condition	**Single memory**	**+ Distractor FT**	**Pipeline**	**Frozen LLM**	**+LoRA**
4 random distractors	42.17	53.50	**89.83**	88.17	85.61
4 BM25-mined distractors	35.50	44.33	85.33	**86.67**	83.26
4 nearest-neighbour distractors	31.00	40.17	80.83	**86.00**	82.28
Evidence at 8,192 tokens	23.55	30.89	**88.99**	86.24	86.93

Table 5: Corpora of 16K–256K tokens (F1, %).

		LAMP	As text	
Corpus	**R@1**	**Top passage**	**Top passage**	**Whole corpus**
16K	100	**92.96**	87.24	78.60
32K	99	**92.53**	87.15	75.01
64K	99	**92.53**	87.15	out of memory
128K	98	**91.81**	86.27	out of memory
256K	95	**88.53**	85.15	out of memory

Table 6: Cost of one memory.

Evidence tokens	**1,024**	**8,192**
Time to first token (ms)		
In-context	112.5	821.6
LAMP	**45.1**	**58.9**
Added memory (MB)		
In-context	481	3,610
LAMP	**104**	**118**

smaller Qwen3-Reranker-0.6B gives the same pattern (Appendix N). Reading the whole corpus, as long-context models do, degrades from 16K to 32K, and runs out of memory on the 32 GB GPU from 64K onward; at 16K it needs 8.1 GB and 2.0 s to the first token, about 58 and 47 times LAMP's cost, close to that of the passage as text. One memory cannot reject noise, but with a reranker, passage memories read retrieved evidence as well as text up to 256K at constant memory.

Cost and limits of one memory. One memory's cost stays nearly flat as evidence grows: at 8,192 tokens, LAMP reaches the first token 13.9× faster than in-context injection with 30.6× less memory (Table 6, encoder included). At $M{=}256$, text is lighter below about 113 to 164 tokens (Appendix L); the budget M sets where accuracy and cost meet: at $M{=}64$, LAMP beats in-context injection (87.3% against 84.7%, Table 1) and uses less memory for every passage over 60 tokens (Figure 6).

4.6 Mid-generation injection

Evidence can also arrive mid-generation, when text must join the cached history but a LAMP memory is added without re-processing it. We let the LLM first generate an unrelated story, so the evidence competes with its own history, and then ask a question with the evidence as text or as a memory. After 32 to 100 story tokens, LAMP answers 5.5 to 8.0 points more accurately than in-context injection, and 3.0 and 1.0 points more after 200 and 400 (Appendix Q). Correction is harder: on CounterFact, the LLM first answers from the original evidence, so the old answer sits in its cache; we then supply a contradictory update and check the answer as generation continues (same turn) and when asked again (new turn). As text, the update loses: the LLM repeats the old answer in 92.1% of same-turn and 84.2% of new-turn answers, giving the new one in only 2.5% and 24.6%. Injected by LAMP, the new answer appears in 85.2% and 87.6%, and 45.4% of same-turn answers still mention the old one, so LAMP adds the correction instead of overwriting the cache (Appendix R). Neither LoRA fine-tuning nor input-layer injection corrects in the same turn (Table 2), so the injection site matters.

5 Conclusion and future work

LAMP delivers evidence to a frozen LLM through a KV pathway that shapes how the question is read and a residual pathway that supplies the answer's details at emission. It outperforms in-context injection and prior latent-memory methods, and with a reranker, its memories read retrieved evidence as well as text up to 256K tokens. In distribution, it leads a fine-tuned in-context baseline by 1.52 points in F1; its larger advantages are robustness to distractors, same-turn injection, and a bounded state per passage. A single memory needs a relevance gate such as our reranker, and compression reduces verbatim fidelity. Our evaluations are single-hop; bounded memory and same-turn injection make multi-hop a natural next step.

AI USE STATEMENT

In this work, we used generative AI tools for helping to implement methods of LAMP. We have not used generative AI tools for design or to provide feedback on research methodology or experiments; and generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, interpret results, are not applicable to this work.

Additionally, we used generative AI tools for create or edit some parts of the software code. The architectural design, system structure, and integration of these components were entirely planned and executed by us. We used AI tools for sourcing/searching for information and related work that we may have missed after our own manual literature search. We also used AI tools to proofread the research paper after we finished writing it, to improve readability and perform grammar checks. We have reviewed all AI-assisted work. The 2 authors verified and tested LLM-generated code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

LAMP makes a frozen LLM follow what an injected memory says, including counterfactual content that overrides its own knowledge (Table 1). The same property that makes it useful for knowledge updates also makes it a possible channel for misinformation when memory sources are untrusted, so deployments should authenticate the evidence they inject. Because the base weights are never modified, removing a memory restores the model's original behaviour exactly.

REPRODUCIBILITY STATEMENT

Section 3.4 and Appendix E give the full training objective, Appendix D describes the hardware and software environment, and Section 3.5 describes the retrieval pipeline. Section 4.1 specifies the data splits, baselines, controls, and scorers, and the anonymous artifact (Appendix A) contains the architecture, dataset loaders, configuration files, evaluation scripts, and unit tests.

REFERENCES

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-Mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs. *arXiv preprint arXiv:2503.01743*, 2025. doi: 10.48550/arXiv.2503.01743. URL https://arxiv.org/abs/2503.01743.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millicah, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. 2025. URL https://arxiv.org/abs/2511.21631.

Vincent-Pierre Berges, Barlas Oguz, Daniel Haziza, Wen-Tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3831–3842. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/berges25a.html.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7485–7514. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/charakorn25a.html.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qxDTe8fIyA.

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 *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, Aurelie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiri Navratil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10109–10126. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/das24a.html.

