The Sink Tax: Auditing Memory Tokens and Assigning Their Roles in Long-Context Language Models

Anonymous authors
Paper under double-blind review

Abstract

In-context memory (ICM) systems reduce the key-value cache of long contexts by splitting the context into chunks, compressing each chunk into a few memory tokens, and evicting the raw tokens. However, whether the model uses the information that each memory token stores has not been examined. In this paper, we introduce a slot audit that tests whether a memory token stores the information of its chunk by replacing its key-value states with those of another document. We apply the audit to designs that evict the raw tokens, from models trained under the mask of gist and beacon tokens to the official checkpoints of AutoCompressor and RMT. We find that the attention sink shifts to the memory tokens, and across these existing designs the memory token that receives the most attention consistently stores no information of its chunk that the model uses. Yet, removing this token still raises the loss under the mask that hides each chunk from later queries. Therefore, the removal test, which existing work uses to validate memory tokens, counts the sink as memory. We term the fraction of memory tokens that the sink takes the *sink tax*. Based on these findings, we introduce Assigned-Role Memory (ARM), which assigns one role to every cache position, whereas in prior designs the roles emerge during training. ARM assigns the sink of the memory slots to one sink slot with a value fixed at zero and a token-like key. This sink slot removes the sink from every memory slot at 124M, 340M, and 0.8B parameters. Once the roles are fixed, ARM keeps one set of memory slots for the entire context and updates the memory slots after every chunk, whereas prior designs accumulate one set per chunk. The cache size therefore remains constant as the context length grows. In long-context language modeling on PG19, ARM matches the long-range perplexity of per-chunk accumulation with 17 cache positions instead of 384 and recalls 79% of the facts placed in earlier chunks compared to 54%. With 17 cache positions, ARM lowers the long-range perplexity of a sliding window by 29%. ARM maintains its perplexity at 128k tokens, 32× the training length, while per-chunk accumulation degrades beyond the training length. At 128k tokens, ARM runs at the decoding throughput of a sliding window with a lower perplexity and at 39× that of per-chunk accumulation at the ratio of prior work.

1 Introduction

Large language models now accept 128k tokens at once (Grattafiori et al., 2024) and process long documents, code repositories, and conversations (Zhang et al., 2025; Yuan et al., 2025). However, a Transformer stores a key and a value for every token at every layer. The key-value (KV) cache therefore grows linearly with the context length (Kwon et al., 2023). For Llama-3.1-8B, a 128k-token context needs 16 GB in 16-bit precision for this cache alone, comparable to the weights of the model. Since the weights occupy a fixed amount of memory, the cache of each request determines how many requests one GPU can serve concurrently, and consequently the throughput (Kwon et al., 2023).

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

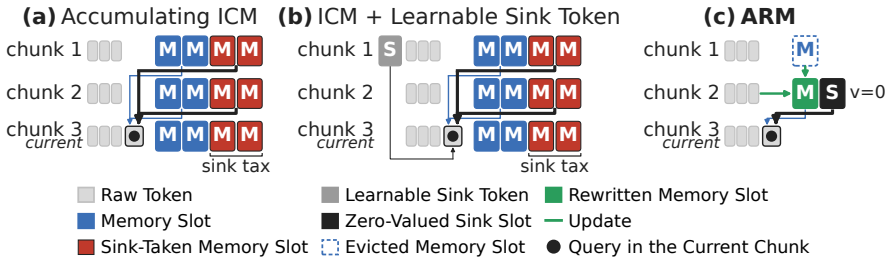


Figure 1: Three designs of in-context memory. (a) Accumulating ICM: one set of memory slots per chunk, and the attention sink takes part of every set, the sink tax. (b) ICM + Learnable Sink Token: the token receives a small part of the sink, and the sink tax remains. (c) ARM: a zero-valued sink slot takes the sink from the memory slots, and one set of memory slots is updated after every chunk. ARM matches the long-range perplexity of per-chunk accumulation at the cache of a sliding window.

Many approaches reduce this cost with training-free cache management or in-context memory. Training-free cache management keeps the tokens that earlier queries attended to most and evicts the rest (Kwon et al., 2023; Zhang et al., 2023; Li et al., 2024). In-context memory (ICM) splits the context into chunks and trains a few memory tokens to summarize each chunk before its raw tokens are evicted (Mu et al., 2023; Zhang et al., 2025; Kuaishou, 2026; Yuan et al., 2025; Behrouz et al., 2025; Deng et al., 2025a). Yet these methods accumulate the memory tokens of every chunk. The cache therefore still grows linearly with the number of chunks.

