FROM LOOKUPS TO LESSONS: TURNING RETRIEVED KNOWLEDGE INTO PARAMETRIC MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) helps large language models (LLMs) answer factual questions beyond their training data, but it adds storage and inference costs. Distilling previously retrieved information into parametric memories enables reuse without online retrieval. However, training the memory to mimic a retriever rewards accurate independent predictions, which do not necessarily improve the final output quality when combined with the LLM's existing predictions. To address this, we introduce **ResMem**, in which the LLM reads retrieved text offline to generate teacher targets. This design avoids storing massive token representations, unlike existing parametric memory approaches such as MLP Memory, which relies on k-nearest neighbor (kNN) datastores. Furthermore, by using the LLM as an offline teacher that reads privileged retrieved evidence alongside the input, we train a parametric memory to learn an additive logit residual. This residual aligns the predictions of a student sharing the same backbone with the teacher distribution, while self-distillation updates remain confined to the parametric memory. On a Wikipedia subset, ResMem reduces the storage retained for teacher construction by 94.7% compared with the evaluated kNN pipeline. Across five question answering datasets, ResMem raises mean token F1 from 32.83% for the frozen LLM to 34.48%. It also breaks 21% fewer of the LLM's correct answers than a kNN-trained memory. These findings validate our motivation of treating retrieval as a learning resource, with parametric memory shaped by how retrieved knowledge is used to inform the model's future predictions.

1 INTRODUCTION

Nowadays, many LLM applications need information beyond the current prompt, such as knowledge from external corpora and facts introduced in earlier interactions (Park et al., 2023; Zhang et al., 2025). One option is to keep such information in the model's active context. However, longer context windows do not ensure reliable use of information at every position (Liu et al., 2024b), and growing histories still require managing a finite active context (Packer et al., 2023). Retrieval-augmented generation (RAG) is a widely used way to recover relevant evidence from external storage (Lewis et al., 2020). This access requires maintaining a corpus and its index, while retrieving and reading the evidence adds inference latency. At corpus scale, these costs motivate storing knowledge for use without online retrieval.

Retriever-pretrained parametric memories offer such retrieval-free storage: they learn what a retriever would return and keep it in network weights, so inference needs no index or search. Building on kNN language models (Khandelwal et al., 2020), MLP Memory and Memory Decoder train a memory to reproduce a retriever's next-token distribution, then interpolate its prediction with a frozen base model (Wei et al., 2025; Cao et al., 2025). MLP Memory reports storage compression and faster inference, while Memory Decoder enables reuse across compatible base models. Both memories, however, learn to match the retriever alone and meet the base model only at inference. Their training loss never sees the base prediction (Figure 1a). A memory that reproduces the retriever well can therefore still change the combined prediction in unhelpful ways. For example, it can override a correct base answer.

We introduce ResMem, which trains the memory through the base prediction it will later modify (Figure 1b). The frozen base model serves as its own teacher by reading the query together with

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.

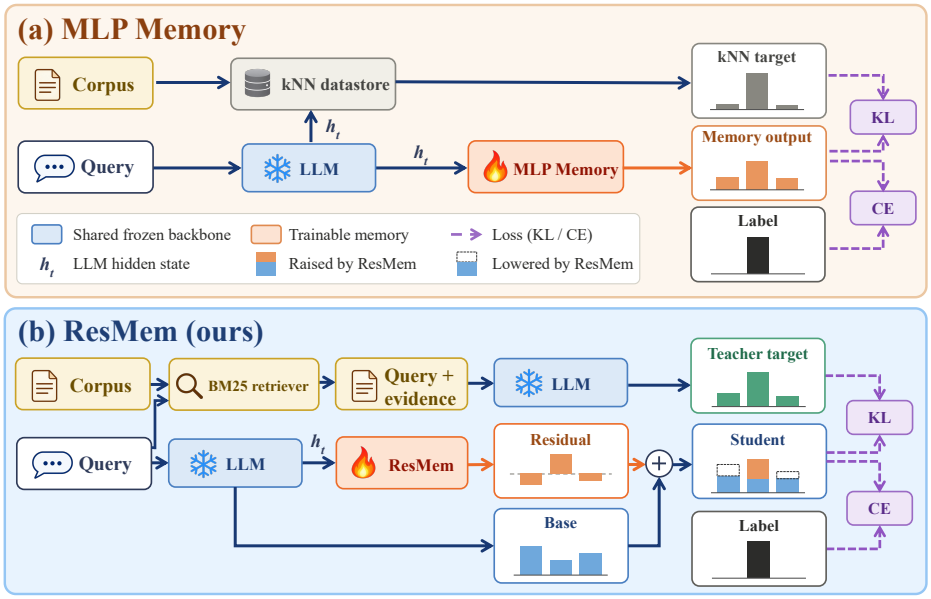


Figure 1: Training pipelines. (a) MLP Memory fits a memory alone to kNN targets; the base enters only at inference (Eq. 5). (b) ResMem fits a logit residual so that base plus residual matches an evidence-reading teacher; inference drops the teacher and scales the residual by γ (Eq. 3).

retrieved evidence, while the student sees only the query. The memory adds a logit residual to the student's logits, and the distillation and next-token losses are computed on this sum. Because the loss is computed on the base prediction plus the residual, positions where the base already matches the training target give little gradient. The memory thus learns a correction to the base, not a second copy of the retriever. This builds on context distillation and retrieval-free dialogue distillation (Snell et al., 2022; Zhu et al., 2024). Inference then needs only the frozen base and the trained memory, with a coefficient γ setting how strongly the residual adjusts the base (Eq. 3).

Because retriever-pretrained memories draw their supervision from a kNN datastore of corpus representations, they also pay a construction cost. Memory Decoder names the neighbor search over this datastore as a training bottleneck (Cao et al., 2025, Sec. 9), and later work scales the search with compression and sharding (Wei et al., 2026). An evidence-reading teacher needs no such datastore. ResMem retrieves passages with BM25 and caches only the teacher's top-64 targets, which cuts retained construction storage from 602.0 to 32.0 GiB on a subset of English Wikipedia, a 94.7% (18.8×) reduction (Section 5.2).

On five knowledge-QA datasets, ResMem improves exact match (EM) and token F1 over the base model on every dataset and raises mean F1 from 32.83 to 34.48. Compared with MLP Memory, it corrects 11% more base errors and breaks 21% fewer correct base answers, so most of its advantage comes from avoided harms. Under a shared teacher, the residual objective mainly preserves correct answers, and its extra repairs come with the evidence-reading teacher. Memory Decoder falls below both memories under the same teacher.

Our main contributions are: (i) We introduce residual self-distillation, which trains an additive logit residual through the joint base-plus-memory prediction and keeps it at inference. (ii) We replace the kNN datastore with text retrieval and an evidence-reading teacher. This cuts retained construction storage by 94.7% (18.8×). The teacher's targets also keep the shape of the base prediction. The kNN targets are much sharper. (iii) We evaluate the recipe and its parts on knowledge QA, answer-level corrections and harms, and shared-teacher ablations.