Sabri Eyuboglu, Ryan Saul Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Ruoyu Liu, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0k5w8O0SNg.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). 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. 7376–7399, Vienna, Austria, July 2025. Association for

Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.366. URL https://aclanthology.org/2025.acl-long.366/.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751/.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=hkujvAPVsg.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: Relation extraction by end-to-end language generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2370–2381, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.204. URL https://aclanthology.org/2021.findings-emnlp.204/.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.74. URL https://aclanthology.org/2021.eacl-main.74/.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023. URL http://jmlr.org/papers/v24/23-0037.html.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. Long-context LLMs meet RAG: Overcoming challenges for long inputs in RAG. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oU3tpaR8fm.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia (eds.), *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL https://aclanthology.org/K17-1034/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Kuan Li, Liwen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Shuai Wang, and Minhao Cheng. LaRA: Benchmarking retrieval-augmented generation and long-context LLMs – no silver bullet for LC or RAG routing. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36846–36867. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/li25dv.html.

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/.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.),

Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 17–23 Jul 2022b. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Joao Monteiro, Pierre-Andre Noel, Étienne Marcotte, Sai Rajeswar, Valentina Zantedeschi, David Vazquez, Nicolas Chapados, Christopher Pal, and Perouz Taslakian. RepliQA: A question-answering dataset for benchmarking LLMs on unseen reference content. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=4diKTLmg2y.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=2DtxPCL3T5.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: visual reasoning with a general conditioning layer. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Charlie Victor Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. 2023. URL https://openreview.net/forum?id=am22IukDiKf.

Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. Online adaptation of language models with a memory of amortized contexts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=RIfgKCknTu.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. WikiBigEdit: Understanding the limits of lifelong knowledge editing in LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59326–59354. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/thede25a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BryMFPQ4L6.

Xi Wang, Taketomo Isazawa, Liana Mikaelyan, and James Hensman. KBLam: Knowledge base augmented language model. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=aLsMzkTej9.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian Mcauley. MEMORYLLM: Towards self-updatable large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50453–50466. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wang24s.html.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. 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. 2526–2547, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.127. URL https://aclanthology.org/2025.acl-long.127/.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1M technical report. 2025. URL https://arxiv.org/abs/2501.15383.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Howard Yen, Tianyu Gao, and Danqi Chen. Long-context language modeling with parallel context encoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2588–2610, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.142. URL https://aclanthology.org/2024.acl-long.142/.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models. 2026. URL https://arxiv.org/abs/2512.24601.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. 2025. URL https://arxiv.org/abs/2506.05176.

Meizhi Zhong, Xikai Liu, Chen Zhang, Yikun Lei, Yan Gao, Yao Hu, Kehai Chen, and Min Zhang. ZigZagKV: Dynamic KV cache compression for long-context modeling based on layer uncertainty. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 8897–8907, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.596/.

A Anonymous review artifact

An anonymized source artifact is available at `https://anonymous.4open.science/r/lamp-memory-1F73`. The repository contains the LAMP architecture implementation, dataset loaders, configuration files, checkpoint-loading and evaluation scripts, diagnostics, and unit tests. During anonymous review, the artifact is intended for source inspection and evaluation-path verification; it does not distribute trained projector weights, generated dataset caches, archived experiment logs, or the exact full-scale training runner. These generated artifacts and the final reproduction runner will be released with the public version after review.

B Extended limitations

Scope. In distribution, LAMP leads In-context+LoRA by 1.52 points of F1 but trails it by 0.84 points in accuracy, so its in-distribution lead is modest and depends on the metric. Its larger advantages are robustness to distractors, same-turn injection, and a per-query state and cost set by the number of memories rather than by the size of the collection. All training and evaluation data are single-hop, so multi-hop reasoning, as in HotpotQA (Yang et al., 2018), remains untested. Two measured properties nevertheless make a multi-hop design practical. First, at a fixed memory per query regardless of corpus size, running one LLM instance per hop is affordable in settings where concurrent long-context inference would exhaust a 32 GB device. Second, and more importantly, in multi-hop reasoning the next piece of evidence is unknown until the previous hop is resolved, so each step must take in new evidence during reasoning. LAMP does this within the same turn in 85.2% of cases, while in-context injection almost never does. An orchestrator could therefore retrieve after each intermediate result and inject the new memory without restarting generation, which keeps the chain of hops explicit and inspectable in the orchestration layer. We offer this as a direction rather than as a result.

Component and system. A single memory does not reject irrelevant evidence unless it is trained to do so, and within a generation an injected memory dominates the model's parametric knowledge (Section 4.3). Injection should therefore be gated by relevance, as the reranker does in our retrieval pipeline. Compression also reduces verbatim fidelity (84.5% against 100% on 4,096-token needle retrieval; Appendix M), and at $M{=}256$, passages shorter than roughly 113 to 164 tokens are cheaper to keep as text than as a memory.

Mechanism, training, and comparison. Our tests establish that the pathways are asymmetric and co-adapted, that the residual pathway is consumed at emission while the KV memory mainly conditions the reading of the question, and that the KV memory carries question-dependent answer semantics without the answer itself (Appendix H); a feature-level description of the KV memory remains open. The projector must be retrained for each target (about 25 hours for Qwen3-VL-8B on one RTX 5090), although the recipe itself transfers across targets without modification. We stopped the long-context study at the target's native 256K window rather than compare against an in-context baseline with RoPE scaling. Finally, some prior latent-memory methods could not be evaluated under our protocol. Larimar trains its decoder jointly with its memory module, and MemoryLLM continually pre-trains its backbone with a pool of memory tokens in every layer, so neither can be attached to a frozen Qwen3-VL-8B without retraining the target itself; the released MemoryLLM models also require at least 16 tokens per injected context, more than the mean evidence length of CounterFact and PopQA. Cartridges trains a KV cache for each corpus offline through synthetic self-study conversations and gradient descent, so it requires test-time training for every new passage and cannot take in evidence that arrives with a query or during generation.