Prior work validates the memory tokens only by the drop in task performance when the tokens are removed (Zhang et al., 2025; Mu et al., 2023; Kuaishou, 2026). The drop shows that the model depends on the presence of the tokens, not on what any token stores. The attention scores of a softmax head sum to one. A query with no relevant token therefore places its unnecessary attention on the attention sink. The sink is a fixed position that every query can see, usually the first token (Xiao et al., 2024; Gu et al., 2025). However, in-context memory evicts the raw tokens of every earlier chunk. The memory tokens then become the only positions that every later query can see. UniGist and Titans prepend a learnable token for the sink (Deng et al., 2025a; Behrouz et al., 2025), but whether this token takes the sink in the trained model has not been examined.

We call the cache position that a memory token occupies a memory slot. In this work, we introduce a slot-level audit that identifies the role of each slot, memory or sink. The memory test replaces a memory slot with what the same slot computes on a different document (Vig et al., 2020; Meng et al., 2022). Thus, only a slot that stores the information of its chunk changes the loss.

We find that the attention sink shifts to the memory slots in accumulating designs that evict the raw tokens. In every model of these designs, the memory slot that receives the most attention stores no content that the model uses. Under the mask of compression training, the sink takes two to three of the four memory slots per chunk at 124M, 340M, and 0.8B. Deleting the slot that takes the most attention raises the loss, but replacing its content has no effect on the loss. The removal test therefore counts such a slot as memory. Even with the kept first token, the learnable sink token, or the biases of prior work (Xiao et al., 2024; Gu et al., 2025; Deng et al., 2025a), the memory slots keep the sink, and the added sink position receives at most 8.0% of the attention. We term the fraction of memory slots that the sink takes the *sink tax* (Figure 1a).

Prior designs supply the positions, but the roles emerge during training: the added sink position does not take the whole sink, and which memory slots the sink takes varies across training runs. Based on the above analysis, we introduce **Assigned-Role Memory (ARM)**, an in-context memory design that assigns one role to every slot. For the sink role, ARM adds one slot that every query can see, with a key computed like the key of any token and a value fixed at zero. The sink duty is the unnecessary attention that softmax must place somewhere. A slot with a value fixed at zero receives this attention without adding a value vector to the output of any head. The sink duty is therefore allocated to this slot during training. This slot takes the sink from the memory slots and receives

42% of the attention at 124M under the mask of compression training. No memory slot keeps a part of the sink at 124M, 340M, and 0.8B parameters. Mixture-of-experts models improve throughput with zero-computation experts (Jin et al., 2025), and the sink slot is the attention counterpart of such an expert. For the memory role, ARM evicts the raw tokens beyond the window, and the memory slots alone store the earlier text.

Finally, we show that when the roles are assigned, one set of memory slots suffices for all chunks, while prior designs accumulate one set per chunk. After every chunk, the memory tokens attend to the previous set of memory slots and to the raw tokens of the chunk. The keys and values of the memory tokens become the new set of memory slots at every layer, and the previous set is evicted. We call this set of memory slots the rewritten set. The cache size therefore remains constant as the context length grows. On PG19, the rewritten set maintains its perplexity at 128k tokens, 32× the training length, with 17 cache positions, while per-chunk accumulation reads 16,256 positions at more than three times the perplexity. The rewritten set recalls 79% of the facts placed in earlier chunks with the same 17 positions, compared to 54% for per-chunk accumulation with 384. At 128k tokens, ARM keeps the throughput of a sliding window and achieves 39× that of per-chunk accumulation.

Our contributions are as follows.

- **The sink tax of in-context memory.** We introduce a slot audit and find that the memory slot that receives the most attention in accumulating designs stores no content that the model uses, while the removal test counts this slot as memory.
- **One sink slot takes the sink from every memory slot.** One zero-valued sink slot shared across all chunks takes the sink from every memory slot at 124M, 340M, and 0.8B parameters.
- **One rewritten set for a constant cache.** One rewritten set matches the long-range perplexity of per-chunk accumulation with 17 cache positions instead of 384, maintains its perplexity at 32× the training length, and achieves 39× the decoding throughput.

2 Related Work and Problem Setting

In-context memory tokens. In-context memory reduces the cache by training a few tokens to replace evicted raw tokens, such as gist, beacon, and summary tokens (Mu et al., 2023; Zhang et al., 2025; Kuaishou, 2026). We refer to all such tokens as memory tokens (extended in Appendix A). To make the tokens store the context, training hides the evicted tokens from later queries. However, prior work validates these tokens only by the removal test. Moreover, whether the model uses the information that a memory token stores has not been examined.