2 Related Work

Retrieval and parametric memory. Retrieval keeps information outside model parameters. RAG, RETRO, and Atlas condition generation on retrieved text (Lewis et al., 2020; Borgeaud et al., 2022;

Izacard et al., 2023), and xRAG compresses each retrieved document into a one-token representation (Cheng et al., 2024). kNN-LM instead forms a next-token distribution from neighboring contexts (Khandelwal et al., 2020). Memorizing Transformers and LongMem attend to stored representations of earlier tokens (Wu et al., 2022; Wang et al., 2023). Product quantization, GPU similarity search, and efficient kNN-LMs reduce the cost of such stores (Jégou et al., 2011; Johnson et al., 2021; He et al., 2021). All of these methods query a store during inference. Parametric memories instead learn to reproduce a store, building on the view of feed-forward layers as key–value memories (Geva et al., 2021). kNN knowledge distillation moves retrieval into preprocessing for translation (Yang et al., 2022). MLP Memory and Memory Decoder train separate parametric memories to imitate retriever distributions (Wei et al., 2025; Cao et al., 2025). Memory Decoder at Scale extends this route to larger memories and corpora with compressed, sharded search (Wei et al., 2026). MemSFT adds learned token-level routing between a retrieval-supervised memory and the frozen backbone (Wang et al., 2026a). We instead train the memory's contribution through the joint prediction. Parametric RAG still retrieves documents, each encoded as LoRA parameters (Su et al., 2025). Test-time training layers, Titans, and δ-mem update memory states online while processing a stream (Sun et al., 2025; Behrouz et al., 2025; Lei et al., 2026b). ResMem instead trains its memory offline.

Distillation and output composition. Knowledge distillation transfers a teacher's predictions to a student (Hinton et al., 2015), and the teacher may use privileged information that is absent at deployment (Lopez-Paz et al., 2016). Context distillation internalizes the benefit of extra instructions, examples, or reasoning in language models (Snell et al., 2022). Cartridges uses it to learn compact KV caches offline (Eyuboglu et al., 2025). TTCD updates a short-context student's fast weights online from long-context supervision (Wang et al., 2026b). Born-Again Networks show that the teacher need not be larger than the student (Furlanello et al., 2018). In ResMem, teacher and student share the frozen backbone, and only the parametric memory is trained. Retrieval-conditioned distillation carries the benefits of retrieval into retrieval-free students, as in QUILL, RA2FD, and InstructRetro (Srinivasan et al., 2022; Zhu et al., 2024; Wang et al., 2024). TS-Memory distills kNN-induced corrections into a time-series adapter (Lyu et al., 2026). TransMem is the closest work. It also trains a memory by evidence-conditioned self-distillation with a shared frozen backbone (Lei et al., 2026a). However, its student reads the full long context and outputs gated hidden-state shifts. Our student lacks the evidence and outputs a logit residual. DExperts, contrastive decoding, proxy-tuning, and context-aware decoding combine or contrast model predictions at decoding time (Liu et al., 2021; Li et al., 2023b; Liu et al., 2024a; Shi et al., 2024). ResMem instead learns its residual through a loss on the combined prediction and needs evidence only offline. ResMem also trains on fixed teacher-forced continuations. On-policy distillation, OPSD, and OPSDL use student-generated trajectories instead (Agarwal et al., 2024; Zhao et al., 2026; Zhang et al., 2026).

3 METHODOLOGY

ResMem writes external knowledge into a parametric memory while keeping the language model frozen. The memory architecture follows MLP Memory. ResMem changes the teacher and the objective (Section 3.2), and the teacher's cached targets replace the kNN datastore (Section 3.4).

3.1 MEMORY INPUT AND ARCHITECTURE

For a token sequence x, causal position t predicts the next token $y_t = x_{t+1}$. The frozen base produces logits $z_{B,t} \in \mathbb{R}^{|V|}$ over vocabulary V. At the same position, we extract $h_t \in \mathbb{R}^d$ at the input to the final Transformer block's feed-forward sublayer (Appendix A).

The memory maps this contextual vector to vocabulary logits, $s_{\theta,t} = m_\theta(h_t) \in \mathbb{R}^{|V|}$. Following MLP Memory (Wei et al., 2025), it consists of a stack of feed-forward blocks with RMS normalization and residual connections, followed by a final normalization and its own trainable vocabulary projection. Each gated block acts on one position; the base representation already carries the preceding context, so the memory needs no attention over tokens. Only memory parameters, including its output projection, are trained.

3.2 RESIDUAL SELF-DISTILLATION

Distilling the privileged gain into memory. On-policy self-distillation (OPSD) (Zhao et al., 2026) turns privileged input into supervision: the same model, given information the student lacks, produces a dense token-level output distribution. Teacher and student share one backbone, so this distribution lives in the base model's own output space. It differs from the base prediction only through what the privileged input adds. A residual on the base must learn exactly this part (Eq. 2). ResMem uses retrieved evidence as the privileged input, as in context distillation (Lopez-Paz et al., 2016; Snell et al., 2022), and distills this gain into the parametric memory. The teacher $\pi_T(\cdot \mid c, x_{\le t})$ is the frozen model reading passages c retrieved by BM25 (Robertson & Zaragoza, 2009) before the span x. The student $\pi_S(\cdot \mid x_{\le t}) = \mathrm{softmax}(z_{B,t} + s_{\theta,t})$ is the same model reading only x, plus the memory's residual. Training minimizes the per-token divergence between the two roles over corpus prefixes,

$$\min_{\theta} \; \mathbb{E}_x \sum_t D_{\mathrm{KL}}\big(\pi_T(\cdot \mid c, x_{\le t}) \,\big\|\, \pi_S(\cdot \mid x_{\le t})\big). \tag{1}$$

Only the memory is updated. The gain that the evidence gives the teacher is written into θ, and the backbone stays fixed.

Teacher targets. Teacher and student follow the same continuation under teacher forcing. After the evidence prefix, teacher outputs are aligned to the student's next-token targets. The evidence changes the supervision, while the memory input remains h_t from the student base. Retrieval and teacher forwarding run once, offline. At each supervised position, we cache the teacher's top-K token probabilities with $K = 64$. We renormalize them and put zero mass elsewhere, which gives $p^{(K)}_{T,t}$. This target is zero outside K tokens, while π_S has full support. Eq. 1 uses the forward KL, because the reverse direction would be infinite. Algorithm 1 summarizes the procedure, and the paragraphs below specify the student and the loss.

Joint prediction. We train the memory through its joint prediction with the base ($\oplus$ in Figure 1b), $z_{R,t} = z_{B,t} + s_{\theta,t}$ and $q_{R,t} = \mathrm{softmax}(z_{R,t})$. During training the residual enters at unit scale, which corresponds to $\gamma = 1$ in the inference rule of Section 3.3. Because the loss evaluates $q_{R,t}$ with $z_{B,t}$ fixed, the memory learns the change needed by the combined prediction.