C Positioning against related families

Table 7 organizes the related families by where evidence enters the model, which is the axis on which our claims are made. Long-context LLMs and RAG add a per-query state that grows with the evidence, and text appended during generation must be read through the existing cache. Weight-space methods change the model for each task or context, so new evidence requires new weights; Text-to-LoRA, in particular, adapts to a natural-language task description rather than to retrieved evidence. Architectural memories are part of the backbone and are trained with it, so a LAMP memory could

Table 7: Where evidence enters the model. Per-query state is what one query adds at inference time; for LAMP, it is k passage memories, each with M KV entries per layer and encoder features bounded by the encoder's window.

Family (examples)	Evidence enters via	Target LLM	Per-query state
In-context (RAG, ProLong, Qwen2.5-1M, RLM)	prompt tokens	unchanged	grows with evidence
Cross-attention readers (FiD, RETRO, CEPE)	cross-attention to encoded evidence	co-trained/frozen	grows with evidence
Weight space (Text-to-LoRA, PERK)	adapter weights	adapted	fixed
Architectural (RMT, Memory Layers)	memory tokens or layers	co-trained	fixed
Latent memory (KBLaM, Larimar, MemoryLLM, Cartridges, MAC)	KV cache or soft tokens	frozen/co-trained	fixed
LAMP (ours)	KV cache + residual stream	frozen	$k \times M$ per layer + encoder features

be attached to such a backbone rather than compete with it. Cross-attention readers encode the evidence outside the decoder and let it cross-attend to the encoded tokens; RETRO's retrofitting and CEPE keep the decoder frozen, which makes them the closest prior art for LAMP's residual pathway. Latent-memory methods are the closest family for the KV pathway. Each delivers evidence through one interface read by attention with a fixed per-query state, and they differ mainly in how the memory is produced: a knowledge-base encoder for KBLaM, a co-trained memory module for Larimar and MemoryLLM, per-corpus training for Cartridges, and an amortized per-document encoder for MAC. LAMP keeps the target frozen, produces each memory with one query-conditioned encoder pass, and adds a second pathway into the upper-layer residual stream.

D Compute resources

The main Qwen3-VL-8B projector runs were trained on a single Ubuntu 24.04 workstation with an Intel Core i9-14900K CPU, 125GiB system RAM, and one NVIDIA GeForce RTX 5090 GPU. The software environment used Python 3.12, PyTorch 2.10 with CUDA 12.8, bfloat16 support, and 4-bit base-model loading via bitsandbytes. Each full Qwen3-VL-8B projector training run takes approximately 25 hours; full evaluation uses the same machine class. All latency and memory measurements in Section 4.5 were taken on one RTX 5090 with 32 GB of VRAM. CPU and Mac/MPS machines are suitable for lightweight source checks and unit tests, but not for full Qwen3-VL-8B evaluation.

E Training objective

This appendix details every term of Equation (1). Let z_{it}^S denote the student logits at example i and position t when the passage is removed and the latent memory is injected, z_{it}^T the aligned teacher logits when the same passage is provided as text, and z_{it}^0 the no-injection baseline logits. Let y_{it} be the target token, $a_{it} \in \{0,1\}$ the answer-token mask, and $m_{it} \in \{0,1\}$ the shared-span mask used for KL and hidden-state matching. Training minimizes Equation (1), restated here:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{HSM}}\mathcal{L}_{\text{HSM}} + \lambda_{\text{nh}}\mathcal{L}_{\text{no-harm}} + \lambda_{\text{kp}}\mathcal{L}_{\text{known-preserve}} + \lambda_{\text{sel}}\mathcal{L}_{\text{selector}} + \lambda_{\text{probe}}\mathcal{L}_{\text{probe}} + \lambda_E\mathcal{L}_E + \lambda_{\text{comp}}\mathcal{L}_{\text{comp}}.$$

The first five are the behavioural terms, and the remaining five are regularizers and auxiliary heads used for stability and diagnostics. The main text and this appendix describe the same objective with the same ten terms.

Answer cross-entropy. The answer-token loss is

$$\mathcal{L}_{\text{CE}} = \frac{1}{N_A}\sum_{i,t} a_{it}\, w_{it}\left[-\log \text{softmax}(z_{it}^S)_{y_{it}}\right], \quad N_A = \sum_{i,t} a_{it}. \tag{2}$$

We upweight the first answer token because it is usually the decision point where the model must switch from its parametric knowledge to the injected evidence. In our experiments, the first answer token has raw weight 5.0 and later answer tokens have raw weight $\sqrt{5.0}$; the answer weights are normalized per example so the total answer-token weight remains equal to the number of answer tokens. Optional length weights can further upweight medium or long memories, but the reported configuration sets them to 1.0.

Top-K skew KL distillation. For temperature τ, define $p^T_{\tau,it} = \mathrm{softmax}(z^T_{it}/\tau)$ and $p^S_{\tau,it} = \mathrm{softmax}(z^S_{it}/\tau)$. Let S_{it} be the K highest-probability teacher tokens, with $K{=}16$. Both distributions are renormalized on S_{it}:

$$\bar{p}^T_{\tau,it}(v) = \frac{p^T_{\tau,it}(v)}{\sum_{u\in S_{it}} p^T_{\tau,it}(u)}, \quad \bar{p}^S_{\tau,it}(v) = \frac{p^S_{\tau,it}(v)}{\sum_{u\in S_{it}} p^S_{\tau,it}(u)}. \tag{3}$$

The implemented KL uses a skewed denominator, $q_{\tau,it}(v) = \alpha\bar{p}^S_{\tau,it}(v) + (1-\alpha)\bar{p}^T_{\tau,it}(v)$, with $\alpha = 0.1$, giving

$$\mathcal{L}_{\mathrm{KL}} = \frac{\tau^2}{N_K}\sum_{i,t} m_{it}\, g_i\, w_{it}\, \frac{1}{\alpha}\sum_{v\in S_{it}} \bar{p}^T_{\tau,it}(v)\log\frac{\bar{p}^T_{\tau,it}(v)}{q_{\tau,it}(v)}. \tag{4}$$

Here $g_i = \mathbf{1}[\mathrm{CE}^T_i < \gamma_{\mathrm{KL}}]$ gates out examples where the teacher is itself unreliable, and $N_K = \sum_{i,t} m_{it}g_i$. The KL temperature anneals from 3.0 to 1.0, and λ_{KL} anneals from 0.3 to 0.65.

