R-NOTES: SLIDING-WINDOW LLMS CAN LEARN TO REMEMBER BEYOND THEIR ATTENTION WINDOW

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world LLM deployments increasingly require processing inputs of effectively unbounded length, calling for models that can handle arbitrarily long inputs with bounded memory. However, existing constant-memory architectures, such as linear attention and pure sliding window attention (SWA), lose information beyond their fixed state or window. To address this, we observe that an SWA model can preserve information much like a reader taking notes: brief self-written notes stay in the recent window and carry key content forward, even as the original tokens scroll out. Trained to do this consistently, pure SWA models already suffice for lengthy inputs. Building on this observation, we propose R-NOTES, which feeds fixed-size chunks of the input as a multi-turn conversation under a single SWA session and trains the model to emit brief notes that propagate key information across chunks within the window. Built on Qwen3-8B, R-NOTES matches or exceeds the 1.75x larger Qwen3-14B full-attention baseline on LongBench cross-task QA (73.4% vs 71.3%) and on the longest settings of RULER-HQA at 112K and RULER-SQuAD at 128K (87.50% vs 78.12% on both), while keeping the KV cache bounded by the window size and total compute linear in the number of processed chunks.

1 INTRODUCTION

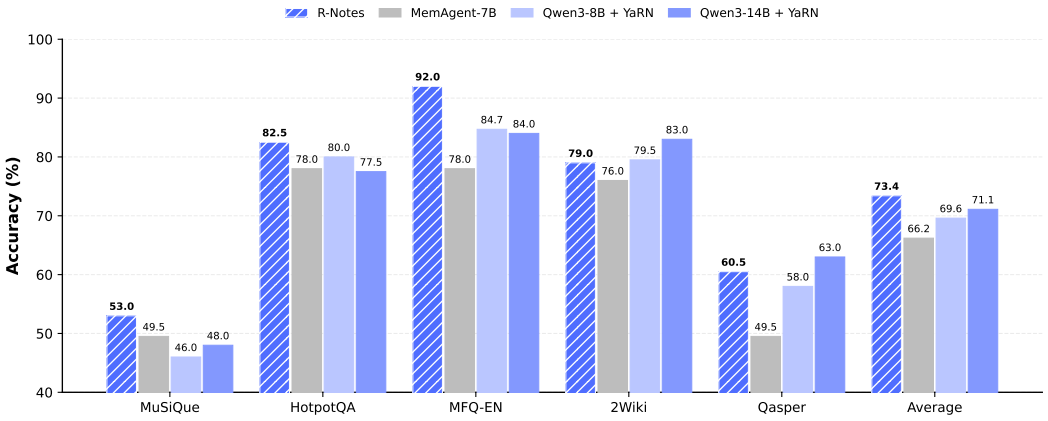

Figure 1: Cross-dataset generalization on the LongBench English QA tasks shown here. R-NOTES leads the shown-task average; Table 1 reports the full eight-task QA subset.

Real-world deployments demand ever stronger long-context capabilities from LLMs, including reading lengthy documents to answer user queries, sustaining infinite streaming dialogue, and, increasingly, executing long-horizon agent workflows (Jimenez et al., 2024; OpenAI, 2026; Google, 2025). These settings share a defining feature: the input has no a priori upper bound on length, and may continue to grow throughout deployment. The cost of supporting such inputs is non-trivial. Under full attention, the KV cache grows linearly with input length, and the cost of generating each new

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.

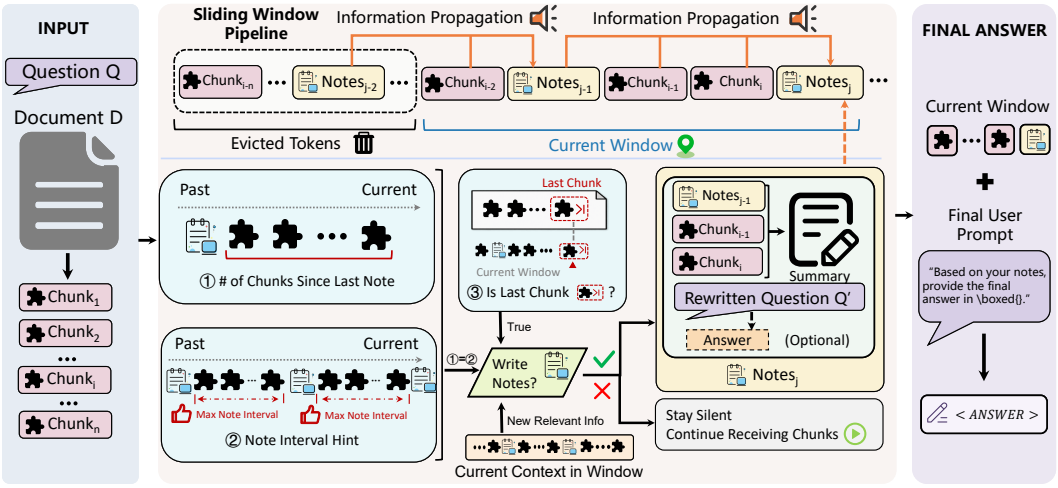

Figure 2: R-NOTES pipeline. The model processes a long input chunk by chunk under a sliding window of W tokens. After each chunk, the model adaptively decides whether to emit a note or stay silent; successive notes propagate the relevant content forward, so that information from chunks that have already left the window remains accessible through the most recent note. Generation terminates with a final answer.

token scales with all preceding tokens, so both memory and compute are unbounded (Pope et al., 2023). Ideally, we would like LLMs to handle arbitrarily long inputs with bounded memory, much like humans do.

Existing work toward bounded-memory long-context inference falls into two categories. *Architecture-level* approaches restrict each layer's attention to a fixed-size active state, such as linear attention (Peng et al., 2025) via a compressed hidden state or pure sliding window attention (SWA) (Fu et al., 2025) via a fixed-size window, but they inevitably lose information that falls outside this state. Hybrid designs interleave such layers with full-attention layers to recover long-range information (Gemma Team, 2025; NVIDIA, 2025), but these layers bring back a KV cache that grows with input length. *Agent-level* approaches instead leave the architecture unchanged and design external mechanisms to control what is preserved across chunks. Some manually trigger a summary whenever the context exceeds a preset window and re-inject the summary into the next chunk (Yu et al., 2025); others hand-design rules for which parts of the context to keep or discard (Liu et al., 2026; Ye et al., 2025). These works show that a language model can carry information across chunks, but rely on hand-crafted triggers or rules and keep the preserved information in external state outside the standard inference loop. This motivates combining the two strengths. We build on pure SWA for its bounded-memory inference, and train the model itself to actively preserve information, entirely within the standard inference loop and without manual triggers or external state.

Our key insight is that an SWA model can take notes like a reader. Brief notes stay inside the recent window and carry key content forward, even as the original tokens scroll out. Trained to do this consistently, pure SWA can handle inputs of effectively unbounded length while keeping memory bounded by the window.