Loss components. Let $\mathcal{C}$ contain all valid causal prediction positions in a training batch and $\mathcal{A} \subseteq \mathcal{C}$ the positions selected for label supervision. As in knowledge distillation with soft targets and labels (Hinton et al., 2015), we combine teacher-to-student KL, $\mathcal{L}_{\mathrm{KL}} = |\mathcal{C}|^{-1} \sum_{t \in \mathcal{C}} D_{\mathrm{KL}}(p^{(K)}_{T,t} \,\|\, q_{R,t})$, with next-token cross-entropy, $\mathcal{L}_{\mathrm{CE}} = -|\mathcal{A}|^{-1} \sum_{t \in \mathcal{A}} \log q_{R,t}(y_t)$. KL transfers the teacher's distribution over possible continuations; CE supervises the observed target token. A hyperparameter $\alpha \in [0, 1]$ weighs the two, $\mathcal{L} = \alpha\, \mathcal{L}_{\mathrm{KL}} + (1 - \alpha)\, \mathcal{L}_{\mathrm{CE}}$. Each term is averaged over its own selected positions, excluding padding. In corpus-continuation training, both terms cover the complete valid span, so $\mathcal{A} = \mathcal{C}$.

What the residual learns. With $\mathcal{A} = \mathcal{C}$, each position contributes $\ell_t = -\sum_{v \in V} \tilde{p}_t(v) \log q_{R,t}(v)$ up to a constant, where the training target $\tilde{p}_t = \alpha\, p^{(K)}_{T,t} + (1 - \alpha)\, e_{y_t}$ mixes the teacher with the one-hot label e_{y_t}. The gradient of ℓ_t with respect to the residual logits is $\partial \ell_t / \partial s_{\theta,t} = \mathrm{softmax}(z_{B,t} + s_{\theta,t}) - \tilde{p}_t$. A position whose joint prediction already matches the target therefore produces no update. While the residual is small, these are the positions where the base already agrees with the target. Let $\mathcal{S}_t$ be the support of $\tilde{p}_t$, the K cached tokens and y_t. If the memory could fit every position exactly, the loss would approach its infimum with $s_{\theta,t} \to -\infty$ outside $\mathcal{S}_t$ and, on $\mathcal{S}_t$,

$$s^{\star}_{\theta,t} = \log \tilde{p}_t - z_{B,t} + c_t, \tag{2}$$

with a free per-position constant c_t. The residual thus learns the log-ratio between the target and the base prediction. The standalone objective of Section 3.3 instead drives the memory's own distribution toward $\tilde{p}_t$, regardless of the base.

3.3 INFERENCE AND MATCHED CONTROL

At each decoding step, the base supplies h_t and $z_{B,t}$, and the memory produces $s_{\theta,t}$. A residual scale $\gamma \ge 0$ multiplies the memory logits before the final softmax:

$$z_{\mathrm{final},t} = z_{B,t} + \gamma s_{\theta,t}, \qquad p_{\mathrm{final},t} = \mathrm{softmax}(z_{\mathrm{final},t}). \tag{3}$$

Algorithm 1 Residual self-distillation

Require: frozen language model f, corpus spans $\mathcal{X}$, retriever R, cache size K, loss weight α

for all $x \in \mathcal{X}$ **do**
  $c \leftarrow R(x)$; $p_T^{(K)} \leftarrow$ renormalized top-K of $f(\cdot \mid [c; x])$ at every position
end for
repeat
  sample spans $x \sim \mathcal{X}$; $(h_t, z_{B,t}) \leftarrow f(x)$
  $q_{R,t} \leftarrow \mathrm{softmax}\big(z_{B,t} + m_\theta(h_t)\big)$
  $\mathcal{L} \leftarrow \alpha\, \mathcal{L}_{\mathrm{KL}} + (1-\alpha)\, \mathcal{L}_{\mathrm{CE}}$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
until the token budget is spent
return m_θ; inference uses $\mathrm{softmax}\big(z_{B,t} + \gamma\, m_\theta(h_t)\big)$

This retains the additive structure used in training; $\gamma = 0$ recovers the base prediction. Evidence retrieval and teacher forwarding are offline; inference uses only the base and learned memory (cost in Appendix D). At the optimum of Eq. 2, the scaled logits on $\mathcal{S}_t$ become $(1-\gamma)\, z_{B,t} + \gamma\, (\log \tilde{p}_t + c_t)$, and c_t cancels in the softmax. In this limit, tokens outside $\mathcal{S}_t$ receive zero probability for $\gamma > 0$, and on $\mathcal{S}_t$ the scaled readout is a geometric interpolation between the base distribution $p_{B,t} = \mathrm{softmax}(z_{B,t})$ and the training target:

$$p_{\mathrm{final},t} \propto p_{B,t}^{1-\gamma}\, \tilde{p}_t^{\gamma}, \tag{4}$$

so γ moves the prediction along a log-linear path from Base at $\gamma = 0$ to the fitted target at $\gamma = 1$.

The residual can still suppress a base answer, as the condition below shows.

For the shared-teacher standalone control, we reuse the architecture, targets, examples, position masks, loss weights, and optimizer. Its loss replaces $q_{R,t}$ with the memory-only distribution $q_{M,t} = \mathrm{softmax}(s_{\phi,t})$, where $s_{\phi,t}$ are the logits of a separately trained memory with parameters ϕ. Inference follows the MLP Memory probability mixture, with interpolation weight $\lambda \in [0, 1]$:

$$p_{\mathrm{final},t} = (1-\lambda)\, \mathrm{softmax}(z_{B,t}) + \lambda\, q_{M,t}. \tag{5}$$

Both readouts recover Base at zero. The mixture interpolates linearly between Base and the memory, and the additive readout interpolates geometrically (Eq. 4). We therefore select λ and γ separately for each readout (Section 5.4).

When a readout overrides the base. Consider a position where the base prefers token y to v, both in $\mathcal{S}_t$, with log-odds margin $m_B = \log p_{B,t}(y) - \log p_{B,t}(v) > 0$, while the fitted target prefers v with margin $m_T = \log \tilde{p}_t(v) - \log \tilde{p}_t(y) > 0$. By Eq. 4, the readout's log-odds of v over y are

$$\log \frac{p_{\mathrm{final},t}(v)}{p_{\mathrm{final},t}(y)} = \gamma\, m_T - (1-\gamma)\, m_B,$$

so the additive readout switches to v if and only if

$$\gamma\, m_T > (1-\gamma)\, m_B, \tag{6}$$

whereas the mixture of Eq. 5, with the memory at its standalone optimum $q_{M,t} = \tilde{p}_t$, switches if and only if $\lambda\, [\tilde{p}_t(v) - \tilde{p}_t(y)] > (1-\lambda)\, [p_{B,t}(y) - p_{B,t}(v)]$. The additive rule compares log-odds, so overriding a confident base token requires a correspondingly confident target. The mixture compares probability gaps, and these never exceed one. A memory that puts all its mass on v therefore overrides any base margin once $\lambda > 1/2$. Preserving confident base answers then forces a small λ. Under greedy decoding, this token-level condition is how a memory breaks a correct base answer (Section 5.3).