Reconstruction MSE. The reconstruction decoder maps the latent memory tokens back to the frozen encoder's token features. If H_{it} is the original ModernBERT feature and $\hat{H}_{it}$ is the reconstruction, then

$$\mathcal{L}_{\mathrm{recon}} = \frac{1}{D\sum_{i,t} r_{it}}\sum_{i,t} r_{it}\,\|\hat{H}_{it} - \mathrm{sg}(H_{it})\|_2^2, \tag{5}$$

where D is the feature dimension and $\mathrm{sg}(\cdot)$ stops gradients into the frozen encoder. The token weight r_{it} is 1 by default, 3 on answer spans, and 2 on question-anchor spans that appear in the evidence passage. Answer and anchor spans are located by string-matching each triple's answer and question terms inside its own passage. The auxiliary decoder is discarded at inference.

Hidden-state matching. For selected decoder layers $\mathcal{J}$, the student and teacher hidden states are aligned on the shared span. The reported setting uses the final Qwen3-VL layer only, $\mathcal{J} = \{35\}$. With hidden states $h^S_{\ell it}$ and $h^T_{\ell it}$,

$$\mathcal{L}_{\mathrm{HSM}} = \frac{1}{|\mathcal{J}|}\sum_{\ell\in\mathcal{J}} \frac{1}{\sum_{i,t} m_{it}}\sum_{i,t} m_{it}\left(1 - \frac{\langle h^S_{\ell it}, h^T_{\ell it}\rangle}{\|h^S_{\ell it}\|_2\|h^T_{\ell it}\|_2}\right). \tag{6}$$

Cosine distance is used instead of MSE so the loss matches direction without forcing hidden-state magnitudes to agree.

No-harm and known-preserve. For real-world and long-tail examples, an additional no-injection baseline forward pass is run. Let C^S_i and C^0_i be average answer-token CE under injection and no injection. The no-harm term applies only to examples where the baseline is poor enough that memory should help:

$$\mathcal{L}_{\text{no-harm}} = \frac{1}{|\mathcal{H}|}\sum_{i\in\mathcal{H}} \max(0, C^S_i - C^0_i), \quad \mathcal{H} = \{i : C^0_i \geq \gamma_{\mathrm{nh}}\}. \tag{7}$$

In our experiments, $\gamma_{\mathrm{nh}} = 2.0$. A complementary preservation branch protects examples the base model already knows. Let $c^{S,1}_i$ and $c^{0,1}_i$ be the first-answer-token CE values with and without injection. Then

$$\mathcal{L}_{\text{known-preserve}} = \frac{\sum_{i\in\mathcal{P}} u_i \max(0, c^{S,1}_i - c^{0,1}_i - \epsilon)}{\sum_{i\in\mathcal{P}} u_i}, \quad \mathcal{P} = \{i : C^0_i \leq \gamma_{\mathrm{kp}}\}. \tag{8}$$

The reported setting uses $\gamma_{\mathrm{kp}} = 2.0$, margin $\epsilon = 0.1$, and length weights $u_i \in \{1.0, 1.25, 4.0\}$ for short, medium, and long memories.

Selector supervision. The block selector predicts scores s_{ib} for evidence blocks. Soft targets t_{ib} mark answer-bearing blocks with weight 1.0 and question-anchor blocks with weight 0.7. The selector loss is

$$\begin{aligned}\mathcal{L}_{\text{selector}} = \mathrm{BCEWithLogits}(s, t) \\ + \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \mathrm{softplus}(1 - \max_{b \in A_i} s_{ib}) \\ + \frac{1}{2|\mathcal{Q}|} \sum_{i \in \mathcal{Q}} \mathrm{softplus}(0.5 - \max_{b \in Q_i} s_{ib}),\end{aligned} \tag{9}$$

where A_i and Q_i are the answer and anchor block sets. This term teaches the selector to retain the right evidence before compression.

Probe, energy, and residual complementarity. The confidence probe pools hidden states over question tokens and predicts whether injection helped. With probe logit o_i, injected CE C_i^S, and baseline CE C_i^0, the delta target is

$$t_i^{\text{probe}} = \sigma(C_i^0 - C_i^S), \quad \mathcal{L}_{\text{probe}} = \mathrm{BCEWithLogits}(o_i, t_i^{\text{probe}}). \tag{10}$$

For residual injection, if $\Delta_{\ell it}$ is the adapter delta and $h_{\ell it}$ is the pre-update hidden state, the energy penalty is the mean relative squared energy

$$\mathcal{L}_E = \frac{1}{|\mathcal{R}|} \sum_{\ell \in \mathcal{R}} \frac{\mathrm{mean}_{i,t,d}\, \Delta_{\ell itd}^2}{\mathrm{mean}_{i,t,d}\, \mathrm{sg}(h_{\ell itd})^2 + \epsilon}. \tag{11}$$

Finally, residual complementarity compares full injection against a KV-only pass on the same real-world/long-tail subset:

$$\mathcal{L}_{\text{comp}} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \max(0, C_i^{KV+R} - C_i^{KV}). \tag{12}$$

This prevents the residual pathway from improving by overwriting a KV path that already solved the example.

F BENCHMARKS

Table 8: Evaluation benchmarks. "Ctx." is the mean context length in tokens.