We instantiate this idea as R-NOTES (RELAY NOTES), named for how each note relays task-relevant information to the next window. As illustrated in Figure 2, R-NOTES feeds the input to an SWA model as fixed-size chunks in a multi-turn conversation, and after each chunk the model adaptively decides whether to emit a <NOTES>...</NOTES> segment. As long as each new note is written before the previous one is evicted, information propagates along the chain $\mathcal{N}_1 \to \mathcal{N}_2 \to \cdots$, even though every input token eventually scrolls out of the window. We train the model in two stages: an SFT cold start on synthetic note-taking trajectories, followed by GRPO (Shao et al., 2024) with an outcome-only reward, from which the model learns what to extract from each chunk and what to carry over from earlier notes. Because all context management happens within a single SWA session, the KV cache is carried across turns rather than rebuilt for each chunk; R-NOTES thus

keeps it bounded by the window, grows total compute linearly with the number of chunks, and needs no runtime support beyond standard SWA inference.

Results. Built on Qwen3-8B (Team, 2025), R-NOTES reaches 73.4% average accuracy on LongBench (Bai et al., 2024) cross-task QA as shown in Figure 1, surpassing the 1.75× larger Qwen3-14B full-attention baseline at 71.3%. On the longest RULER (Hsieh et al., 2024) settings, 112K for HotpotQA and 128K for SQuAD, R-NOTES reaches 87.50% on both, while the baseline reaches 78.12%. These results support our argument that pure SWA models trained to actively preserve information can handle long inputs without hybrid layers.

2 RELATED WORK

Long-context LLMs. Deploying LLMs on long inputs requires both extending the trained context length and keeping inference cost manageable as the KV cache grows (Pope et al., 2023). Position extrapolation modifies the positional encoding so that pretrained models generalize beyond their training window (Peng et al., 2024; Ding et al., 2024; Chen et al., 2023; An et al., 2024). A second line of work redesigns the attention mechanism itself, either by interleaving local and global attention layers (Gemma Team, 2025; Meta AI, 2025) or by replacing softmax attention with fixed-state alternatives such as linear attention (Katharopoulos et al., 2020), RNNs (Peng et al., 2025), and state space models (Gu & Dao, 2023; Dao & Gu, 2024; Gu et al., 2022). Long-context post-training instead extends the usable window through additional training on long sequences (Yang et al., 2025; Wan et al., 2025). Finally, training-free schemes such as StreamingLLM (Xiao et al., 2024) adapt inference-time behavior without retraining the model.

Context management. A complementary line of work explicitly manages what information remains available as the context grows. Memory-augmented agents maintain external stores, including hierarchical memory, memory banks, and linked notes (Packer et al., 2023; Zhong et al., 2024; Kang et al., 2025; Xu et al., 2025; Chhikara et al., 2025), while methods such as Memory-R1 learn memory-update policies (Yan et al., 2025). Other approaches reconstruct or edit the active context: MemAgent (Yu et al., 2025) passes generated memory between separate chunk-level conversations; MEM1 (Zhou et al., 2025) consolidates memory and reasoning into a state that replaces earlier interaction history; AgentFold (Ye et al., 2025) selectively folds past interactions into summaries; and StateLM (Liu et al., 2026) uses explicit note-taking and context-deletion actions. Closer to our setting, Self-Notes (Lanchantin et al., 2023) interleaves generated notes with the input to support online reasoning and state tracking. However, it relies on full attention and decides at each token whether to insert a note. This requires sequential decoding over the entire input, precluding parallel prefill and limiting scalability to long inputs. In contrast, R-NOTES learns a chunk-level write-or-skip policy under pure SWA via SFT and reinforcement learning from final-answer correctness, writing notes only when worthwhile.

3 METHOD

R-NOTES trains an SWA model to write brief notes during chunk-by-chunk reading, keeping task-relevant information within the sliding window. Section 3.1 formalizes the conditions for the SWA model to preserve information across inputs longer than the window. Section 3.2 then describes the concrete implementation, including how the input is segmented and exposed to the model, and how and when notes are written.

3.1 PROBLEM SETUP AND ANALYSIS

SWA constraints. Let the input sequence be $D = (x_1, x_2, \ldots, x_L)$, where L denotes its length in tokens. Under SWA with window size W, when decoding at position t, the model attends only to tokens whose positions fall within $[\max(1,\, t - W + 1),\, t]$, so its KV cache contains at most W tokens regardless of L. Consequently, any token at position $p \leq t - W$ has already been evicted by decoding step t and receives zero attention weight. As input length grows, an increasing fraction of D thus becomes inaccessible through attention alone.

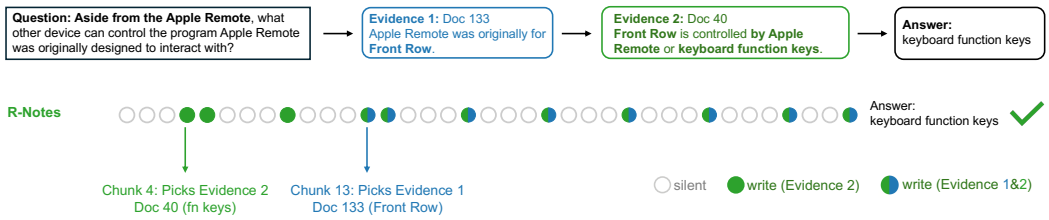

Figure 3: Note-emission trace on a long HotpotQA example whose answer requires chaining two pieces of evidence: "Apple Remote was originally for Front Row" (Doc 133) and "Front Row is controlled by keyboard function keys" (Doc 40). R-NOTES picks up Evidence 2 at chunk 4 and Evidence 1 at chunk 13, then carries both forward to answer *keyboard function keys*. Color of filled circles indicates which evidence each note carries; hollow circles indicate silent turns.

This limitation underlies the common view that pure SWA is insufficient for long inputs, reflected in hybrid designs that interleave SWA with full-attention layers (Gemma Team, 2025; Meta AI, 2025). This view, however, treats SWA purely as an attention pattern over a fixed input, and does not consider what the model's own generation can contribute. We argue below that the model can use its own generation to carry critical information forward before it is evicted, allowing pure SWA to handle inputs of effectively unbounded length.

Unbounded preservation under SWA. We now make this concrete: how can the model retain information through its own generation before the window evicts it? A *note* is a short segment generated by the model itself, intended to summarize task-relevant content seen so far. Let $\mathcal{N}_i$ denote the i-th note and L_{notes} an upper bound on the length of any single note. For the analysis, we divide D into chunks of C tokens and assume that consecutive notes are separated by at most n chunks. Successive notes form a chain $\mathcal{N}_1 \to \mathcal{N}_2 \to \cdots$, where each $\mathcal{N}_{i+1}$ is expected to inherit task-relevant content from $\mathcal{N}_i$ and incorporate new content from the intervening chunks.