3.4 Construction storage

The cached targets are all that training needs from the teacher. Both routes cache them. The kNN route also retains a datastore and a FAISS IVF-PQ index. The RAG route retains the retrieved text and a BM25 index instead. Per valid position, a datastore entry holds a d-dimensional FP16 key and

an INT32 token ID. An index entry holds a P-byte PQ code and an 8-byte ID. A cached target holds K INT32 token IDs and K FP16 probabilities. Over n positions, the two routes retain

$$\begin{aligned} S_{\mathrm{kNN}}(n) &= (2d+4)n + (P+8)n + 6Kn, \\ S_{\mathrm{RAG}}(n) &= 6Kn + bn, \end{aligned} \tag{7}$$

where b is the text and BM25 index per position. With $d = 4096$, $P = 64$, and $K = 64$, the kNN route needs $8196 + 72 + 384 = 8652$ bytes per position. The ratio $S_{\mathrm{kNN}}/S_{\mathrm{RAG}} = 8652/(384 + b)$ does not depend on n, so the saving holds at every corpus size. It approaches 22.5 as b goes to zero, since the datastore alone takes $(2d + 4)/6K \approx 21.3$ times the bytes of the targets. Section 5.2 measures b and both routes.

4 Experimental Setup

Models. We use Mistral-7B-v0.3 (Jiang et al., 2023) as the base model. ResMem and MLP Memory each attach an eight-layer memory with hidden size 4096, intermediate size 14336, and approximately 1.54B parameters. Additional baselines are BM25-RAG (Robertson & Zaragoza, 2009; Lewis et al., 2020), LoRA (Hu et al., 2022), and Memory Decoder (Cao et al., 2025). On the Wikipedia subset, we train two memories with this architecture on a kNN teacher built from the subset. Both start from the same initialization, follow the same example order, and train for three epochs (220.8M tokens). One uses the standalone objective, and the other uses the residual objective (Section 5.3; Appendix A). For knowledge QA, the correction analysis, HaluEval, and the precision study, ResMem, MLP Memory, and Memory Decoder are each trained for one epoch on full English Wikipedia. MLP Memory and Memory Decoder follow their official pipelines, including the kNN teacher. The shared-teacher ablation trains eight new paired seeds for each objective and for Memory Decoder, all with equal exposure (Section 5.4).

Datasets. Following MLP Memory, we use the December 2021 English Wikipedia dump as the corpus for memory training. The Wikipedia subset, used for the storage comparison and these two memories, contains 74 of its 4,716 source shards, about 1/64 of the corpus. Knowledge QA contains 38,627 questions from Natural Questions (NQ) (Kwiatkowski et al., 2019), WebQuestions (Berant et al., 2013), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), and HotpotQA (Yang et al., 2018). We also use HaluEval's Dialogue, QA, and Summarization splits (Li et al., 2023a). Hardware, training schedules, and evaluation configurations are given in Appendix A.

5 Results

5.1 Knowledge QA Task Performance

The QA checkpoints are each trained for one epoch on full Wikipedia (Section 4). We report EM, token F1, and reference-answer PPL on 38,627 questions, averaging the five datasets equally. For each memory and dataset, we select λ or γ to maximize EM on the validation set. We then use that coefficient to compute EM, F1, and PPL on the evaluation set (Appendix A). A QA-supervised tokenwise gate changes mean EM by only $+0.07$ points and mean F1 by -0.02 points, so ResMem is reported without it (Table 6 in Appendix E).

ResMem has the highest EM and F1 and the lowest PPL in all three task families (Table 1). Over Base, it raises mean EM from 25.91 to 26.74 and mean F1 from 32.83 to 34.48. Both metrics improve on every dataset (Appendix B). Its F1 margin over MLP Memory grows from 0.32 points on open-domain QA to 1.08 on long-tail and 1.77 on multi-hop QA. On HaluEval hallucination detection, the same checkpoints raise mean accuracy from 54.83 to 63.09, against 60.34 for MLP Memory (Appendix C). Answer likelihood alone does not explain these results. BM25-RAG lowers open-domain answer PPL below Base (10.02 versus 11.11), yet it loses 5.42 EM points. Generation must choose its own words and decide when to stop.

5.2 Construction storage

We measure both routes of Eq. 7 on nested shard sets of the corpus, from the Wikipedia subset to 652M positions (Figure 2). At every set, the kNN datastore files match $(2d + 4)n$ bytes. A BM25 index and its passages, built on the same shards, grow linearly at 3.5 to 3.7 bytes per position.

Table 1: Knowledge QA on 38,627 questions. Open-domain QA averages NQ, WebQuestions, and TriviaQA; long-tail QA is PopQA; multi-hop QA is HotpotQA (per-dataset results in Appendix B). Subscripts give changes from Base: percentage points for EM and F1, and the relative reduction for PPL. Bold marks the best value per column.

	Open-domain QA			Long-tail QA			Multi-hop QA		
Method	EM ↑	F1 ↑	PPL ↓	EM ↑	F1 ↑	PPL ↓	EM ↑	F1 ↑	PPL ↓
Base	29.97	38.51	11.11	22.01	25.15	11.96	17.64	23.47	6.27
Non-parametric methods									
+ BM25-RAG	24.54$_{-5.42}$	31.87$_{-6.64}$	10.02$_{\downarrow 9.82\%}$	22.65$_{+0.64}$	25.86$_{+0.71}$	12.29$_{\uparrow 2.72\%}$	18.51$_{+0.87}$	26.27$_{+2.80}$	5.88$_{\downarrow 6.28\%}$
Parametric methods									
+ LoRA	30.00$_{+0.03}$	38.71$_{+0.20}$	10.60$_{\downarrow 4.67\%}$	22.43$_{+0.42}$	26.51$_{+1.36}$	11.61$_{\downarrow 2.99\%}$	17.63$_{-0.01}$	25.32$_{+1.85}$	5.93$_{\downarrow 5.42\%}$
+ Memory Decoder	30.19$_{+0.23}$	38.95$_{+0.44}$	10.84$_{\downarrow 2.47\%}$	22.12$_{+0.11}$	25.40$_{+0.24}$	11.80$_{\downarrow 1.38\%}$	17.77$_{+0.14}$	23.76$_{+0.28}$	6.17$_{\downarrow 1.66\%}$
+ MLP Memory	30.40$_{+0.44}$	39.41$_{+0.90}$	10.43$_{\downarrow 6.12\%}$	22.38$_{+0.37}$	25.83$_{+0.68}$	11.07$_{\downarrow 7.47\%}$	18.30$_{+0.66}$	24.52$_{+1.05}$	6.02$_{\downarrow 4.01\%}$
+ ResMem (Ours)	**30.61**$_{+0.65}$	**39.73**$_{+1.22}$	**9.77**$_{\downarrow 12.09\%}$	**22.85**$_{+0.84}$	**26.91**$_{+1.75}$	**10.99**$_{\downarrow 8.13\%}$	**19.03**$_{+1.39}$	**26.29**$_{+2.82}$	**5.75**$_{\downarrow 8.30\%}$

The index over all of English Wikipedia takes 16.7 GB. On the subset, b is 72.9 bytes. The stored files add small overheads, such as span token IDs. They take 8665 and 461 bytes per position, a ratio of 18.8. The kNN route retains a 570.011 GiB datastore and a 5.069 GiB FAISS index. The RAG route retains 5.067 GiB of retrieved text and BM25 index. Both cache 26.967 GiB of targets. The totals are 602.047 versus 32.033 GiB, a 94.7% (18.8×) reduction in retained assets.