Category	Benchmark	Ctx.	Description
Fictional	RepLiQA	100.2	Fictional-entity QA isolating pure memory recall.
Real-world	QA-ZRE	40.6	Relation extraction from compact passages.
	TriviaQA	183.6	Factoid QA with supporting evidence passages.
	NQ	122.9	Wikipedia factoid QA; stresses passage compression.
	REBEL	33.8	Relation-centric QA from Wikipedia triples.
Counterfactual	CounterFact	13.6	Override of conflicting parametric knowledge.
	WikiBigEdit	17.9	Consistency after factual updates.
Long-tail	PopQA	13.2	Recall of obscure, low-popularity facts.

G PRIOR LATENT-MEMORY BASELINES

Both baselines are evaluated on the same held-out examples, prompt template, greedy decoding, and metrics as LAMP.

MAC. We follow the MAC design for 7B-scale targets: a T5 amortization network encodes each passage into a modulation of 24 tokens, an aggregation network conditioned on the question combines the modulations, and a P-tuning v2 prefix injects the result into every layer of the frozen Qwen3-VL-8B. The amortization and aggregation networks are trained on our 146K triples for 5 epochs with learning rate 10^{-5}.

KBLaM. KBLaM encodes knowledge as (name, property, value) triples. For CounterFact, WikiBigEdit, REBEL, QA-ZRE, and PopQA, the triples come directly from the dataset annotations; for RepLiQA, TriviaQA, and NQ, we extract them from each passage with Qwen3-VL-8B. The sentence encoder is ada-002, and the linear adapters and KB query heads are trained on the triple form of our 146K training examples with the frozen Qwen3-VL-8B for 5 epochs.

H WHAT THE KV MEMORY CARRIES AND WHEN EACH PATHWAY IS USED

All analyses use the checkpoint of Table 1, the same held-out questions, greedy decoding, and 4-bit loading of the target. The instrumentation that records and ablates the pathways is inert: generations are identical with it on and off, and the instrumented runs reproduce the intact model (91.83%) and the residual knockout (25.33%) exactly.

Phase ablation. We remove one pathway during one phase only. *Read* covers the prompt positions that emit nothing; *emit* covers the positions that emit a token, namely the last prompt position, which emits the first answer token, and every decoding step. The recorded attention and residual writes at the ablated positions are exactly zero.

Table 9: Accuracy (%) when a pathway is removed during one phase, Qwen3-VL-8B (intact: 91.83). All drops are significant under an exact McNemar test against the intact model ($p \leq 9\times10^{-9}$).

Pathway removed	During read	During emit	Whole pathway
Residual adapters	84.67 (−7.2)	48.50 (−43.3)	25.33 (−66.5)
KV memory	74.83 (−17.0)	79.50 (−12.3)	51.83 (−40.0)

Removing the residual pathway at emission costs six times more than removing it during reading, so its content is consumed when the answer is produced. The KV memory is used in both phases and more while the question is read. For both pathways, the two phase drops sum to less than the whole-pathway drop, so each phase partly compensates when the other is removed. Like the whole-pathway knockouts, these ablations act on a model trained with both pathways, and they measure where that model relies on each one.

Evidence crossover. A direct comparison between the KV memory and the closed-book model is confounded by answer form: under the same instruction, closed-book answers are full sentences (median 11 words), whereas the trained KV memory makes the model answer with short entities (median 2 words), so token F1 and embedding similarity mix content with format. We therefore compare the same KV-only configuration (the residual pathway disabled at inference) with its own evidence and with another question's evidence, assigned by a fixed random derangement, so that the answer format is identical and only the injected content differs.

Table 10: Evidence crossover with the residual pathway disabled, over the 434 questions answered wrongly in both conditions (similarities), and over all questions (accuracy on the donor's gold answer). Cosine similarity uses Qwen3-Embedding and MiniLM encoders.

	Own evidence	Donor evidence
Wrong answers vs. own gold, token F1	**0.089**	0.051
Wrong answers vs. own gold, cosine (Qwen3-Emb / MiniLM)	**0.735 / 0.442**	0.701 / 0.393
Answers vs. donor's gold, cosine (Qwen3-Emb / MiniLM)	0.557 / 0.184	0.554 / 0.186
Accuracy on the donor's gold answer	0.0	0.17
Full model, accuracy on the donor's gold answer	–	17.83

A question's own evidence moves wrong answers closer to its gold answer on all three measures ($p < 10^{-5}$, paired Wilcoxon test), while the similarity to the donor's gold answer does not change ($p > 0.18$). The KV memory alone therefore carries answer-relevant semantics that depend on the question, but it does not carry an answer that the model can extract: evidence about another question almost never produces that question's answer, whereas the full model produces it 17.83% of the time. Against the closed-book model, with measures that are robust to answer form, the KV-only answers

are also slightly closer to the gold answer (best-span cosine above the answer-type floor, $+0.019$ and $+0.028$, $p \leq 2\times10^{-4}$), an effect concentrated on the fictional RepLiQA questions, where parametric knowledge cannot help.

Attention timing. The residual adapters' cross-attention on the gold answer span is 3.37× higher at steps that emit answer tokens than at other steps (95% CI [3.24, 3.50]). The LLM's attention mass on the injected KV entries is lower at answer steps (0.735×), but an empty-passage memory receives 97% of the same mass with the same profile, so attention mass reflects where the injected block sits rather than whether its content is used. For this reason, the conclusions above rest on the causal ablations.

I TRAIN-TEST CONTAMINATION AUDIT

CounterFact is the benchmark on which a perfect score most naturally raises the question of train-test leakage, so we audited it directly. Checked against all 146K training triples, the CounterFact test edits show no full-triple overlap, no shared (subject, relation) pair, and no test subject anywhere in the training text. As a dataset-level contamination test, we also removed the entire CounterFact family from training and retrained LAMP from scratch. CounterFact answers deliberately contradict the frozen model's parametric knowledge, yet without a single CounterFact example in training, LAMP still reaches 72.97% on the held-out edits. For comparison, in-context injection with the passage in the prompt reaches 54.1%, and LAMP trained with the family included reaches 100.0%.