For this chain to actually carry information forward, two conditions must hold at every step. The first, the *refresh condition*, bounds the interval between consecutive notes: $\mathcal{N}_i$ must still be in the window when $\mathcal{N}_{i+1}$ is being written; otherwise, the chain is broken. Since the window holds at most W tokens and must simultaneously accommodate $\mathcal{N}_i$ at its left edge, the in-progress $\mathcal{N}_{i+1}$ at its right edge, and the nC tokens of intervening chunks, this requires

$$nC + 2L_{\text{notes}} \leq W. \tag{1}$$

The second, the *coverage condition*, concerns what each note contains: when $\mathcal{N}_{i+1}$ is written, it must record all task-relevant information seen so far, both what is carried over from $\mathcal{N}_i$ and what is newly introduced in the intervening chunks. The refresh condition alone only keeps $\mathcal{N}_i$ alive in the window; whether the relevant content survives the chain depends on what $\mathcal{N}_{i+1}$ chooses to write down.

When both conditions hold at every step, information propagates along the chain $\mathcal{N}_1 \to \mathcal{N}_2 \to \cdots$ no matter how long D is. In practice, we do not enforce n as a hard constraint: the prompt only provides it as a suggested interval, and the model is trained, with final-answer correctness as the only reward and no supervision on note content, to write notes at appropriate moments and to copy forward whatever the answer requires. The training procedure is given in §4.1, and Appendix D.2 reports empirical note lengths showing that the default schedule satisfies the refresh condition.

3.2 R-NOTES WITH SWA

We divide the input document D into fixed-length chunks of C tokens and present them to the model as a multi-turn conversation under a single SWA inference session; by default $C = 1600$, $n = 4$, and $W = 12$K, the same configuration as in training. The query Q is placed in the first user message alongside the first chunk. Once L exceeds W, this first message is evicted, and the query with it; each note is therefore required to begin by restating the query, so that the question, like the evidence, survives only through the note chain. After all chunks have been processed, a final user message

asks the model to answer from its notes, wrapped in `\boxed{}`: everything available at that point, the question included, lives in the token stream inside the window. Full prompt templates are given in Appendix A.

On each turn, the model may emit a segment wrapped in `<NOTES>...</NOTES>` as part of its assistant reply. To make timing observable to the model, each user message carries three meta-fields:

- `chunks_since_notes`: the number of chunks since the previous note.
- `notes_interval_hint`: a suggested maximum interval n.
- `is_last_section`: whether the current chunk is the last one.

These fields are signals, not commands: whether to write a note on a given turn is left to the model. We train the model to write a note when the suggested interval is reached, when new task-relevant content appears, or when the last chunk is being processed, and to remain silent otherwise (an empty assistant reply).

As shown in Figure 3, with this protocol and the training of §4.1, the model satisfies both conditions of §3.1: its note spacing respects the refresh condition (Eq. 1), and its note content realizes the coverage condition, so the chain $\mathcal{N}_1 \to \mathcal{N}_2 \to \cdots$ actually emerges. The persistent two-color band starting at chunk 13 is what this chain looks like on real input: once Evidence 1, one of the two facts the answer requires, enters a note, it survives across every later note and is still available at the final turn, even though every input token from chunk 13 has long been evicted from the window.

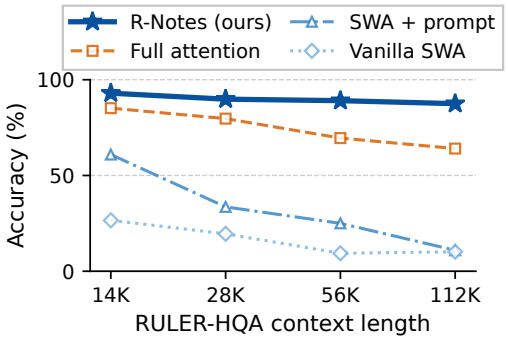

Figure 4: RULER-HQA accuracy across input lengths. Training turns the notes pipeline from an unstable prompt into a long-context policy close to the full-attention reference.

The trace of Figure 3 confirms the chain on a single example. Figure 4 shows that this is enough to more than close the gap between SWA and full attention on RULER-HQA, a multi-document QA task with inputs up to 112K tokens (§4.2): across document lengths, vanilla SWA collapses, the notes prompt without training (SWA + prompt) is not enough, and R-NOTES surpasses the full-attention reference at every length, with the margin widening as inputs grow.

4 EXPERIMENTS

4.1 SETUP

Training. We build R-NOTES on Qwen3-8B (Team, 2025), applying sliding window attention with $W = 12$K to all layers during both training and inference. Training proceeds in two stages: (i) an SFT cold start and (ii) reinforcement learning with GRPO (Shao et al., 2024). Both stages use the chunked multi-turn protocol of Section 3.2, identical to the one used at inference. For SFT, we distill 3,773 note-taking trajectories from a gpt-5 teacher on HotpotQA training questions, retaining only those that reach the correct answer. The RL stage is initialized from the SFT checkpoint and uses final-answer correctness as the only reward, with no supervision on note content. All training inputs in both stages are HotpotQA examples of at most 32K tokens, so all evaluation settings beyond 32K are extrapolations beyond the training length. Additional training details and hyperparameters are in Appendix B.

Evaluation. We evaluate along three axes. First, *cross-task generalization* is measured on the 8-task QA subset of LongBench (Bai et al., 2024), comprising 6 English and 2 Chinese tasks, without any task-specific fine-tuning. Second, *length generalization* is measured on inputs far longer than those seen in training. We use two QA tasks from RULER (Hsieh et al., 2024): RULER-HQA, a

Table 1: Cross-task accuracy (%) on the LongBench QA subset. **Bold** marks the best result.

Method	NarrativeQA	HotpotQA	2Wiki	MuSiQue	Qasper	MFQ-EN	MFQ-ZH	DuReader	Avg.
Sliding window attention									
Qwen3-8B	24.0	52.5	74.0	17.5	60.0	82.0	84.5	62.0	57.1
Full attention with YaRN									
Qwen3-8B	52.0	80.0	79.5	46.0	58.0	84.7	84.0	72.5	69.6
Qwen3-14B	**56.0**	77.5	**83.0**	48.0	**63.0**	84.0	84.5	74.5	71.3
External-state method									
MemAgent-7B	47.5	78.0	76.0	49.5	49.5	78.0	86.0	73.0	67.2
Ours									
R-NOTES	49.5	**82.5**	79.0	**53.0**	60.5	**92.0**	**92.0**	**79.0**	**73.4**

multi-document task built on HotpotQA, from 14K to 112K tokens, and RULER-SQuAD, a single-document task built on SQuAD, from 16K to 128K tokens, reaching up to 4× the 32K training length. As an extreme case, we test retrieval on RULER NIAH with three single-needle and three multi-key tasks at five lengths from 32K to 512K, i.e., up to 16× the training length. Third, *inference efficiency* is measured by KV-cache memory and inference FLOPs from 8K to 1M tokens.