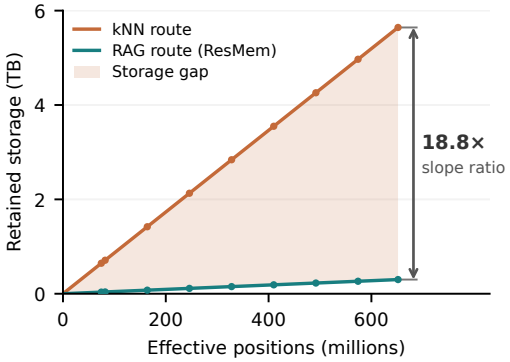


Figure 2: Retained storage of the kNN and RAG routes, measured on nested shard sets of English Wikipedia up to 652M positions.

5.3 Correction and harm analysis

A change in mean EM is the net of two opposing effects: the memory fixes some questions the base model answers wrongly and breaks some it answers correctly. We pair the answers to all 38,627 questions with the frozen Base answers and count corrections C (Base wrong, memory right) and harms H (Base right, memory wrong) under normalized EM. The net EM gain is $(C - H)/Q$ over the Q questions. Each method uses one global coefficient chosen on 4,091 NQ/TriviaQA development questions disjoint from the test sets: $\lambda^* = 0.3$ for MLP Memory, $\gamma^* = 0.3$ for ResMem, and $\lambda^* = 0.2$ for Memory Decoder. Figure 3a traces correction against harm rate as the coefficient varies. Net EM rises while a curve is steeper than the $\Delta\mathrm{EM} = 0$ line. Figure 3b splits the net gain into corrections and harms at the selected coefficients.

ResMem corrects 704 answers and harms 381, against 633 and 482 for MLP Memory. Its net EM gain is 0.84 points, against 0.39 (paired difference 0.45, 95% interval [0.25, 0.64]). Most of this difference comes from the 101 avoided harms. Memory Decoder nets -0.04 points, with 124 corrections and 138 harms.

One source of the harm gap is the shape of the kNN target (Figure 4a). Over all 74.6M cached positions, a single token carries at least 95% of its mass at 87% of positions. Its median entropy is 0, and its mean top-1 probability is 0.95. The evidence-reading teacher is shaped like the base instead. It does so at 24% of 1.05M positions, and the base at 22%. Their mean top-1 probabilities are 0.60 and 0.59. This sharpness is a known property of kNN-LM. The kNN distribution comes from $k \ll |V|$ neighbors through a softmax over negative distances, so it is sharper than a model's output (Xu et al., 2023). A higher temperature spreads the mass (Khandelwal et al., 2021). However, it also gives distant, noisier neighbors almost the same weight as the nearest one (Zheng et al., 2021). The original kNN-LM therefore mixes this distribution at a small weight, $\lambda = 0.25$ on WikiText-103 (Khandelwal et al., 2020). Our kNN teacher follows the MLP Memory pipeline at $T = 16$. This is flatter than the original $T = 1$, so the choice favors the baselines. Even so, the target stays sharp.

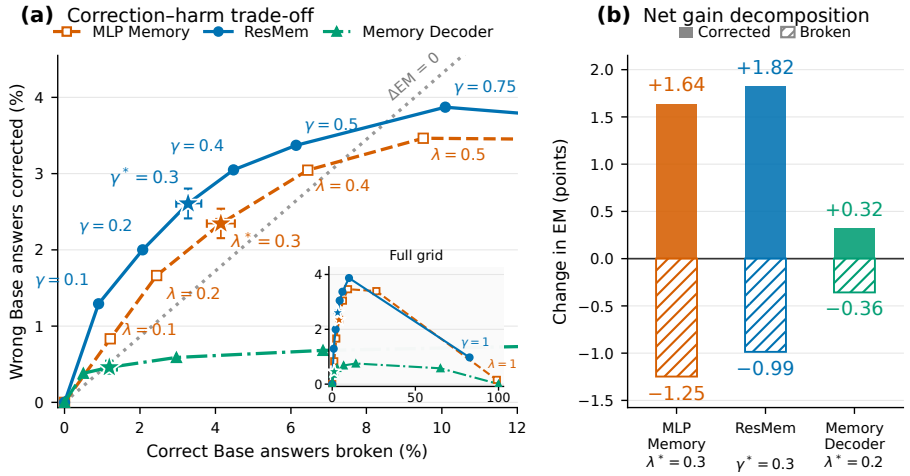


Figure 3: Correction–harm trade-off on 38,627 QA questions. (a) Correction versus harm rates across coefficients; stars (*) mark coefficients selected on development data, and the dotted line marks zero net gain. (b) Correction and harm contributions at the selected coefficients.

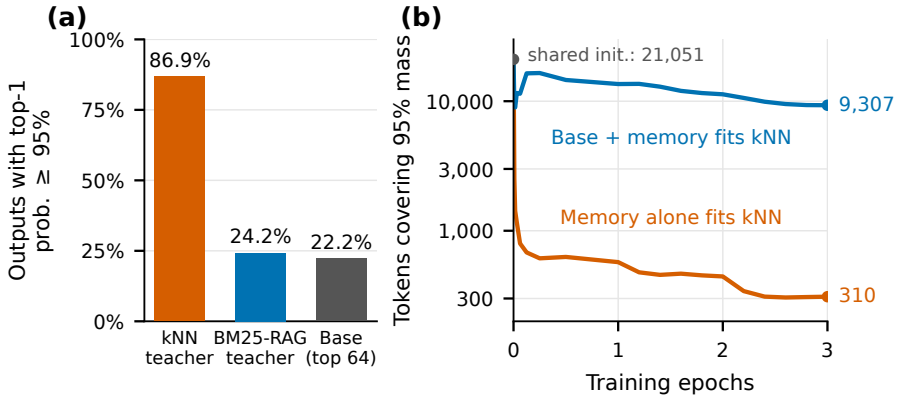


Figure 4: Distribution concentration. (a) Percentage of positions whose most likely token has probability at least 95%; the base prediction is truncated to its top 64. (b) Tokens covering 95% of the memory's own softmax, averaged over positions, for memories trained on the kNN teacher from a shared initialization, either alone (as in MLP Memory) or through base plus memory (as in ResMem). The standalone memory's softmax is its mixture component. The residual's softmax is not used at inference and is shown for reference.

