Long-Context Modeling: Liquid Recurrence, Spiking Gating, Resonant Memory, Write-Time Reasoning and Memory-Efficient Inference

Anonymous authors
Paper under double-blind review

Abstract

Long-context benchmarks have grown to tens of millions of tokens. At that scale the key–value cache of an attention model occupies terabytes, while a recurrent state of fixed size has no way to hold on to a sentence seen 10^8 steps ago. This paper studies *self-written memory* (SWM): a language model reads a stream in chunks of 240 tokens, judges for each chunk whether it holds a fact worth keeping, writes such facts down as short notes, and later answers questions from a small memory drawn from those notes. Whether to write is decided by the model's own probability of answering "Nothing" to a fixed writing instruction; there are no extraction rules, and the criterion is the same for every task and every length. We run SWM with a 247M-parameter attention-free model made of liquid recurrent layers and spiking delta-memory layers, pretrained on roughly one billion tokens. On streams of 1M, 10M, 50M and 100M tokens in which BABILong facts and needle tasks are buried in Project Gutenberg text, SWM with a fine-tuned reader checkpoint scores 61.4%, 55.0%, 52.1% and 47.1%. With the writer checkpoint itself as reader the scores are 42.1%, 42.9%, 41.4% and 42.9%, compared with 24.3%, 25.7%, 17.9% and 5.7% when the same checkpoint answers from BM25-retrieved raw chunks. Because the task formats appear in our generated fine-tuning data, these numbers measure in-distribution scaling rather than zero-shot ability. The writer flags 98.2–100% of the fact-bearing chunks and 0.20–0.46% of the filler chunks; GPU memory during writing and reading is independent of stream length, and only the note store grows. Ablations, an oracle-memory bound and an error analysis trace most of the remaining errors to multi-hop reasoning in the reader and to facts that were written but not selected into the memory.

1 Introduction

BABILong (Kuratov et al., 2024) and RULER (Hsieh et al., 2024) test whether a model can find a few facts hidden in a very long text and combine them. Once the text runs to tens of millions of tokens, simply keeping it available is the hard part. The key–value cache of Llama-2-7B (Touvron et al., 2023) costs 0.5 MiB per token, about 24 TiB for 50M tokens; Llama-3-8B (Dubey et al., 2024) uses grouped-query attention and still needs about 6 TiB. Recurrent and state-space models (Gu & Dao, 2023; De et al., 2024) have a state of constant size, but that state is overwritten at every step, and nobody should expect it to preserve one sentence through 10^8 steps of unrelated prose.

Our approach keeps recall and reasoning apart. The model reads the stream in short chunks and, for each one, judges whether it contains something worth remembering; if so, it writes the fact down as a note. When a question comes, a small memory is put together from the notes and a reader checkpoint of the same model answers from it, never seeing the stream. We call this *self-written memory* (SWM). What matters to us is that the writing decision belongs to the model. There are no extraction rules for movements, locations, possessions or passwords. The only signal is the model's probability of answering "Nothing" to one fixed writing instruction, and we use that signal unchanged across tasks and lengths.

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.

The model we use has 247M parameters and no attention. Its layers alternate between liquid recurrent layers, where each channel carries a learned time constant in the manner of liquid time-constant networks (Hasani et al., 2021), and spiking delta-memory layers, where leaky integrate-and-fire neurons (Maass, 1997) gate a delta-rule associative matrix (Schlag et al., 2021; Yang et al., 2024). It was pretrained on about one billion tokens, a small budget by current standards, and then fine-tuned to write and read notes.

Our contributions are:

- A simple protocol, SWM, in which one learned probability decides what to write and lexical retrieval over the written notes builds the memory, applied without task-specific rules to streams of up to 100M tokens (Section 4).
- An evaluation at 1M, 10M, 50M and 100M tokens on streams built with the BABILong recipe and on needle tasks, with a retrieval baseline that uses the same model, an oracle-memory bound, a memory-composition ablation and an error analysis (Sections 5–6).
- Evidence that the writer is reliable (it flags 99.6–100% of fact-bearing chunks and 0.25% of filler chunks) and that most remaining errors come from the reader checkpoint, whose accuracy falls as the memory grows beyond 10–20 notes and which fails on most multi-hop questions (Section 7).