All QA tasks share one scoring pipeline: a prediction is counted correct if it matches the gold answer after normalization, and otherwise gpt-5-mini judges its correctness, which avoids penalizing correct free-form answers phrased differently from the reference. NIAH uses the official rule-based `string_match_all` metric. Unless otherwise specified, R-NOTES runs with $W = 12$K, chunk size $C = 1600$, and note-interval hint $n = 4$, the same configuration as in training. Since LongBench-HotpotQA and RULER-HQA are drawn from the HotpotQA dev split, we verify that they share no questions with our training set. Additional details, including dataset sizes, the overlap audit, and baseline generation settings, are in Appendix C.

Compared models. We compare against three groups of baselines that differ in how they handle long inputs. The first group uses Qwen3-8B under the same SWA window ($W = 12$K) to isolate the effect of training: the model either reads the whole input in a single turn, or follows our chunked multi-turn protocol without any training (R-NOTES w/o training, reported in the ablations). The second group uses full attention. Qwen3-8B and Qwen3-14B are extended to 128K with YaRN (Peng et al., 2024); Qwen3-8B serves as the same-backbone reference without the window constraint. We also include QwenLong-L1-32B (Wan et al., 2025), which is trained with RL for long-context reasoning. The third group is MemAgent-7B (Yu et al., 2025), a comparable-size agent-level method that carries a fixed-length memory across chunks through separate model calls. For the efficiency analysis, we additionally compare against two hybrid architectures that pair constant-memory layers with full-attention layers: Gemma-3-12B (Gemma Team, 2025), which interleaves local sliding-window layers with global attention layers, and Nemotron-H-8B (NVIDIA, 2025), which interleaves Mamba-2 layers with full-attention layers.

4.2 MAIN RESULTS

R-NOTES matches or exceeds the accuracy of full-attention baselines of equal or larger size. Meanwhile, its KV cache remains bounded by the fixed window, and its total compute grows linearly with the number of processed chunks.

Cross-Dataset Generalization Although trained only on HotpotQA, R-NOTES achieves an average accuracy of 73.4% on LongBench, surpassing the 1.75x larger Qwen3-14B full-attention baseline at 71.3%. R-NOTES is the best method on 5 of the 8 tasks, with especially clear gains on MFQ-EN and the two Chinese QA tasks, MFQ-ZH and DuReader, and pulls further ahead of the comparable-size MemAgent baseline. It trails full attention mainly on NarrativeQA and Qasper, whose answers rely on implicit narrative evidence or long enumerations that are harder to carry in brief notes (Section 4.4). These results indicate that the note-taking capability instilled by our two-stage training on English multi-hop QA transfers across tasks and languages under a fixed sliding window. As shown in Appendix D.5, the transfer also extends beyond QA: R-NOTES improves

Table 2: Length generalization accuracy (%) on **RULER-HQA** and **RULER-SQuAD**. **Bold** marks the best result in each column.

Method	RULER-HQA					RULER-SQuAD				
	14K	28K	56K	112K	Avg.	16K	32K	64K	128K	Avg.
Sliding window attention										
Qwen3-8B	26.56	19.53	9.38	10.16	16.41	15.62	14.84	16.41	10.94	14.45
Full attention with YaRN										
Qwen3-8B	85.16	79.69	69.53	64.06	74.61	82.81	78.91	68.75	69.53	75.00
Qwen3-14B	**93.75**	**89.84**	79.69	78.12	85.35	82.81	78.91	80.47	78.12	80.08
External-state method										
MemAgent-7B	87.50	86.72	84.38	85.94	86.14	85.94	87.50	82.81	82.81	84.77
Ours										
R-NOTES	92.97	**89.84**	**89.06**	**87.50**	**89.84**	**90.62**	**91.41**	**83.59**	**87.50**	**88.28**

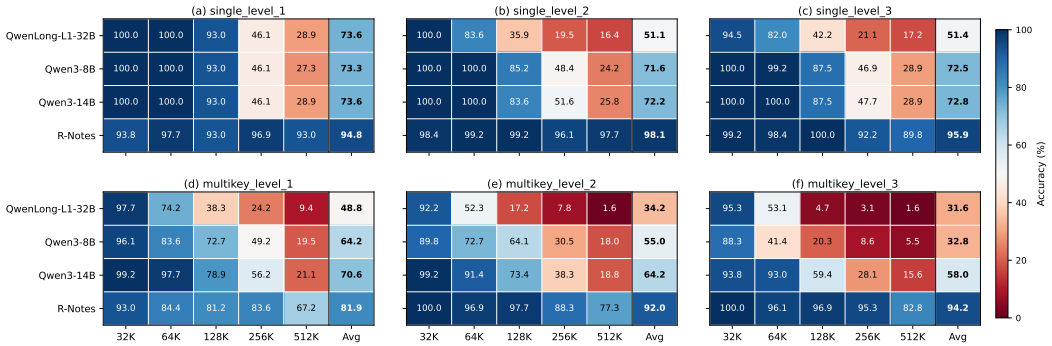

Figure 5: Per-task RULER NIAH retrieval accuracy across context lengths. R-NOTES preserves retrieval performance at the longest contexts while full-attention baselines degrade sharply.

over Qwen3-8B on all three non-QA LongBench categories, namely summarization, few-shot learning, and synthetic tasks, and outperforms Qwen3-32B and SearchR1-32B on the BrowseComp-Plus deep-research agent task. Meanwhile, training leaves general capabilities intact: short-context MMLU and GSM8K scores are unchanged, and the model retains its mathematical reasoning while carrying notes (Appendix D.4).

Context Length Scaling R-NOTES achieves the best average on both benchmarks and is strongest at the longest settings as Table 2 shows. The full-attention baselines degrade markedly as input length grows, while R-NOTES remains stable through the longest settings, even though all training inputs are within 32K tokens. This stability is what the argument of §3.1 predicts: both conditions are local to each note-to-note step and independent of total input length, so a policy that satisfies them at 32K satisfies them at any length.

As an extreme case of length scaling, we evaluate RULER NIAH at up to 512K tokens, 16× the training length. R-NOTES achieves the highest average retrieval accuracy among all methods in Figure 5. Notably, R-NOTES also outperforms QwenLong-L1-32B, a larger model trained with long-context RL, indicating that its gains are not merely attributable to model scale or long-context training. A per-task view further reveals when the differences emerge. On the simplest single-key task, the four methods are hard to distinguish at moderate lengths; the gap only opens up at longer lengths, where full-attention baselines drop sharply while R-NOTES maintains high accuracy. The harder the task, the earlier the divergence; on the hardest multikey task, QwenLong-L1-32B is already close to failure at moderate lengths. Overall, R-NOTES remains robust as retrieval becomes longer and harder, while competing long-context baselines degrade substantially.

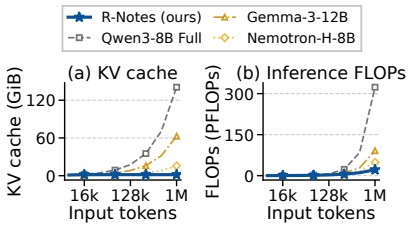

Figure 6: Inference efficiency of R-NOTES, Qwen3-8B full attention, Gemma-3-12B, and Nemotron-H-8B. Batch size is 1 and KV caches use BF16.