A teacher shaped like the base suits a residual. The residual learns the log-ratio between the target and the base (Eq. 2). Our teacher is the base itself, reading retrieved evidence. The two share an output space and a similar shape. The residual therefore mainly encodes what the evidence changes. The teacher also keeps the base's uncertainty. When several continuations are plausible, it does not force one of them. The kNN target commits to one token at most positions. A memory that fits it learns to commit as well.

MLP Memory shows this effect. On the same kNN teacher, initialization, and example order, a memory trained alone narrows from 21,051 tokens covering 95% of its mass to 310 (Figure 4b). In MLP Memory, this distribution is the component that the mixture adds to the base (Eq. 5). A sharp component can override a confident base answer once λ grows (Section 3.3). The additive readout only shifts base log-odds (Eq. 6). The residual's own softmax is not used at inference, so we judge it by the combined predictions. Figure 3 shows the expected pattern. MLP Memory breaks 482 correct answers, and ResMem breaks 381. The MLP Memory curve flattens earlier. Section 5.4 holds the teacher fixed to test the objective and readout alone.

Table 2: Shared-teacher QA ablation with the BM25-RAG teacher, equal exposure, and eight paired seeds. Top: native configurations on 38,627 questions; Gold NLL is the token-weighted reference-answer NLL. Bottom: F1 at the development-selected coefficient / coefficient one on 4,096 diagnostic questions (Base 32.29).

Native configuration	EM	F1	Gold NLL
Base	25.91	32.83	2.295
Standalone + mixture	25.87	33.12	2.291
Memory Decoder + mixture	25.90	32.87	2.329
Residual + additive	26.03	33.24	2.273

Training objective	Mixture readout	Additive readout
Standalone	32.65 / 21.87	32.29 / 28.44
Residual	32.29 / 0.06	32.66 / 30.68

5.4 Ablations

The comparisons above differ in teacher, objective and readout, and memory architecture at once. Holding the teacher fixed, we first vary the objective and readout, and then the architecture.

Objective and readout. Both memories use the same BM25-RAG teacher, cached inputs, eight-layer architecture, paired initialization, and example order. Eight paired seeds each receive 8,388,608 supervised-position exposures over eight epochs. Each configuration selects one coefficient on 1,024 development questions disjoint from the test sets. Pairing the seeds lets each comparison share its initialization and data order.

We call the standalone objective with mixture readout and the residual objective with additive readout the native configurations. Both improve mean F1 over Base (Table 2). The residual configuration leads by 0.117 points, with a 95% question-cluster bootstrap interval of $[-0.020, 0.258]$, or $[-0.180, 0.477]$ when seeds are also resampled.

In the four-cell diagnostic on 4,096 shared questions, both crossed configurations select coefficient zero and recover Base. Residual scores read as a standalone distribution yield an F1 of 0.06, so they are useful only when added to the base logits. Pooled over the eight runs at native operating points, the residual configuration preserves 96.11% of Base-correct answers, against 94.05%. It repairs 1.51% of Base errors, against 2.25% (point estimates). With the teacher fixed, it thus trades some repair for preservation.

Architecture. Memory Decoder replaces the hidden-state memory with a separate 0.47B-parameter decoder that reads the token sequence (Appendix A). With the same supervision, exposure, and optimizer schedule and a mixture readout, it reaches F1 32.87. This is 0.26 and 0.37 points below the standalone and residual configurations ($[-0.42, -0.09]$ and $[-0.49, -0.25]$). Development selection sets its coefficient to zero for six of the eight seeds (Table 2). Under this training budget and at these sizes, the hidden-state memory is the stronger design.

Unlike the shared-teacher runs, full-recipe ResMem both corrects more and harms less than MLP Memory (Section 5.3). The objective and readout thus contribute to the better preservation. The extra repairs come with the evidence-reading teacher, in line with the sharper kNN targets in Figure 4a.

6 Conclusion

ResMem distills a retrieval-augmented teacher into an additive logit residual on a frozen base model. On a Wikipedia subset, it cuts retained construction storage by 94.7% ($18.8\times$). On five knowledge-QA datasets, it raises mean F1 from 32.83 to 34.48. It also corrects more base errors and harms fewer correct answers than MLP Memory. With a shared teacher, the residual objective mainly preserves correct answers, and the extra repairs come with the evidence-reading teacher. The storage saving and the extra repairs thus share one source, the teacher that reads retrieved text.

With only the parametric memory trained, the base model stays unchanged. A limitation of this design is that the memory reads one hidden state and writes only to the output logits. The teacher's gain is therefore not integrated through the base model's intermediate layers. Memories that read from and write to several layers are a natural next step.

AI USE STATEMENT

In this work, generative AI tools played a supporting role, and the authors led all research decisions. Among tasks with required disclosure, we used generative AI tools to help formalize mathematical derivations, give feedback on experiments, and help implement experiment code. The authors developed the method and verified each formalized derivation. The authors also reviewed and tested the AI-assisted code before using its results. AI feedback on experiments informed decisions that the authors made. We have not used generative AI tools to generate synthetic data, propose research ideas or hypotheses, or clean datasets. Among tasks with recommended disclosure, we used generative AI tools to suggest the structure of the paper and to edit it for readability. We have reviewed all AI-assisted work. The authors checked every reported number against the stored experiment outputs and every reference against its DBLP record. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 and Appendix A describe the models, data, and evaluation settings. Code for the ResMem training pipeline and the main evaluation pipelines is available at `https://anonymous.4open.science/r/ResMem-4329`.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*. OpenReview.net, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *NeurIPS*, 2025.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pp. 1533–1544. ACL, 2013.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George 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 W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 2022.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. In *NeurIPS*, 2025.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. In *NeurIPS*, 2024.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Ré. Cartridges: Lightweight and general-purpose long context representations via self-study. *CoRR*, abs/2506.06266, 2025.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1602–1611. PMLR, 2018.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *EMNLP (1)*, pp. 5484–5495. Association for Computational Linguistics, 2021.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *EMNLP (1)*, pp. 5703–5714. Association for Computational Linguistics, 2021.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

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 *ICLR*. OpenReview.net, 2022.

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. *J. Mach. Learn. Res.*, 24:251:1–251:43, 2023.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL (1)*, pp. 1601–1611. Association for Computational Linguistics, 2017.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*. OpenReview.net, 2020.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. In *ICLR*. OpenReview.net, 2021.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019.

Haodong Lei, Junming Liu, Yirong Chen, Pinlong Cai, Botian Shi, Ding Wang, and Hongsong Wang. Transmem: Transforming hidden states into memory for large language models. *CoRR*, abs/2607.29032, 2026a.

Jingdi Lei, Di Zhang, Junxian Li, Weida Wang, Kaixuan Fan, Xiang Liu, Qihan Liu, Xiaoteng Ma, Baian Chen, and Soujanya Poria. δ-mem: Efficient online memory for large language models. *CoRR*, abs/2605.12357, 2026b.