J ACCURACY AND F1

The paper reports two metrics. *Accuracy* counts a prediction as correct when its normalized form contains the primary gold answer, and *F1* is the token-level F1 score, maximized over all accepted answer aliases. Neither metric bounds the other. A verbose answer that contains the gold answer is counted as correct by accuracy but receives a low F1, while a short alias that differs from the primary gold answer receives a high F1 but is counted as incorrect by accuracy. Table 11 reports both metrics for the systems trained on the same 146K triples, and Table 12 reports them for WikiBigEdit.

Table 11: Systems trained on the same 146K triples, Qwen3-VL-8B, all held-out examples (%).

System	Accuracy	F1
In-context+LoRA	92.67	91.66
Input-layer injection	46.33	54.36
LAMP	91.83	93.18

Table 12: WikiBigEdit, Qwen3-VL-8B, under both metrics (%).

System	Accuracy	F1
In-context+LoRA	100.00	83.74
In-context	91.49	82.98
LAMP	80.85	83.03

The two metrics disagree in a consistent direction. Accuracy credits In-context+LoRA more than F1 does, and F1 credits LAMP more than accuracy does. This pattern is consistent with In-context+LoRA more often wrapping the gold answer in a longer response, which accuracy credits and F1 penalizes, and with LAMP more often giving an accepted alias that differs from the primary gold answer. In-context+LoRA leads by 0.84 points in accuracy, and LAMP leads by 1.52 points in F1. We use F1 as the main metric for this comparison because the accuracy lead comes from crediting longer responses, as the WikiBigEdit results show directly.

WikiBigEdit shows this effect most clearly. In accuracy, LAMP trails in-context injection by 10.64 points, but in F1 it is 0.05 points ahead, and all three systems are within 0.76 points of each other. On the same set of predictions from In-context+LoRA, accuracy is 100.00% while F1 is 83.74%, a disagreement of 16.26 points. WikiBigEdit gold answers are long and often admit several correct phrasings, which is the case that containment-based accuracy handles worst. We keep accuracy as the default metric in the per-benchmark tables for consistency across benchmarks.

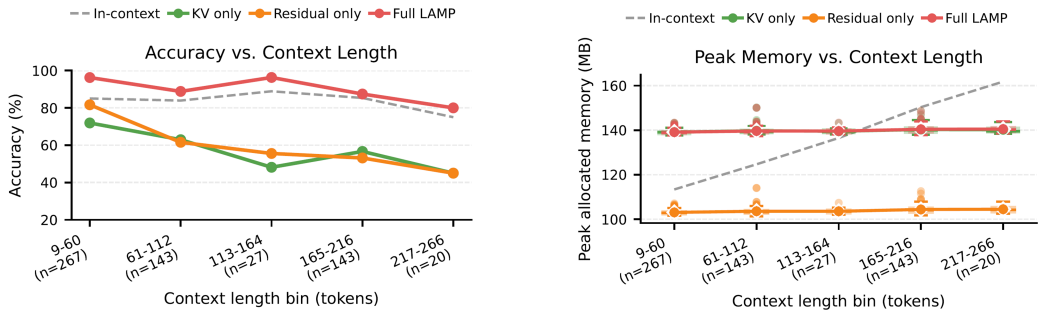

Figure 3: Context-length analysis for the injection-pathway ablation. Left: answer accuracy across context-length bins for in-context injection, KV-only, and residual-only injection. Right: peak allocated memory across the same bins.

K Pathway ablation across context lengths

Either pathway alone underperforms in-context injection, while the two together outperform it (Table 1). For the shortest evidence, each pathway alone performs close to in-context injection (Figure 3), which matches the behaviour that KBLaM and Larimar report for KV-style injection of short evidence. The residual-only pathway also uses markedly less memory than every other configuration, which makes it a lightweight option for injecting short content.

L Context-length analysis

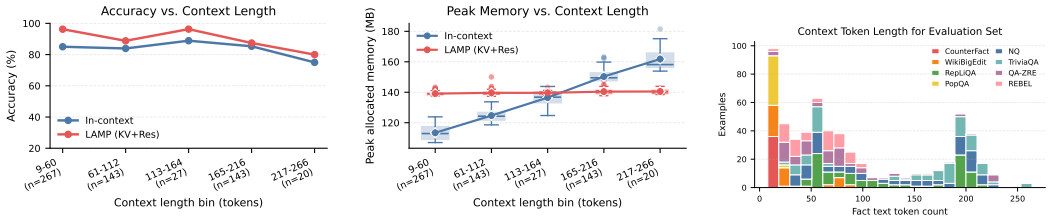

Figure 4: Context-length analysis, Qwen3-VL-8B. Left: answer accuracy across context-length bins for in-context injection and LAMP (KV+Res). Middle: peak allocated memory across the same bins. Right: context-token-length distribution of the evaluation set.

LAMP outperforms in-context injection in every context-length bin of the benchmark suite (Figure 4). The lead is largest in the short-to-medium bins (9 to 164 tokens) and remains positive for longer passages. Peak allocated memory stays nearly flat for LAMP, while the memory of in-context injection grows with passage length and crosses over around the 113 to 164 token bin. Below that bin, a passage is cheaper to keep as text than as a 256-latent memory.

The Phi-4-Mini results follow the same pattern (Figure 5). LAMP outperforms in-context injection in every context-length bin, and the memory crossover again falls around the 113 to 164 token bin.

M Verbatim recall on RULER

RULER's single-needle tasks hide one key-value pair inside a long filler context and ask for the value (Hsieh et al., 2024). We use the two single-needle tasks at 4,096 tokens. The whole context is one continuous passage, so there are no separate passages for the reranker to rank, and the memory itself must preserve the needle exactly. We encode the context as one memory with the distractor-tuned checkpoint of Table 4; in-context injection reads the same context as text with the frozen Qwen3-VL-8B.