2 Related work

Long-context benchmarks. bAbI (Weston et al., 2015) defines small reasoning tasks over short stories. BABILong (Kuratov et al., 2024) hides the facts of bAbI tasks in book text from PG-19 (Rae et al., 2020) and measures accuracy as the text grows. RULER (Hsieh et al., 2024) adds synthetic needle-retrieval and multi-key tasks. Accuracy of attention models is known to drop for facts in the middle of long inputs (Liu et al., 2024).

Recurrent, linear and spiking models. Liquid time-constant networks (Hasani et al., 2021) make the decay of each neuron input-dependent. Gated linear recurrences (De et al., 2024) and selective state-space models (Gu & Dao, 2023) reach competitive language modeling with constant-size state. Linear attention can be read as a fast-weight memory updated by the delta rule (Schlag et al., 2021; Yang et al., 2024). Spiking language models (Zhu et al., 2023) replace dense activations with binary spikes. Our model combines these components but does not rely on its recurrent state to carry facts across the stream.

Memory-augmented models. The Recurrent Memory Transformer (Bulatov et al., 2022) and its associative variant (Rodkin et al., 2024) pass memory tokens or an associative matrix from segment to segment. Infini-attention (Munkhdalai et al., 2024) and Titans (Behrouz et al., 2024) add a compressive or test-time-trained memory to attention. MemGPT (Packer et al., 2023) lets a large model page information in and out of its context. In these methods the memory is a vector state or is managed by a large instruction-following model; in SWM the memory is plain text written by a 247M model, and the decision to write is a single probability, the no-fact probability $p_\varnothing$.

Retrieval augmentation. Retrieval-augmented generation (Lewis et al., 2020) retrieves raw passages with a retriever such as BM25 (Robertson & Zaragoza, 2009). We use retrieval over raw chunks (BM25) as a baseline and a BM25-style lexical score over notes as one part of SWM; the difference is that SWM retrieves among at most a few thousand short notes that the model chose to write, not among up to 416,667 raw chunks.

3 Model

Architecture. The model has 16 layers of width 1024 and 246.9M parameters, uses a byte-level BPE tokenizer with 8,192 entries, and has no attention. With layers numbered from 0, even layers are *liquid recurrent layers*: a short convolution followed by a gated linear recurrence $\boldsymbol{h}_t = \boldsymbol{a}_t \odot \boldsymbol{h}_{t-1} + \boldsymbol{b}_t \odot \boldsymbol{x}_t$ whose decay is $\boldsymbol{a}_t = \exp(-\boldsymbol{r}_t/\boldsymbol{\tau})$, where $\boldsymbol{r}_t$ is an input-dependent rate and $\boldsymbol{\tau} = \exp(\boldsymbol{\lambda})$ is a learned time constant per channel group. Odd layers are *spiking delta-memory layers*: an associative matrix is updated with the delta rule from projected keys and values with an input-dependent write strength, and its output is gated by leaky integrate-and-fire neurons with learned leak, threshold and adaptation,

which emit binary spikes. Every layer ends with a SwiGLU feed-forward block (Shazeer, 2020). An optional auxiliary module in our implementation is disabled in all experiments; its parameters are included in the count above but are not used. Appendix A lists the hyperparameters.

Training. The model was pretrained on about one billion tokens of general text. It was then fine-tuned with a mixture of instruction data and task data (Appendix B): general instruction following, translation and short answers, note writing, note reading, question answering over short bAbI-style stories, needle-style facts, and a small share of plain text. The note-writing, note-reading and bAbI-style data are produced by our own generators; they share the task formats and the entity vocabulary of bAbI, which we discuss in Section 8. Each training window holds a single document, loss is computed only on assistant tokens and on the end-of-turn marker, and each source has its own loss weight.