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. In *NeurIPS*, 2020.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *EMNLP*, pp. 6449–6464. Association for Computational Linguistics, 2023a.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *ACL (1)*, pp. 12286–12312. Association for Computational Linguistics, 2023b.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL/IJCNLP (1)*, pp. 6691–6706. Association for Computational Linguistics, 2021.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. *CoRR*, abs/2401.08565, 2024a.

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. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024b.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *ICLR (Poster)*, 2016.

Sisuo Lyu, Siru Zhong, Tiegang Chen, Weilin Ruan, Qingxiang Liu, Taiqiang Lv, Qingsong Wen, Raymond Chi-Wing Wong, and Yuxuan Liang. Ts-memory: Plug-and-play memory for time series foundation models. *CoRR*, abs/2602.11550, 2026.

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 *ACL (1)*, pp. 9802–9822. Association for Computational Linguistics, 2023.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *UIST*, pp. 2:1–2:22. ACM, 2023.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In *NAACL (Short Papers)*, pp. 783–791. Association for Computational Linguistics, 2024.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *CoRR*, abs/2209.15189, 2022.

Krishna Srinivasan, Karthik Raman, Anupam Samanta, Lingrui Liao, Luca Bertelli, and Michael Bendersky. QUILL: query intent with large language models using retrieval augmentation and multi-stage distillation. In *EMNLP (Industry Track)*, pp. 492–501. Association for Computational Linguistics, 2022.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. Parametric retrieval augmented generation. In *SIGIR*, pp. 1240–1250. ACM, 2025.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. Instructretro: Instruction tuning post retrieval-augmented pretraining. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51255–51272. PMLR / OpenReview.net, 2024.

Jiarui Wang, Xiang Shi, Jiaqi Cao, Rubin Wei, Xiquan Wang, Hao Sun, Jingzhi Wang, Zhiqi Yang, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Memsft: Mitigating alignment tax with an external parametric memory. *CoRR*, abs/2607.25614, 2026a.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *NeurIPS*, 2023.

Zixuan Wang, Xingyu Dang, Rui-Jie Zhu, Zixin Wen, Hengyu Fu, Wenhao Chai, and Jason D. Lee. Learning what to remember: Test-time training via context distillation. *CoRR*, abs/2608.01672, 2026b.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. MLP memory: A retriever-pretrained memory for large language models. *CoRR*, abs/2508.01832, 2025.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Junming Zhang, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Memory decoder at scale: A pretrained, parametric long-term memory. *CoRR*, abs/2607.27919, 2026.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *ICLR*. OpenReview.net, 2022.

Frank F. Xu, Uri Alon, and Graham Neubig. Why do nearest neighbor language models work? In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38325–38341. PMLR, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pp. 2369–2380. Association for Computational Linguistics, 2018.

Zhixian Yang, Renliang Sun, and Xiaojun Wan. Nearest neighbor knowledge distillation for neural machine translation. In *NAACL-HLT*, pp. 5546–5556. Association for Computational Linguistics, 2022.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, pp. 12360–12371, 2019.

Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. OPSDL: on-policy self-distillation for long-context language models. *CoRR*, abs/2604.17535, 2026.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6):155:1–155:47, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *CoRR*, abs/2601.18734, 2026.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. Adaptive nearest neighbor machine translation. In *ACL/IJCNLP (2)*, pp. 368–374. Association for Computational Linguistics, 2021.

Zhiyuan Zhu, Yusheng Liao, Chenxin Xu, Yunfeng Guan, Yanfeng Wang, and Yu Wang. RA2FD: distilling faithfulness into efficient dialogue systems. In *EMNLP*, pp. 12304–12317. Association for Computational Linguistics, 2024.

A Implementation details

Memory input. In the Mistral backbone, h_t is the state after the attention residual connection and its following RMS normalization (Zhang & Sennrich, 2019). Base logits are computed separately through the remaining feed-forward computation, final normalization, and frozen vocabulary head.

Subset kNN teacher and Memory Decoder. The kNN teacher for the Wikipedia subset is built with the original teacher's builder. It reproduces the original position count and 26.967 GiB target cache. Memory Decoder uses a Qwen3-0.6B-Base body with the Mistral vocabulary (0.47B parameters).

Knowledge-QA configuration. For each dataset, we select λ for MLP Memory and Memory Decoder and γ for ResMem by the highest EM on the validation set. Each memory then uses its selected coefficient for all three metrics on the evaluation set: EM, F1, and PPL, with EM and F1 computed from the same generated answers. LoRA uses strength 0.25, chosen on development data. BM25-RAG retrieves five Wikipedia passages with a 4096-token input and up to 64 output tokens; the other methods use 384 input tokens and at most 12 output tokens. PPL scores answer tokens and excludes the prompt and EOS. On a separate Wikipedia sample, with coefficients selected on 128 calibration sequences and scored on 256 others, the QA checkpoints reach perplexity 7.441 (MLP Memory), 8.660 (ResMem), and 9.959 (Base). MLP Memory fits held-out text more closely, yet ResMem answers more questions correctly (Table 1). A closer fit to the corpus thus does not by itself give better final answers.

Hardware. The two memories trained on the subset's kNN teacher use A100-SXM4-80GB GPUs. Knowledge-QA memory sweeps use NVIDIA A100-SXM4-40GB GPUs. QA inference uses BF16 base computation, eager attention, batch size 16, and greedy decoding. The shared-teacher ablation and precision measurements use two eight-GPU A100-SXM4-80GB hosts, with one model worker per GPU. The precision study uses BF16 Base with FP32 or BF16 memory and SDPA/KV caching. Deployment timing uses one GPU. Detailed schedules, checkpoint sources, and scoring configurations are recorded in the repository's experimental-protocol note.

B Per-dataset knowledge-QA results

Table 3 lists the values behind Table 1. ResMem improves EM and F1 over Base on every dataset and has the lowest PPL on each. MLP Memory is slightly higher on WebQuestions EM and F1, the smallest dataset.

C Hallucination detection

Knowledge stored in the memory may also help the model judge whether a given answer is supported. HaluEval tests this with few-shot Yes/No hallucination detection. We reuse the QA checkpoints and method-specific coefficients without task-specific tuning; BM25-RAG supplies five Wikipedia passages. ResMem raises mean detection accuracy from 54.83 for Base to 63.09, compared with 60.34 for MLP Memory (Table 4). Its margin over MLP Memory is 7.00 points on QA and 1.48 on summarization, while MLP Memory is 0.22 points higher on dialogue.

D Precision and deployment cost

The construction saving leaves open how much computation the memory adds at inference and whether lower precision can reduce it. We load the knowledge-QA checkpoints with FP32 or BF16 memory, keeping a BF16 Base and fixed coefficients $\lambda = 0.25$ and $\gamma = 0.4$ for all datasets. Across all 38,627 questions, BF16 changes mean F1 by -0.0159 points for ResMem (95% paired question-cluster interval $[-0.0539, 0.0195]$), $+0.0227$ for MLP Memory ($[-0.0195, 0.0684]$), and $+0.0129$ for Memory Decoder ($[-0.0063, 0.0338]$).