Figure 7: Training-stage ablation. Left: accuracy by training stage, where RULER results are averaged over the evaluated lengths. Right: RULER-HQA accuracy by length.

Table 3: Inference schedule robustness on RULER-HQA (%). **Bold**: our default setting (C=1600, n=4).

Chunk size C	**Note interval** n 3	4	5	6
800	85.74	88.09	87.30	86.72
1600	85.74	**89.84**	88.48	88.28
3200	88.28	*violates Eq. 1*		

Table 4: Window-size sensitivity on RULER-HQA (%). †With C=256, n=4.

Window	**14K**	**28K**	**56K**	**112K**	**Avg.**
SW-2K†	59.4	57.0	50.8	42.2	52.3
SW-8K	89.1	90.6	83.6	83.6	86.7
SW-12K	**93.0**	89.8	**89.1**	**87.5**	**89.8**
SW-16K	93.0	89.1	89.1	87.5	89.7

Efficiency Analysis As shown in Figure 6, R-NOTES's KV cache stays constant at the window size, and its inference FLOPs grow linearly with input length. Full attention instead has a KV cache linear in input length and attention FLOPs quadratic in it. Gemma-3 and Nemotron-H keep a few full-attention layers, so their KV caches still grow linearly and their FLOPs retain a quadratic term.

4.3 ABLATION STUDIES

We study four design questions for R-NOTES: the role of each training stage, sensitivity to inference-time chunk size and note interval, sensitivity to the sliding-window size, and adaptive versus fixed-interval triggering. All ablations use the same model and training data as the main results.

Training stages. We first ask whether the notes pipeline works without training. Figure 7 (left) shows that it helps only partially: although R-NOTES w/o training improves over single-turn SWA, it remains far below the trained variants. SFT is the main step that turns the prompt into a usable note-taking policy under the fixed sliding window. RL then refines this policy for harder long-context cases. As shown in Figure 7 (right), the RL checkpoint improves over SFT at every RULER-HQA length, and the gain becomes larger as the context grows. While SFT accuracy drops steadily beyond the 32K training length, RL keeps it nearly flat.

Inference schedule robustness. We further ask whether the learned note-taking behavior depends on the exact chunking and interval schedule used during inference. Using the same RL checkpoint, we vary the chunk size C and note interval n; the refresh condition (Eq. 1, with L_{notes} from Appendix D.2) delimits the feasible schedules, and combinations that violate it are marked in Table 3. Within the feasible region, the default schedule $C = 1600$, $n = 4$ is the strongest setting, but nearby schedules retain competitive accuracy, indicating that training learns a robust note-writing policy rather than overfitting to a single chunk boundary or interval hint.

Window size. Table 4 evaluates the same RL checkpoint with different SWA windows on RULER-HQA. Reducing W to 8K lowers the average by 3.1 points. At $W = 2$K accuracy collapses even though the refresh condition still holds (with $C = 256$, $nC + 2L_{\text{notes}} \approx 1.3$K): a 112K input now takes roughly 6× more note-to-note hops, and coverage losses at each hop compound. The two conditions of §3.1 thus fail separately: refresh keeps the chain alive, while a too-small window

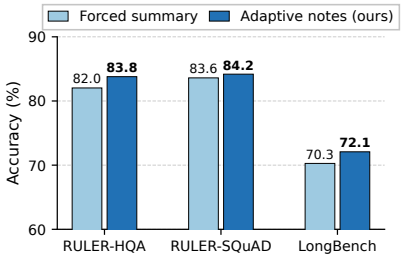

Figure 8: Adaptive vs. forced triggering.

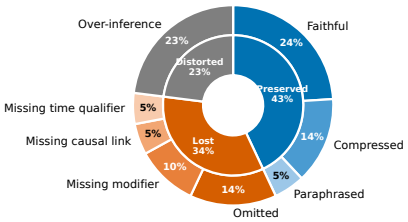

Figure 9: How answer-bearing facts are represented in notes.

overloads coverage. Increasing W to 16K provides no gain, so $W = 12$K is a practical operating point balancing coverage load and attention cost.

Trigger choice. Finally, we ask whether letting the model decide when to write notes outperforms forcing a summary at fixed intervals. For each protocol, we generate trajectories from the same question set using the same teacher model, then train a checkpoint with identical SFT hyperparameters. Only the triggering protocol differs (Appendix D.1). As shown in Figure 8, adaptive triggering outperforms forced summarization on all three benchmark averages. A fixed-interval schedule forces a note on every chunk, including those with no relevant information, diluting the useful content in each note, whereas adaptive triggering leaves such chunks silent. The largest gain on LongBench suggests that adaptive triggering also generalizes better outside the training distribution, so we adopt it in the main method.

4.4 Analysis

Lossiness and failure modes. To understand where R-Notes still fails, we examine what information is lost during note propagation. On NarrativeQA, Qasper, and MuSiQue, the three LongBench tasks where R-Notes scores lowest, we classify how ground-truth facts are represented in the notes, using three tiers. A fact is *preserved* if the note records it faithfully, in compressed form, or as a paraphrase with equivalent meaning; *lost* if the note omits it entirely or drops a key modifier, causal link, or temporal qualifier; and *distorted* if the note over-infers, adding content not supported by the source. Overall, 43% are preserved, 34% are lost, and 23% are distorted (Figure 9). The dominant error differs by task: on NarrativeQA, implicit narrative evidence is never written into a note (lost); on Qasper, notes over-enumerate methods instead of isolating the queried one (distorted); and on MuSiQue, notes drop the modifiers or temporal qualifiers needed for multi-hop selection (lost). Notably, nearly all losses occur when writing notes rather than when carrying them forward: once a fact is captured, it survives propagation with over 92% conditional retention (Appendix D.3).

5 Conclusion

We presented R-Notes, a training framework that lets pure SWA models process effectively unbounded inputs by writing brief self-generated notes that remain inside the recent attention window and relay task-relevant information across chunks. Unlike external-state agents or hand-designed compression rules, R-Notes keeps context management inside the standard SWA inference loop, and learns through an SFT cold start followed by outcome-only GRPO. Built on Qwen3-8B, R-Notes reaches 73.4% on LongBench cross-task QA, surpassing the 1.75x larger Qwen3-14B full-attention baseline, and reaches 87.50% on both RULER-HQA at 112K and RULER-SQuAD at 128K, while keeping the KV cache bounded by the window and total compute linear in the number of chunks. These results support a simple conclusion: with the right training signal, sliding-window LLMs can learn to remember beyond their attention window without architectural changes or external memory.

AI USE STATEMENT

In this work, we used generative artificial intelligence tools to generate synthetic training data, assist with translation, clean and reformat datasets, and interpret results. Specifically, gpt-5 generated note-taking teacher trajectories, and gpt-5-mini was used to filter these trajectories and judge model answers during evaluation. We have reviewed all AI-assisted work. For example, we reviewed, edited, and verified assistant-generated code and manuscript text before including them in the paper or experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A gives the full prompt templates and the LLM-judge prompt. Appendix B details training data construction and the SFT and RL hyperparameters (Tables 5 and 6). Appendix C specifies the scoring pipeline, the train–test overlap audit, and baseline generation settings (Table 7). We did not run a broad automated hyperparameter search; the default schedule was selected from small manual sweeps, with robustness across schedules reported in Table 3.