Checkpoints. The *writer checkpoint* W is step 8,000 of the fine-tuning run. A second run of the same fine-tuning recipe gives the checkpoint W_2 (step 7,350), which we use as writer in a replication of the 50M and 100M streams. The *reader checkpoint* R is obtained from W_2 by 900 further steps in which note reading has the largest share of the mixture. We fixed the selection rule for R before scoring any stream: take the checkpoint with the highest accuracy on 800 held-out note-reading items. The held-out accuracies were 63.2%, 65.5%, 66.8% and 71.8% at steps 7,500, 7,750, 8,000 and 8,250, so R is step 8,250. We also report step 8,000 of this second run, denoted R' (not to be confused with W, which is step 8,000 of the first run), to show how much the stream scores depend on the reader checkpoint.

4 Self-written memory

Writer pass. A stream $x_{1:L}$ is split into chunks $c_1, \ldots, c_n$ of $C = 240$ tokens. Each chunk is placed in one fixed prompt,

```
### User: Read the passage and write down every fact worth
remembering, one per line. If there is none, answer Nothing.
<passage> c_i </passage> ### Assistant:
```

and the writer computes, with one forward pass from a reset state, the *no-fact probability* $p_\varnothing(c_i)$, the probability that the first answer token is "Nothing". A chunk is *flagged* if $p_\varnothing(c_i) < 0.5$. For flagged chunks the writer then generates its answer greedily (at most 96 tokens); every non-empty line that does not start with "Nothing" becomes a *note*. The *note store* of the stream is the set of triples $(p_\varnothing(c_i), i, \text{line})$. Because the state is reset for every chunk, chunks are independent: the writer pass is batched and parallel, and its GPU memory use does not depend on L. We fixed the threshold at 0.5 before any stream experiment and did not tune it.

Memory. For a question q, the *memory* $M(q)$ is the union of two sets of notes: the k_r notes with the highest lexical score against q, and the k_c notes with the lowest $p_\varnothing$. The lexical score of a note is $\sum_{w \in q \cap \text{note}} \log\big(\frac{N - n_w + 0.5}{n_w + 0.5} + 1\big)$, the inverse-document-frequency term of BM25 (Robertson & Zaragoza, 2009) computed over the N notes of the stream, where n_w is the number of notes containing word w and function words are ignored. The selected notes are listed in stream order. We use $k_r = k_c = 10$, chosen on the 50M streams with W as reader (Section 8). In the ablation with $k_r = 0$, the k_c flagged chunks with the lowest $p_\varnothing$ contribute all of their notes, so the memory can hold slightly more than k_c notes.

Reader pass. The reader receives

```
### Memory: M(q) ### User: q ### Assistant:
```

and decodes greedily until a newline (at most 32 tokens). The reader never sees the stream itself.

What is learned and what is fixed. The model decides which text is worth keeping, how to phrase the notes, and what to answer. Everything else is fixed: the chunk length, the threshold 0.5, the memory sizes k_r and k_c, and the lexical score, all shared across tasks and lengths. Of these, only k_r and k_c were chosen on test streams (Section 8).

Table 1: Accuracy (%) on long streams. SWM uses the writer checkpoint W and the memory of Section 4 ($k_r = k_c = 10$); the columns differ only in the reader checkpoint. R is the reader chosen by the held-out rule; R' is the previous checkpoint of the same run. The retrieval baseline uses W. One binomial standard error is about 4 points at 1M–50M (140 streams) and about 6 points at 100M (70 streams); it does not capture variation across filler texts, because each length uses one shared filler text.

	SWM, reader checkpoint			Retrieval over
Stream length	R	R'	W	raw chunks (W)
1M	61.4	65.7	42.1	24.3
10M	55.0	54.3	42.9	25.7
50M	52.1	58.6	41.4	17.9
100M	47.1	57.1	42.9	5.7
50M†	52.9	49.3	39.3	–
100M†	47.1	47.1	37.1	–

†The writer pass used W_2 (Section 3) instead of W, and the column "W" reports W_2 as reader.