BF16 loading lowers latency and memory (Table 5). At batch 16, BF16 ResMem reduces fixed-length total latency from 1,254 to 1,098 ms and raises decode throughput from 595 to 727 tokens/s. Its additional peak allocated memory over Base falls from 5.750 to 2.875 GiB. It adds 8.6% latency over Base, or 16.3% at batch one. Memory Decoder is smaller (0.47B parameters) and adds only

Table 3: Per-dataset knowledge-QA results behind Table 1, whose group values are means of these columns before rounding. Coefficients are selected per dataset (Appendix A). Bold marks the best value per column.

Method	NQ (n=3,610)	WebQ (n=2,032)	TriviaQA (n=11,313)	PopQA (n=14,267)	HotpotQA (n=7,405)
			EM ↑		
Base	18.31	16.88	54.71	22.01	17.64
+ BM25-RAG	15.87	7.48	50.28	22.65	18.51
+ LoRA	18.55	16.73	54.71	22.43	17.63
+ Memory Decoder	18.50	17.18	54.90	22.12	17.77
+ MLP Memory	18.59	**17.37**	55.26	22.38	18.30
+ ResMem (Ours)	**18.98**	17.32	**55.55**	**22.85**	**19.03**
			F1 ↑		
Base	24.52	30.77	60.24	25.15	23.47
+ BM25-RAG	21.89	16.76	56.95	25.86	26.27
+ LoRA	24.56	30.92	60.64	26.51	25.32
+ Memory Decoder	24.88	31.47	60.49	25.40	23.76
+ MLP Memory	25.11	**32.08**	61.05	25.83	24.52
+ ResMem (Ours)	**25.69**	31.94	**61.57**	**26.91**	**26.29**
			PPL ↓		
Base	8.46	10.57	14.31	11.96	6.27
+ BM25-RAG	7.88	9.14	13.05	12.29	5.88
+ LoRA	8.08	10.17	13.54	11.61	5.93
+ Memory Decoder	8.32	10.25	13.95	11.80	6.17
+ MLP Memory	8.07	9.54	13.69	11.07	6.02
+ ResMem (Ours)	**7.62**	**8.81**	**12.89**	**10.99**	**5.75**

Table 4: HaluEval hallucination-detection accuracy (%). Average weights the three tasks equally. Subscripts denote percentage-point changes from the base model; bold marks the column best.

Method	Dialogue	QA	Summarization	Average
Mistral-7B-v0.3	58.02	56.04	50.42	54.83
+ BM25-RAG	63.57$_{+5.55}$	52.75$_{-3.29}$	50.74$_{+0.32}$	55.69$_{+0.86}$
+ MLP Memory	**65.48**$_{+7.46}$	63.08$_{+7.04}$	52.45$_{+2.03}$	60.34$_{+5.51}$
+ ResMem (Ours)	65.26$_{+7.24}$	**70.08**$_{+14.04}$	**53.93**$_{+3.51}$	**63.09**$_{+8.26}$

1.01 GiB of peak memory in BF16, but it runs its own 28-layer Transformer with a KV cache at every step, adding 63.5% latency over Base measured on the same host. The timing experiment fixes 384 input and 32 output tokens with SDPA and KV caching.

Table 5: Precision and deployment cost for the knowledge-QA checkpoints at fixed $\lambda = 0.25$ (MLP Memory and Memory Decoder) and $\gamma = 0.4$ (ResMem) on every dataset; because Table 1 selects coefficients per dataset by EM and decodes with eager attention, its F1 values differ. Quality covers 38,627 questions with EOS-or-12-token decoding; timing uses batch 16, 384 input tokens, and exactly 32 output tokens on one A100-SXM4-80GB with BF16 Base, SDPA and KV caching. Times and peak allocated memory are medians over ten steady batches; decode throughput uses pooled generated tokens per second. Memory Decoder rows were measured on a second identical host, where Base takes 1,014 ms in total. These checkpoints and runtime differ from Table 2.

Method	Memory	F1	TTFT (ms)	Total (ms)	Decode (tok/s)	Peak (GiB)
Base	—	32.79	413.30	1011.22	829.19	14.942
MLP Memory	FP32	33.68	420.77	1252.34	596.23	20.692
MLP Memory	BF16	33.70	416.16	1099.91	725.03	17.817
ResMem	FP32	34.77	420.79	1254.14	594.72	20.692
ResMem	BF16	34.75	415.95	1097.76	726.77	17.817
Memory Decoder	FP32	33.07	830.74	2029.42	413.20	17.921
Memory Decoder	BF16	33.08	484.51	1658.11	422.42	15.955

E TOKENWISE GATE ABLATION

Table 6: Fixed-strength ResMem with and without the tokenwise gate. ResMem matches Table 1.

	ResMem		ResMem + gate	
Dataset	EM	F1	EM	F1
NQ	18.98	25.69	19.20	25.95
WebQ	17.32	31.94	17.13	31.65
TriviaQA	55.55	61.57	55.68	61.61
Open-domain mean	30.61	39.73	30.67	39.74
PopQA (long-tail)	22.85	26.91	22.91	26.81
HotpotQA (multi-hop)	19.03	26.29	19.15	26.30
Five-dataset mean	26.74	34.48	26.81	34.46

The gate decides when the memory should act on the base output. The residual matters only at decoding steps where it changes the base's top token. At such a step, it either repairs a base error or overrides a correct base token. The gate aims to keep the repairs and block the overrides, using only signals available during decoding.

At step t, we compute the base logits $z_{B,t}$ and the fused logits $z_{B,t} + \gamma s_{\theta,t}$. From both, we form 22 features. They describe each distribution (entropy, top-token probability, and top-two margin) and how the two top tokens compare, such as the residual's advantage for the fused token. They also cover the step index, the generated prefix, and whether either top token would end the answer. A logistic classifier maps the features to a score g_t. The step uses the fused logits if $g_t \geq 0.5$ and the base logits otherwise, so the residual scale is $\gamma_t \in \{0, \gamma\}$. The gate uses no labels or future tokens. On the test questions, the fused top token differs from the base top token at 19,254 of 463,520 decoding steps (4.2%). The gate accepts 10,614 of these changes and blocks 8,640. Of the accepted changes, 7,493 occur at the first answer token.

The gate is fitted on 264 NQ/TriviaQA training questions and selected on a disjoint 4,091-question development split, so it adds QA supervision beyond memory training. On all 38,627 questions, relative to fixed-strength ResMem, the gate changes mean EM by $+0.07$ points (95% question-cluster bootstrap interval $[-0.06, 0.19]$) and mean F1 by -0.02 points. It alters 190 answers, 113 for the better and 77 for the worse, and the interval for every dataset includes zero. Table 1 therefore reports ResMem without the gate.