Attention sinks. The attention sink is one cache position that every query can see, usually the first token. In addition, the value vector of that position has a small norm (Xiao et al., 2024; Gu et al., 2025; Sun et al., 2024; Wong et al., 2025; Guo et al., 2024). The loss of a sliding window rises sharply when the sink position is evicted (Xiao et al., 2024). Remedies give the sink a dedicated position by retaining the initial tokens, prepending a learnable sink token, or adding a per-head bias or a scalar logit to the softmax (Xiao et al., 2024; Gu et al., 2025; Deng et al., 2025a; Sun et al., 2024; OpenAI, 2025). Other remedies remove the need for a sink with a sigmoid normalization or an output gate, but both require a model trained with them (Gu et al., 2025; Qiu et al., 2025). Prior work verifies these remedies in models without memory tokens (Xiao et al., 2024; Gu et al., 2025; Qiu et al., 2025). In contrast, we examine whether one position takes the sink from the memory tokens after the raw tokens are evicted, and whether any memory token keeps a part of it.

Problem setting and notation. A model with in-context memory reads its input in chunks of C raw tokens and appends R learnable tokens after each chunk. We call the cache positions of these tokens registers (Darcet et al., 2024). We call the R registers appended after a chunk the register set of the chunk. A memory slot is a register intended to store the information of its chunk. Existing methods retain the registers of every earlier chunk in the cache (Mu et al., 2023; Zhang et al., 2025; Kuaishou, 2026). We denote this design as Accumulating ICM. A sliding window keeps the most recent W raw tokens visible, as in summary attention (Kuaishou, 2026). Chunk-local masking restricts a query to the raw

tokens of its own chunk and the registers of earlier chunks. This mask is the training mask of Gisting and Activation Beacon (Mu et al., 2023; Zhang et al., 2025). Our goal is a design that gives the sink one position, keeps every memory slot free of it, and holds the cache size constant as the context grows.

3 The Slot Audit Protocol

Audited positions and verdict rule. We index cache positions by k, layers by l, and heads by h. We write $\mathcal{L}$ for the validation loss. We examine every register and the position of any prepended sink token. We also examine the first raw token still visible to a query. When the usual sink position is evicted, the sink shifts to this token (Gu et al., 2025; Sun et al., 2024; Wong et al., 2025). The validation loss $\mathcal{L}$ is the mean next-token loss over the raw tokens of every chunk after the first. A query in the first chunk reads no memory slot, because the first register set is written after that chunk. We write Δ for the increase in this loss after an intervention on one position, over the original model. For content, we apply the interchange intervention of causal analysis (Geiger et al., 2021; Vig et al., 2020; Meng et al., 2022) and refer to this intervention as swap. Swap replaces only the key and value that the position computes from the audited document. Thus, the loss rises only if the position stores the information of its chunk. A small content effect shows that the loss does not depend on the content of the position, and the value ratio shows whether its value vector can contribute to the output. A sink acts as a fixed bias with a value norm much lower than the norm of raw tokens (Guo et al., 2024; Gu et al., 2025). We therefore take the norm of v_k relative to the mean value norm of the raw tokens and refer to this ratio as the value ratio. Based on the swap and the value ratio, we assign one of three verdicts to each position. A position is *content* if the swap raises the loss by more than a threshold τ_c, which means that the position stores the information of its chunk, and content-free otherwise. For a threshold τ_v and a margin δ, a content-free position is *housekeeping* if the value ratio is at most $\tau_v - \delta$, which means that the position acts as a sink. The position is *idle* if the value ratio is at least $\tau_v + \delta$, which means that the position does neither. A content-free position with a value ratio inside the margin, between $\tau_v - \delta$ and $\tau_v + \delta$, receives no verdict and is reported as *undetermined*. We select τ_c, τ_v, and δ in §4. We call a memory slot with the housekeeping verdict *annexed*, intended for memory but taken by the sink. The sink tax of a design is the fraction of the memory slots of the design that are annexed.