5 Experimental setup

Streams. We take the facts and questions of qa1–qa5 from the official BABILong release (the 0k split, which contains only the facts) and insert the facts, at sentence boundaries and in their original order, at random positions of filler text from public-domain Project Gutenberg books, following the BABILong construction, which uses PG-19 books as filler text. The question is asked after the stream. We add two needle tasks in the style of RULER: a single needle ("the special magic number for X is N") and a multi-key task, in which several keys are inserted and one is asked. Stream lengths are 1M, 10M, 50M and 100M tokens of our tokenizer; on the filler text one of our tokens corresponds to 0.877 GPT-2 tokens, so 50M and 100M of our tokens are about 43.8M and 87.7M GPT-2 tokens. We use 20 samples per task at 1M, 10M and 50M (140 streams per length) and 10 per task at 100M (70 streams). All streams of a given length share the same filler text; only the inserted facts and the question differ. This lets the writer pass over the filler be computed once per length.

Scoring. For qa1–qa5 we use the BABILong rule: the answer is correct if the set of task labels (locations or entities) that appear in the answer and not in the question has exactly one element and that element is the target. For needle tasks the answer is correct if it contains the target number.

Baselines and bounds. *No memory*: the reader receives an empty memory. *Retrieval over raw chunks*: BM25 selects the 8 chunks of 240 tokens most similar to the question and the writer checkpoint W answers from them in the BABILong prompt format; using W controls for the reader checkpoint when this baseline is compared with the SWM column that uses W as reader, although the two methods give the reader different inputs. *Oracle memory*: the reader receives exactly the inserted facts and nothing else, which bounds what a perfect writer and a perfect retrieval step could give with the same reader.

Hardware. The writer pass for 50M tokens (208,334 chunks) took 2,777 s on one NVIDIA GB10 (75 chunks per second, about 18k tokens per second). For 100M tokens (416,667 chunks) it took 4,644 s on four RTX 3090 GPUs that were shared with other jobs.

6 Results

Main results. Table 1 shows accuracy as the stream grows from 1M to 100M tokens. With the same checkpoint as reader (W), SWM is 17–37 points above retrieval over raw chunks from 1M to 100M tokens (17.8, 17.2, 23.5 and 37.2 points), because lexical retrieval among hundreds of thousands of raw chunks rarely finds the right ones: the 8 retrieved chunks contain 8.5% of the fact-bearing chunks pooled over the 1M and 10M streams and 2.0% pooled over the 50M and 100M streams. Replacing W by R as reader adds another 11–19 points from 1M to 50M and 4.2 points at 100M; because R was trained from W_2, a checkpoint of a second fine-tuning run, this gain combines reader fine-tuning

Table 2: Memory composition on the 50M streams (140 streams, writer checkpoint W; accuracy in % for reader checkpoints R and R'). "Notes" is the mean number of notes in the memory; "Facts in memory" is the share (%) of inserted facts that appear verbatim in the memory.

Memory (k_r by lexical score, k_c by lowest $p_\varnothing$)	Notes	Facts in memory	R	R'
Empty ($k_r = k_c = 0$)	0	0.0	5.0	2.9
$k_r = k_c = 10$ (SWM)	17.5	47.9	52.1	58.6
$k_r = 10$, $k_c = 0$	10.0	35.3	60.0	62.1
$k_r = k_c = 5$	8.6	33.5	54.3	57.1
$k_r = k_c = 20$	34.7	65.3	38.6	44.3
$k_r = 0$, $k_c = 20$	20.1	42.0	32.1	42.1
$k_r = 0$, $k_c = 100$	100.1	88.5	28.6	24.3

with the difference between the two runs. Repeating the streams with the writer checkpoint W_2 of the second fine-tuning run gives, with R, 52.9% at 50M and 47.1% at 100M (52.1% and 47.1% with W as writer), so the result does not depend on one writer checkpoint. From 1M to 50M the accuracy of SWM with reader R drops by 9 points while the stream grows 50 times, and at 100M it is 47.1%, more than eight times the 5.7% of retrieval over raw chunks. The difference between R and R' (6.5 points at 50M and 10.0 points at 100M, where one standard error is about 6 points) is at most about 1.7 standard errors; we report R because it was fixed by the held-out rule. Per-task results are in Appendix C: with R, single-fact questions (qa1) are answered in 85–90% of the streams at every length up to 50M and three-argument questions (qa5) and the single needle in 65–100%, while the multi-hop tasks qa2–qa4 stay between 15% and 40%.