REFERENCES

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 1493–1510. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/an24b.html.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595, 2023. doi: 10.48550/ARXIV.2306.15595. URL https://doi.org/10.48550/arXiv.2306.15595.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 10041–10071. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending LLM context window beyond 2 million tokens. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 11091–11104. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/ding24i.html.

Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. Sliding window attention training for efficient large language models. *arXiv preprint arXiv:2502.18845*, 2025.

Gemma Team. Gemma 3 technical report, 2025. arXiv preprint arXiv:2503.19786.

Google. Deep research is now available on gemini 2.5 pro experimental., 2025. URL https://blog.google/products-and-platforms/products/gemini/deep-research-gemini-2-5-pro-experimental/.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023. doi: 10.48550/ARXIV.2312.00752. URL https://doi.org/10.48550/arXiv.2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pp. 5156–5165. PMLR, 2020. URL http://proceedings.mlr.press/v119/katharopoulos20a.html.

Jack Lanchantin, Shubham Toshniwal, Jason Weston, Arthur Szlam, and Sainbayar Sukhbaatar. Learning to reason and memorize with self-notes. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Xiaoyuan Liu, Tian Liang, Dongyang Ma, Deyu Zhou, Haitao Mi, Pinjia He, and Yan Wang. The pensieve paradigm: Stateful language models mastering their own context, 2026. arXiv preprint arXiv:2602.12108.

Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, April 2025.

NVIDIA. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models, 2025. arXiv preprint arXiv:2504.03624.

OpenAI. Deep research system card, 2026. URL https://openai.com/index/deep-research-system-card/.

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. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schutze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *ArXiv*, abs/2508.19828, 2025. URL https://api.semanticscholar.org/CorpusID:280918480.

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. *CoRR*, abs/2501.15383, 2025. doi: 10.48550/ARXIV.2501.15383. URL https://doi.org/10.48550/arXiv.2501.15383.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context management, 2025. arXiv preprint arXiv:2510.24699.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 2025. arXiv preprint arXiv:2507.02259.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents, 2025. URL https://arxiv.org/abs/2506.15841.

A PROMPT TEMPLATES

We use no system message. The document is split into token chunks, and each chunk is sent as one user turn in the same SWA inference session. The placeholders {prompt} and {chunk} are filled at runtime. The scalar meta-fields expose the current position in the stream but do not force a note update.

First chunk prompt

```
You are presented with a problem and the
first section of an article. Please read
and extract ONLY information that is directly
relevant to answering the problem.

<problem>
{prompt}
</problem>

<section>
{chunk}
</section>

Meta: chunks_since_notes={chunks_since_notes},
notes_interval_hint={notes_interval_hint},
is_last_section={is_last_section}.

Output NOTHING unless you should write notes.
Write notes if:
- this section adds new info relevant to the
  problem, OR
- chunks_since_notes >= notes_interval_hint
  (refresh so notes do not fall out of window), OR
- is_last_section=true
  (must output non-empty notes).

If you decide to write notes, follow format
and rules:
Note format:
<NOTES>
...updated running notes...
</NOTES>

Notes rules:
- MUST start by restating the original
  problem/question.
- extract ONLY information that is directly
  relevant to answering the problem.
- If facts answer the question, state:
  "Answer: [the answer]".

Output:
```

Subsequent chunk prompt

```
Here is another section of the article.
Please continue extracting:

<section>
{chunk}
</section>

[Meta line, note-writing conditions, note format,
 and notes rules identical to the first chunk
 prompt.]

Output:
```

Final-answer prompt

```
Based on your previous notes, provide the
final answer in \boxed{}.

INSTRUCTIONS:
- If the notes contain facts that answer the
  question,
  extract and provide the answer in \boxed{answer}
- You may need to combine information from
  multiple documents - this is expected
- Only return empty \boxed{} if the notes
  explicitly states no relevant information was found
- Be concise: only include the direct answer,
  not explanations

Your answer:
```

For answer scoring, we first apply exact match after normalization. If the rule is inconclusive for a sample, we call gpt-5-mini with temperature 0 using a single user message and no system message. The judge prompt is:

LLM judge prompt

```
You are an expert short-answer QA judge. Decide whether the predicted answer should be counted correct
 for the given question.

Question:
{question}

Ground-truth answer(s):
{ground_truth}

Predicted answer:
{prediction}

How to judge:
1. Read the question and identify the requested answer slot (what is being asked: month, year, date,
city, country, person, organization, number, song title, award name, etc.).
2. Use the ground-truth answer(s) as the reference for the correct fact -- not as a literal string
that must be copied word-for-word.
3. Mark correct when the prediction gives the same correct fact as the ground truth for the requested
slot, even if the ground truth includes extra words the question did not require (for example:
question asks "what month", ground truth "January 2001", prediction "January" -> correct; question
asks "what city", ground truth "Caistor is a town", prediction "Caistor" -> correct).
4. Allow harmless formatting differences: capitalization, punctuation, articles, pluralization,
aliases, abbreviations, numeric word/digit forms, and equivalent date/quantity phrasing. Ignore
wrappers such as \boxed{{...}}, "Answer:", and reasoning when the final answer is clear.
5. For yes/no questions, treat LaTeX-wrapped answers like \text{{Yes}} / \text{{No}} as equivalent to
Yes / No.
6. If all ground-truth answers are "Unanswerable" (any casing) or clearly mean the text does not
support an answer, mark correct when the prediction abstains in good faith and does not assert a
contradictory specific fact.
7. Mark incorrect when the prediction is missing or empty; answers the wrong slot (for example song
title when artist is asked, person name when tournament is asked); contradicts the ground-truth fact;
gives multiple incompatible candidates without choosing one; or is too vague/generic when the ground
truth is specific.
8. Do not use outside knowledge. Do not re-solve the question from scratch -- only compare whether the
 prediction satisfies the question's slot and matches the ground-truth fact.

Return ONLY valid JSON:
{"correct": true or false, "reason": "short reason"}
```

B Training and Implementation Details

Streaming execution. At inference time, R-NOTES keeps the full message history in a single chat session and relies on the model's sliding-window KV eviction, rather than manually restarting the conversation. The final prompt does not receive a separate memory variable; it can only use tokens that remain reachable through the current SWA window. When the model emits a non-empty <NOTES> block, we reset `chunks_since_notes` to zero for the next chunk. Otherwise the counter increments by one.

Model and data. We use Qwen3-8B as the base model and run both training and evaluation under pure sliding window attention with window size $W = 12$K and chunk size $C = 1600$. The note-interval hint is implemented via `NOTES_INTERVAL=3`, corresponding to $n = 4$ in the notation of Section 3.1. The difference is only an indexing convention: the implementation counter is reset after emitting notes and counts the chunks since that note, whereas n denotes the full note-to-note span. The hint is exposed to the model as a visible signal rather than a hard constraint.