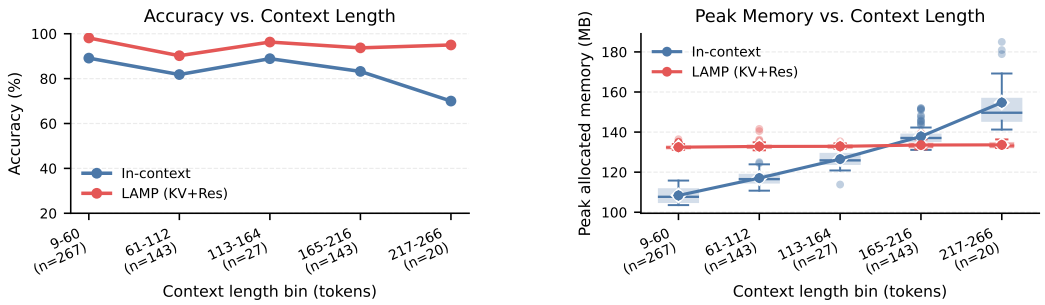

Figure 5: Context-length analysis for Phi-4-Mini. Left: answer accuracy across context-length bins for in-context injection and LAMP (KV+Res). Right: peak allocated memory across the same bins.

Table 13: Needle retrieval on RULER's single-needle tasks at 4,096 tokens (accuracy, %).

System	Single needle 1	Single needle 2	Average
LAMP, distractor-tuned memory	84	85	84.5
In-context injection	**100**	**100**	**100**

This is the expected limit of a fixed-size memory. Answering a question from evidence tolerates paraphrase, whereas a needle must be reproduced exactly, and a 256-latent memory of a 4,096-token context does not keep every string intact. When exact strings matter, the passage is better kept as text.

N RETRIEVAL-AUGMENTED TEXT AT 16K–256K

For each corpus size we use 100 questions. Qwen3-Embedding-0.6B retrieves the 48 passages nearest to the question by cosine similarity, Qwen3-Reranker-4B reranks them (Zhang et al., 2025), and the top passage is given either to LAMP as one memory or to the frozen Qwen3-VL-8B as text with the in-context template, with greedy decoding in both cases. As a robustness check, we repeat the comparison with the smaller Qwen3-Reranker-0.6B. Each ranking is computed once per corpus and stored with passage hashes, so both readers receive byte-identical passages. Table 14 gives both metrics over all questions.

Table 14: Accuracy / F1 (%) over all 100 questions per corpus size, reading the top passage of each reranker.

	Qwen3-Reranker-4B		Qwen3-Reranker-0.6B		
Corpus	**LAMP**	**Text**	**LAMP**	**Text**	**Whole corpus as text**
16K	91.0 / 92.96	91.0 / 87.24	89.0 / 90.96	91.0 / 86.91	87.0 / 78.60
32K	90.0 / 92.53	91.0 / 87.15	89.0 / 91.53	91.0 / 87.15	80.0 / 75.01
64K	90.0 / 92.53	91.0 / 87.15	89.0 / 91.53	91.0 / 87.15	out of memory
128K	89.0 / 91.81	90.0 / 86.27	88.0 / 90.81	90.0 / 86.27	out of memory
256K	86.0 / 88.53	89.0 / 85.15	86.0 / 88.53	88.0 / 84.15	out of memory

With either reranker, LAMP has the higher F1 at every corpus size and text the equal or higher accuracy. Table 15 separates retrieval from reading.

When the supporting passage is retrieved, each reader scores almost the same at every corpus size, so the decline at 256K comes from retrieval, which both systems share. Given the supporting passage directly, without retrieval, LAMP and text both reach 91.0% accuracy, with F1 of 92.96 and 87.24, while the closed-book model reaches 33.0% on the same questions.

Table 15: Retrieval and reading at each corpus size. R@1: how often the supporting passage is ranked first. The last two columns give accuracy / F1 (%) with Qwen3-Reranker-4B, restricted to questions whose supporting passage was retrieved.

		R@1 (%)			Supporting passage retrieved	
Corpus	**Passages**	**Embedding**	**0.6B**	**4B**	**LAMP**	**Text**
16K	269	95	98	100	91.0 / 92.96	91.0 / 87.24
32K	560	95	98	99	90.9 / 92.89	91.9 / 87.45
64K	1,145	93	98	99	90.9 / 92.89	91.9 / 87.45
128K	2,314	93	97	98	90.8 / 92.81	91.8 / 87.32
256K	4,610	90	94	95	90.5 / 92.59	92.6 / 87.97

Why accuracy and F1 disagree. Both readers answer with a median of two words, but about 8 of text's correct answers per corpus size list several candidates, for example "Germany, Austria-Hungary, Ottoman Empire, and Bulgaria" for the gold answer Austria-Hungary. Accuracy counts these as correct and F1 does not, and LAMP gives no such answer at any corpus size. Text also paraphrases correct answers more often ("Arsenal Football Club" for Arsenal), which F1 partly penalizes, so neither metric alone settles the comparison, and we report both.

Cost. Table 16 gives medians at 16K over the 100 questions (10 for the whole corpus), measured on the top passages of Qwen3-Reranker-0.6B; a reader's cost depends only on the length of the passage it reads. Retrieval is shared by both readers: embedding the query takes about 10 ms, and reranking 48 candidates with Qwen3-Reranker-0.6B takes 154 ms. LAMP's cost is the same at every corpus size.

Table 16: Cost per question at 16K. Memory is the peak for the whole request.