Memory composition. Table 2 varies how the memory is built. With an empty memory, R and R' answer 5.0% and 2.9% of the questions correctly. The 100 notes with the lowest $p_\varnothing$ contain 88.5% of the inserted facts but yield only 24–29% accuracy: R and R' cannot find the relevant notes among 100. Accuracy is highest with about 10 notes and lower with 20–35 notes, even though more facts are then present. Lexical retrieval over notes is therefore important, and the reader checkpoint, not the recall of the writer checkpoint, limits the useful memory size. Using only the 10 notes with the highest lexical score is better than our configuration at 50M (60.0 vs. 52.1 with R and 62.1 vs. 58.6 with R', up to about 2 standard errors), so adding the notes with the lowest $p_\varnothing$ did not help at this length. We keep the configuration of Section 4 because it was fixed before the 1M, 10M and 100M streams were scored, and we report this row as an ablation.

7 Analysis

The writer. On held-out note-writing items, W flags every chunk that contains a fact, flags none of the chunks without a fact, and writes the exact reference note for 60.1% of the fact-bearing chunks. On the streams it flags 2,050 of 2,058 fact-bearing chunks at 1M and 2,069 of 2,069 at 10M, and the notes of the fact-bearing chunks contain the inserted fact verbatim for 98.6%, 99.3% and 99.5% of the facts at 1M, 10M and 50M. Of the 208,334 filler chunks of the 50M streams it flags 528 (0.25%). On the 100M streams it flags 1,196 of 1,198 fact-bearing chunks and 822 of 416,667 filler chunks (0.20%), and its notes contain 99.7% of the inserted facts verbatim. Since the filler is shared, the note store of one 50M stream comes from these 528 chunks and the about 15 fact-bearing chunks of that stream, a text of roughly 25 KB. With W_2 as writer, 2,037 of 2,072 fact-bearing chunks (98.3%) and 912 of 208,334 filler chunks (0.44%) are flagged on the 50M streams, and 1,176 of 1,198 fact-bearing chunks (98.2%) and 1,916 of 416,667 filler chunks (0.46%) on the 100M streams; its notes contain 95.3% and 95.2% of the inserted facts verbatim. Facts are therefore almost always written; when a fact is missing from the memory, it was written but not selected.

Where the errors come from. Table 3 gives the reader accuracy when the memory contains exactly the inserted facts. With oracle memory, R reaches 66.2% and R' 68.6%, and the pattern across tasks is the same as on the streams: qa1, qa5 and the single needle are nearly solved, whereas qa2–qa4 remain low. On the 50M streams, 83 of the 140 memories built by SWM contain every inserted fact of their stream; on these 83 streams R answers 60.2% and R' 67.5% correctly, close to their

Table 3: Reader accuracy (%) with oracle memory (exactly the inserted facts; 30 samples per task). W_2 is the fine-tuned checkpoint from which R and R' were trained, before reader fine-tuning.

Reader	qa1	qa2	qa3	qa4	qa5	Multi-key	Single needle	Mean
W_2	97	40	27	27	30	67	100	55.2
R'	100	47	40	27	97	70	100	68.6
R	87	53	37	27	97	63	100	66.2

oracle-memory means of 66.2% and 68.6% (the task mix differs, so the comparison is indicative). For these streams the remaining gap to perfect accuracy is a reasoning gap of the reader. On the other 57 streams, facts that were written but not selected into the memory also cost accuracy: R answers 40.4% of them correctly.