The SFT data consists of 3,773 correct teacher trajectories collected on HotpotQA, split into 3,584 training examples and 189 validation examples. We generate these trajectories with a full-attention gpt-5 teacher and use a gpt-5-mini judge for filtering. During teacher-data construction, we manually trim the message history across turns so that the teacher only sees the retained notes together with the recent chunk sequence, matching the intended note-propagation pattern without relying on native SWA at generation time. All training inputs are within 32K tokens; the longer contexts reported in Section 4.2 are evaluation-time extrapolations under the same fixed $W = 12$K window.

SFT and RL hyperparameters. Our training uses a two-stage setup: an SFT cold start, followed by GRPO fine-tuning initialized from the SFT checkpoint. The RL stage uses final-answer correctness as the only reward, with no supervision applied to note content. Tables 5 and 6 list the main optimization hyperparameters for the two stages.

Table 5: SFT hyperparameters.

Hyperparameter	Value
Epochs	3
Train batch size	32
Per-device batch size	1
Gradient accumulation	4
Learning rate	1×10^{-5}
LR schedule	Cosine, warmup ratio 0.03
Weight decay	0.01

Table 6: RL (GRPO) hyperparameters.

Hyperparameter	Value
Train batch size	16
Rollouts per prompt	8
Temperature	0.7
Top-p	1.0
Top-k	-1
Per-turn generation cap	2,048
Actor learning rate	1×10^{-6}
KL coefficient	0.001, low-variance

C Evaluation Details

All trained-checkpoint evaluations keep the default Qwen3 chat template. For HTTP rollout, we use vLLM with `VLLM_USE_V1=1`, and route each sample consistently to the same server across all chunk turns so that prefix caching remains effective.

For RULER-HQA and RULER-SQuAD, each reported cell uses 128 examples and is evaluated with the QA scoring pipeline described in Appendix A. For LongBench, we use 200 examples for each task except MFQ-EN, for which we use 150, and score all tasks with the same QA pipeline. For RULER NIAH, we use the official `string_match_all` metric over the six `niah_single_1/2/3` and `niah_multikey_1/2/3` tasks.

Our HotpotQA training data is drawn from the official HotpotQA train split, while LongBench HotpotQA and RULER-HQA are drawn from the official HotpotQA dev split. We additionally audit question overlap by lowercasing each question, removing punctuation and articles, and collapsing whitespace, and find zero overlap between the training questions and either evaluation set.

Table 7 summarizes the evaluation-time context and generation configuration of the baseline methods.

D Additional Results

D.1 Additional Ablations

Controlled setup for trigger choice. For the trigger-choice ablation in Figure 8, the forced-summary and adaptive-notes SFT checkpoints use the same 3,773 HotpotQA questions and the

Table 7: Evaluation-time generation configurations for baseline methods. For full-attention baselines, context length is the configured single-call maximum context. For MemAgent, it is the recurrent per-call window used at each chunk step; longer documents are processed across multiple recurrent calls. Input and output tokens are per model call.

Method	Context Length	Input Tokens	Output Tokens
Qwen3-8B/14B + YaRN	128K	120K	8K
QwenLong-L1-32B	128K	120K	8K
MemAgent-7B	8K/call	5K/chunk	1K memory

same teacher, with trajectories generated under their respective prompts. Both checkpoints are then trained with the same SFT optimization setting. The comparison therefore controls for the question set, teacher model, data scale, and SFT hyperparameters, while varying the triggering protocol.

D.2 NOTE STATISTICS

Table 8: Length of emitted notes for R-NOTES on the RULER-HQA runs used in Table 2. **Tokens/note** reports mean $\pm$ standard deviation over emitted notes. **P90** is the 90th percentile of emitted-note length.

RULER-HQA length	Runs	Examples	Notes/ex.	Emit rate	Tokens/note	P90	Tokens/ex.
14K	3	384	4.54	49.2%	90.0 ± 38.8	129	408
28K	3	384	7.14	39.7%	90.2 ± 33.7	135	644
56K	3	384	12.18	34.3%	98.4 ± 55.3	151	1199
112K	3	384	21.83	30.9%	99.6 ± 88.8	152	2175
All	12	1536	11.42	34.3%	96.9 ± 70.2	145	1107

Note length. To characterize the actual size of the notes, we measure R-NOTES on the RULER-HQA runs used in Table 2. As shown in Table 8, notes are short: across 1,536 examples and 17,546 emitted notes, the mean emitted-note length is 96.9 tokens and the 90th percentile is 145 tokens. These empirical lengths are well below the fixed $W = 12$K window; under the default schedule, $C = 1600$ and $n = 4$, using the 90th-percentile note length gives $nC + 2L_{\text{notes}} \approx 6.7$K tokens, comfortably satisfying the refresh condition (Eq. 1).

Global compression ratio. We measure the global compression ratio as the number of tokenizer tokens inside all emitted <NOTES> blocks divided by the input-context length. On RULER-HQA, the ratio decreases from 3.76% at 7K to 1.23% at 112K (Table 9). Thus, the total note stream grows more slowly than the input context.

Information density. We use the number of bullet items in each note as a proxy for atomic information density. The density is task-dependent: RULER-HQA notes contain about 1.7 items on average, whereas Qasper notes contain 4.4 items because its answers more often require lists. This pattern suggests query-adapted selection rather than indiscriminate summarization (Table 10).

Table 9: Global compression ratio on RULER-HQA.

Input length	7K	14K	28K	56K	112K
Compression ratio	3.76%	2.06%	1.61%	1.49%	1.23%

D.3 CAPTURE AND RETAIN ANALYSIS

To understand how training makes the notes mechanism effective, we analyze note propagation through two complementary abilities. *Capture* is the ability to write answer-bearing evidence into a <NOTES> block before it leaves the sliding window, while *Retain* is the ability to carry captured

Table 10: Number of bullet items per emitted note, used as a proxy for information density.

Dataset	Number of notes	Items per note
RULER-HQA-14K	58	1.7
RULER-HQA-28K	103	1.7
RULER-HQA-56K	172	2.1
RULER-HQA-112K	299	1.7
NarrativeQA	163	3.3
MuSiQue	95	2.3
Qasper	50	4.4

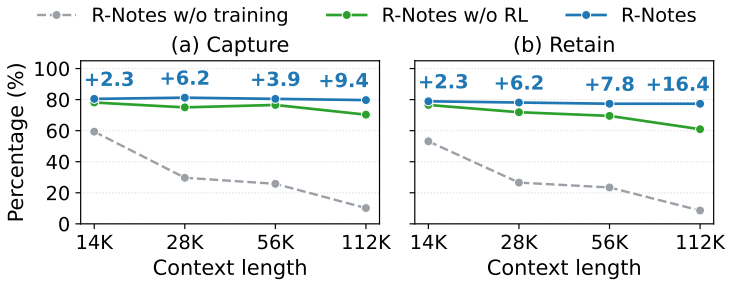

Figure 10: Capture–retain decomposition on RULER-HQA. SFT and RL both strengthen the two abilities, with RL further tightening propagation at longer inputs.

information through later notes until the final answer is produced. Figure 10 measures both abilities on RULER-HQA across four input lengths and three training stages; the measurement procedure is detailed below. R-NOTES without training can sometimes complete the chain near the window size, but both abilities collapse once the input significantly exceeds the window.