System	**Evidence read (tokens)**	**Time to first token (ms)**	**Memory (MB)**
LAMP, top passage	∼60	43 (16 of them encoding)	140
Text, top passage	∼60	38	131
Text, whole corpus	18,420	2,014	8,072

O LATENT TOKEN BUDGET

LAMP's per-layer KV budget is set by M. A smaller M gives a smaller cache but compresses the evidence more aggressively. We train Qwen3-VL-8B variants with $M \in \{16, 64, 256\}$ and keep all other settings fixed. Even $M{=}16$ reaches 84.8%, level with the 84.7% of in-context injection, so the dual-pathway design stays competitive with a small budget (Table 1). Short-evidence benchmarks (PopQA, REBEL, QA-ZRE) are nearly saturated at $M{=}16$, while the longest and most detail-heavy passages gain the most from a larger budget: from $M{=}16$ to $M{=}256$, TriviaQA improves by 11.1 points (86.1% to 97.2%) and NQ by 14.4 points (79.3% to 93.7%).

Across context lengths, smaller budgets match $M{=}256$ on short evidence but fall behind as passages grow. The memory plot also shows that every budget stays flat while in-context injection grows: $M{=}16$ uses less memory than in-context injection at every length and $M{=}64$ beyond the shortest bin, while $M{=}256$ crosses it at around 113 to 164 tokens (Figure 6).

P PERCEIVER DEPTH

The LAMP memory encoder uses a Perceiver-style resampler to compress variable-length backbone token features into a fixed set of latent memory tokens. Each Perceiver layer applies cross-attention from the latent tokens to the selected backbone features, followed by latent self-attention and feed-forward updates. Increasing the depth therefore gives the memory encoder more capacity to integrate evidence across the supporting passage before projection into the target LLM.

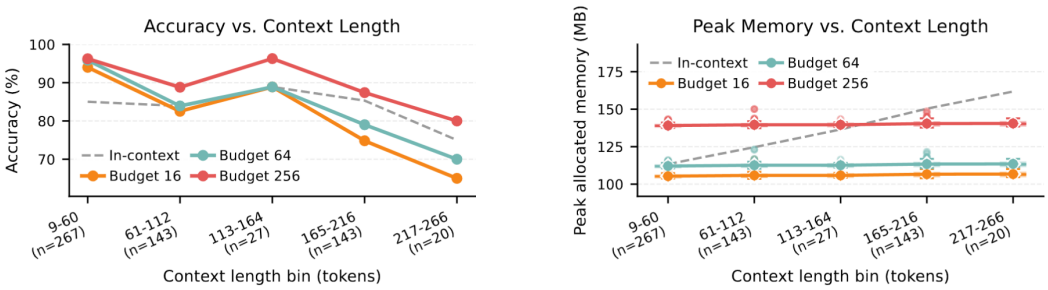

Figure 6: Context-length analysis under different injected-token budgets. Left: answer accuracy for the 16-, 64-, and 256-latent variants, with in-context injection as a dashed reference. Right: peak allocated memory for the same budgets.

The ablation block of Table 1 shows that increasing the Perceiver depth from 4 to 6 layers improves overall accuracy from 90.3% to 91.8%. The gains are concentrated on benchmarks requiring more detailed evidence integration, including RepLiQA, TriviaQA, and Natural Questions, while relation-extraction and counterfactual benchmarks remain unchanged. This suggests that additional Perceiver depth mainly improves the quality of the compressed memory for longer or more compositionally demanding evidence, rather than raising performance uniformly across tasks.

Q MID-GENERATION UPDATE

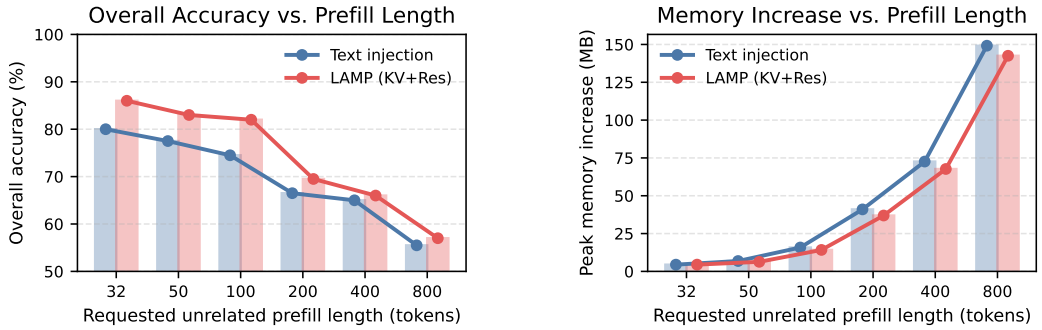

Figure 7: Mid-generation update. Left: accuracy versus the number of tokens generated before injection (prefill length). Right: peak-memory increase versus prefill length.

In the mid-generation update setting, the LLM first generates an unrelated story and is then redirected to answer questions from the same test set, with the evidence injected as text or through LAMP before it answers. LAMP is more accurate at every prefill length (Figure 7), by about 6.0, 5.5, and 8.0 points at 32, 50, and 100 tokens. The gap narrows to 3.0 and 1.0 points at 200 and 400 tokens, where steering the model becomes harder. Because LAMP adds evidence without autoregressively processing it through the existing self-attention history, its memory also grows slightly more slowly than that of in-context injection. At 200 tokens, the peak increase is 36.9 MB for LAMP against 41.0 MB for in-context injection, and at 400 tokens it is 67.7 MB against 72.6 MB.

R LAMP COUNTERFACT CORRECTION EXAMPLES

LAMP examples containing both `target_true` and `target_new`.

Example 1 **Q:** Manu Chao speaks the language
target_true: French
target_new: Italian

output: "Manu Chao speaks **Italian**. While he was born in France and speaks **French** fluently, he is also known to speak Italian . . . "

Example 2 **Q:** Fish Islands is located in the continent
target_true: Antarctica
target_new: Africa
output: "Fish Islands is located in the continent of **Africa**. This is a correction based on geographic accuracy. While the phrase 'Fish Islands' may evoke **Antarctica** due . . . "