GPU memory cost. The model keeps no per-token cache. The recurrent state of one sequence is about 1.25 MB, and one inference process used about 3.0 GB of GPU memory, most of it weights and workspace, when it consumed a 1M-token context at about 1,960 tokens per second on a shared RTX 3090. The writer pass uses the same GPU memory for 1M and 100M tokens because chunks are processed independently. What grows with the stream is the note store, about 25 KB of text for a 50M stream. For comparison, a key–value cache for 50M (100M) tokens takes about 24 (48) TiB for Llama-2-7B and 6.0 (11.9) TiB for Llama-3-8B in 16-bit precision (Appendix D).

Spiking activity. Measured on 204,800 tokens with W, the leaky integrate-and-fire neurons of the eight spiking delta-memory layers fire in 6.1% of the positions on average, from 18.0% and 24.9% in the two lowest spiking delta-memory layers to 0.3–1.7% in the upper six (Appendix E).

8 Limitations

Where long-range memory lives. The notes, and only the notes, carry facts across the stream; every chunk is read from a reset state. Nothing here shows that the liquid or spiking dynamics retain information over 100M tokens. What the results show is narrower: a small recurrent model can decide what to write down and can answer from what it wrote.

Training data and test data. Our fine-tuning data include generated bAbI-style stories, notes and needle facts that share the formats and the entity vocabulary of the test tasks, although no test sample is used in training. The results should be read as in-distribution for the task formats, not as zero-shot long-context ability.

Choices made on test streams. The memory configuration $k_r = k_c = 10$ was chosen on the 50M streams using W as reader. The 50M numbers for this configuration are therefore not fully held out; the 1M, 10M and 100M streams were not used for any choice. The reader checkpoint was chosen on held-out note-reading items only.

Benchmark construction. The 50M and 100M streams are our own construction following the BABILong recipe, not an official release, and the needle tasks are generated in the style of RULER, not taken from it. All streams of one length share the same filler text. Lengths are counted in our tokenizer, which yields about 14% more tokens than the GPT-2 tokenizer on the filler text, so our 50M and 100M streams hold about 43.8M and 87.7M GPT-2 tokens. Sample sizes are small (140 or 70 streams per length) and each number comes from a single run. In a first run of the 50M replication with W_2, most notes of fact-bearing chunks were empty because inference processes failed while the GPUs were shared with other jobs (23.6% of the facts were written, against 95.3% after the rerun); we discarded that run, reran note generation with the same settings, and report the rerun.

Reasoning. The reader checkpoints fail on most multi-hop questions (qa2–qa4) even with oracle memory, and their accuracy falls as the memory grows beyond 10–20 notes.

9 Conclusion

Given permission to take notes, a 247M attention-free model pretrained on about one billion tokens can answer questions about streams of up to 100M tokens. It picks out the chunks that hold facts, writes them down, and answers from a small memory drawn from its own notes. The writer checkpoint misses very few fact-bearing chunks, and GPU memory during writing and reading does not depend on the stream length; only the note store grows. The weak point is the reader, which handles single-fact questions well and multi-hop questions poorly even when handed exactly the right facts. A larger reader, or better reader fine-tuning, is the obvious next step.

AI use statement

In this work, we used generative AI tools (coding agents based on large language models) to implement and debug the training and evaluation code, to write the programs that generate the synthetic fine-tuning data, the needle tasks and the streams, to design and run experiments, to analyze and interpret results, to draft the text of this paper and to translate it. The synthetic note-writing, note-reading, bAbI-style and needle data are produced by these programs, not sampled from a language model; the general instruction data is an existing collection that we did not create and that may contain text generated by language models. Mathematical claims and proofs are not applicable to this work. All AI-assisted code, data generators and numbers were checked by the authors against the raw result files, and the authors reviewed all text. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

Ethics statement

This work uses public-domain books from Project Gutenberg and the publicly released BABILong data; it involves no human subjects and no personal data. The model is small and was not trained or evaluated for deployment; any deployment would require the usual safety evaluation.