This two-ability view shows that training improves the entire propagation chain rather than only one side of it. The SFT cold start, shown as R-NOTES without RL, produces large gains in both capture and retain, showing that demonstrations teach both evidence selection and propagation. RL further improves both abilities under an outcome-only reward, especially at long lengths. Together, SFT and RL make note propagation reliable even far beyond the sliding-window size.

Table 11: Per-cell capture / retain statistics on the same RULER-HQA evaluation set as Table 2. *Cap.* and *Ret.* measure whether the gold answer appears in any note and in the last non-empty note, respectively; $R|C$ is conditional retain among captured samples. All numbers are percentages.

Stage	**Length**	**Cap.**	**Ret.**	**R|C**	**Acc**
R-NOTES w/o training	14K	59.4	53.1	89.5	60.9
	28K	29.7	26.6	89.5	33.6
	56K	25.8	23.4	90.9	25.0
	112K	10.2	8.6	84.6	10.9
R-NOTES w/o RL	14K	78.1	76.6	98.0	89.8
	28K	75.0	71.9	95.8	85.9
	56K	76.6	69.5	90.8	82.8
	112K	70.3	60.9	86.7	76.6
R-NOTES	14K	80.5	78.9	98.1	92.97
	28K	81.3	78.1	96.2	89.84
	56K	80.5	77.3	96.1	89.06
	112K	79.7	77.3	97.1	87.50

Measurement details. We compute the capture and retain statistics in Figure 10 directly from the saved conversation traces of each evaluation cell, using the same RULER-HQA set as Table 2. For each sample, we slice off the final user/assistant turn that produces the answer, parse the remaining

assistant turns with the regular expression `<NOTES>(.*?)</NOTES>` to obtain a list of per-chunk note texts, and treat any turn without a tag as a silent turn. Comparison against the gold answer follows the standard SQuAD/HotpotQA exact-match normalization: lowercase, remove punctuation, drop articles `a/an/the`, and collapse whitespace; a sample's gold answer is considered present in a note if any normalized gold string is a substring of the normalized note. Capture is then the indicator that the gold answer is present in at least one note across the chunk turns, and retain is the indicator that it is present in the most recent non-empty note. Cell-level rates in Table 11 are sample averages of these indicators; conditional retain (R|C) divides the joint count by the number of captured samples. Accuracy is reported separately by the gpt-5-mini judge used in Table 2 and is not part of this calculation.

The matching procedure inherits the limitations of substring exact match: it can miss paraphrases of the gold answer, and is most reliable when gold answers are short entities (as in HotpotQA), rather than long descriptive spans.

Capacity under note load. The following analysis pools 1,240 trajectories: 640 RULER-HQA trajectories and 600 LongBench trajectories from NarrativeQA, Qasper, and MuSiQue. Ground-truth evidence is defined at the level of the complete answer, rather than independently for each hop. As shown in Table 12, once an answer-bearing fact has been captured, its conditional retain rate remains above 92% for every observed note load K. Capture is the bottleneck: it drops from 68.0% for an empty note to 56.5% at $K = 1$–2, then recovers and stabilizes around 65–67% for larger loads. The mild initial drop does not compound as more bullets accumulate.

Table 12: Conditional retain and capture rates under increasing note load. Retain is conditioned on the fact having been captured; Capture measures newly appearing answer-bearing facts.

K (# bullets)	0	1	2	3	4	5–6	7–10	≥ 11
$P(\text{Retain} \mid \text{Capture})$	–	92.6%	94.8%	95.6%	98.1%	92.7%	100.0%	100.0%
Capture rate	68.0%	56.5%	56.5%	64.8%	64.8%	66.7%	66.7%	–

D.4 Preservation of General Capabilities

General capabilities. We compare the base Qwen3-8B and R-NOTES on standard capability benchmarks without long-context prompting. The differences are below 0.4 percentage points on all reported MMLU and GSM8K settings (Table 13), suggesting that the note-taking training does not materially damage ordinary mathematical or knowledge reasoning.

Table 13: General-capability benchmark results (accuracy, %).

Benchmark	Qwen3-8B	R-Notes-8B
MMLU (5-shot)	74.93	74.62
GSM8K flexible (5-shot)	87.57	87.49
GSM8K strict (5-shot)	87.19	87.04

Reasoning under memory load. To test whether notes interfere with unrelated reasoning, we insert GSM8K problems into HotpotQA trajectories either midway through the stream or after the full trajectory. The no-context baseline is 95.0% for both models. As shown in Table 14, R-Notes degrades by roughly 3–5 points under the inserted context, while Qwen3-8B falls to 72.5–83.2% in the corresponding settings. This indicates that the trained model retains substantial mathematical reasoning ability while carrying notes.

D.5 Generalization beyond QA

Non-QA long-context tasks. R-NOTES also improves over the base model on all three non-QA LongBench categories (Table 15). The model is trained only on HotpotQA, so these results test whether learned information preservation transfers beyond multi-hop QA.

Table 14: GSM8K accuracy (%) under inserted HotpotQA trajectories and different note loads.

Case	Note load K	R-NOTES	Qwen3-8B
No context baseline	0	95.0	95.0
Mid-trajectory	1–2	92.1	83.2
Mid-trajectory	3–4	89.4	76.1
Mid-trajectory	≥ 6	90.0	75.0
Mid-trajectory total	–	91.5	81.6
After full trajectory	1–2	89.7	78.1
After full trajectory	4	87.5	72.5
After full trajectory total	–	89.5	77.5

Table 15: Non-QA LongBench results (accuracy, %).

Task	Qwen3-8B	R-Notes-8B
Summarization	20.98	22.11
Few-shot ICL	59.53	61.02
Synthetic retrieval	86.63	89.40
Overall non-QA average	55.71	57.51

Deep-research agent task. We evaluate R-Notes as the backbone of a BrowseComp-Plus agent. The agent uses Qwen3-Embed-0.6B for retrieval with top-$k = 5$, truncates each retrieved result to its first 512 tokens, and keeps the task query fixed throughout the interaction. Accuracy is scored by an LLM judge against the ground truth; Recall is the number of human-verified evidence documents retrieved during the complete interaction. R-Notes outperforms both a Qwen3-32B baseline and SearchR1-32B, despite using a backbone that is four times smaller (Table 16).

Table 16: BrowseComp-Plus results for a query-driven deep-research agent.

Model	Recall	Accuracy
Qwen3-32B	3.45	4.10
SearchR1-32B	5.30	5.66
R-Notes	**6.60**	**9.28**