Reproducibility statement

Section 3 and Appendices A and B describe the model and the fine-tuning mixture, Section 4 gives the exact prompts, thresholds and memory rule, and Section 5 describes the stream construction and the scoring rule. The training code, the data generators, the stream construction scripts, the evaluation scripts and the per-stream outputs are available at the repository linked in the camera-ready version and as anonymized supplementary material.

References

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, 2022.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

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? In *Conference on Language Modeling*, 2024.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

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 *Advances in Neural Information Processing Systems*, 2020.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Ivan Rodkin, Yuri Kuratov, Aydar Bulatov, and Mikhail Burtsev. Associative recurrent memory transformer. *arXiv preprint arXiv:2407.04841*, 2024.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, 2024.

Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K. Eshraghian. SpikeGPT: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

A Model hyperparameters

Parameters	246,941,229
Layers	16 (8 liquid recurrent, 8 spiking delta-memory, alternating)
Width	1024
Tokenizer	byte-level BPE, 8,192 entries
Feed-forward	SwiGLU in every layer
Attention	none
Pretraining	about 10^9 tokens
Reader fine-tuning	900 steps, 8 sequences of 1,024 tokens per step, peak learning rate 3×10^{-5}, AdamW ($\beta = 0.9, 0.95$), weight decay 0.1, gradient clipping 1.0

B Fine-tuning mixture

The fine-tuning recipe that produced W and W_2 mixes the following sources; the first number is the sampling share and the second the loss weight on target tokens: general instruction data (0.20, 1), translation and short answers (0.05, 2), question answering with the facts of a short story in the context (0.04, 20), bAbI-style stories with questions (0.12, 40), note writing (0.25, 15), note reading (0.20, 30), dialogues that mix note writing and note reading (0.05, 3), short reasoning questions (0.03, 10), and plain web and encyclopedia text (0.06, 0.3 for web text). Note-writing examples ask for every fact of a passage, one per line, or "Nothing" when the passage contains none. The reader fine-tuning raises the share of note reading to 0.45. Instruction examples that begin with one of 23 fixed refusal openings are removed, and 1 in 200 documents of every source is held out.

C Per-task results

Length	Reader	qa1	qa2	qa3	qa4	qa5	Multi-key	Single needle
1M	R	85	35	20	25	95	70	100
1M	R'	100	40	20	25	95	80	100
10M	R	85	40	25	25	90	55	65
10M	R'	85	40	20	25	85	40	85
50M	R	90	40	15	30	70	35	85
50M	R'	90	35	10	20	90	65	100
100M	R	80	30	0	30	80	40	70
100M	R'	90	30	0	30	100	60	90
50M†	R	95	40	30	35	90	15	65
50M†	R'	100	45	25	40	75	45	15
100M†	R	100	30	10	40	80	30	40
100M†	R'	100	10	20	20	70	40	70

†Writer checkpoint W_2; 20 streams per task at 50M and 10 at 100M.

Retrieval over raw chunks with W: 1M 24.3%, 10M 25.7%, 50M 17.9%, 100M 5.7%. Without any long stream, on the official BABILong 0k, 1k and 4k splits read directly in context (100 samples per task), W reaches 33.0%, 30.8% and 29.2% on average over qa1–qa5.

D Key–value cache sizes

A key–value cache stores, per token, 2 × layers × KV heads × head dimension values. In 16-bit precision this is $2 \times 32 \times 32 \times 128 \times 2$ B $= 0.5$ MiB per token for Llama-2-7B and $2 \times 32 \times 8 \times 128 \times 2$ B $= 128$ KiB per token for Llama-3-8B. For 50M tokens this gives 23.8 TiB and 6.0 TiB, and twice as much for 100M tokens.

E Spiking activity

Mean firing rate of the spiking neurons per delta-memory layer, measured on 204,800 tokens with W: layer 1: 18.0%, layer 3: 24.9%, layer 5: 0.49%, layer 7: 1.6%, layer 9: 0.63%, layer 11: 0.31%, layer 13: 0.94%, layer 15: 1.65%; mean 6.1%.