The memory test: drop and swap. *Drop* masks the KV of a slot at every layer and for every query. Drop is the removal test that prior work applies to the sink position (Xiao et al., 2024; Sun et al., 2024). We call the increase in the loss $\Delta_{\text{drop}}(k) = \mathcal{L}_{\text{drop}(k)} - \mathcal{L}$ the *presence cost* of the slot. Removing a sink raises the loss regardless of whether the sink stores content. Therefore, the presence cost does not enter the verdict. *Swap* replaces the slot at every layer with the KV that the same slot computes on a different document. We call the increase in the loss $\Delta_{\text{swap}}(k) = \mathcal{L}_{\text{swap}(k)} - \mathcal{L}$ the *content effect* of the slot. The *joint content effect* is the increase in the loss when every memory slot is swapped at once.

The sink test: attention and value. The value ratio is $\rho_k = \text{mean}_l \, \|v_k^l\| / \bar{v}^l$, where v_k^l is v_k at layer l and $\bar{v}^l$ is the mean value norm of the raw tokens at layer l. The attention mass $m_k = \frac{1}{NH} \sum_{l,h} \bar{a}_k^{l,h}$ is the proportion of attention that the position receives. Here $\bar{a}_k^{l,h}$ is the attention of head h at layer l on position k averaged over queries, and N and H are the numbers of layers and heads. The sink rate $s_k = \frac{1}{NH} \sum_{l,h} \mathbb{1}[\bar{a}_k^{l,h} > \varepsilon]$ is the proportion of heads with attention on the position above a threshold ε. This proportion is the attention sink metric of prior work (Gu et al., 2025).

4 The Sink Tax: What the Audit Finds

We apply the audit of §3 to a sliding-window model with memory slots, Accumulating ICM under chunk-local masking, and Accumulating ICM with each sink remedy of §2. We train models with 124M parameters from scratch on 2.5B tokens of FineWeb-Edu, following the Llama architecture. The sequence length is 4096 raw tokens, in 4 chunks of C=1024 with R=4 memory slots per chunk. We measure every audit on 64 validation sequences of 4096 tokens and the attention maps on 16 of them. The recipe, per-slot values, further seeds, the 340M and 0.8B audits, and the threshold sensitivity are provided in Appendices B.1 to C.2.

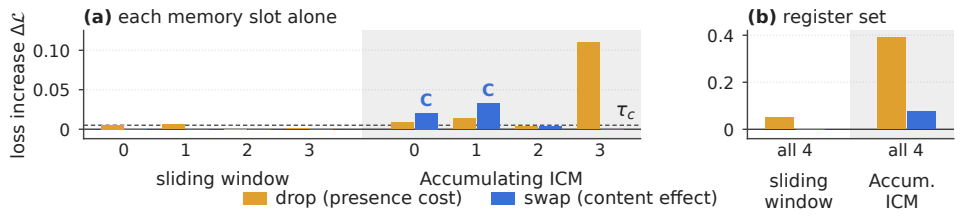


Figure 2: The memory test on the 124M models. Bars give the presence cost and the content effect, (a) per memory slot and (b) for the register set. The dashed line indicates τ_c, C indicates a content effect above it, and the gray area indicates chunk-local masking. There, the memory slot with the largest presence cost has no content effect.

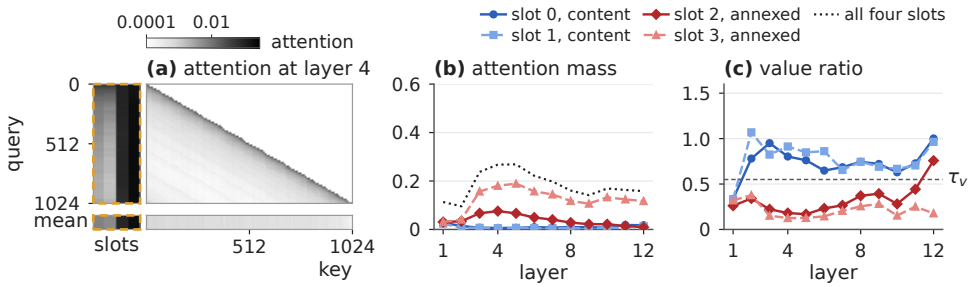


Figure 3: The sink test on Accumulating ICM at 124M. (a) Attention of the last-chunk queries at layer 4, where the dashed rectangle indicates the memory slots of the preceding chunk. (b) Attention mass and (c) value ratio of each memory slot per layer, where the dashed line indicates τ_v. One annexed slot takes most of the attention on the register set and acts as the sink.

All models are pre-trained from scratch under identical settings except for the attention mask and the sink handling. The attention mask is the same in training and evaluation.

Selection of thresholds. The content threshold τ_c separates a content effect from resampling noise. We set τ_c=0.005, the largest change under a repeated audit with a larger evaluation sample or other swap documents (Appendix C.2). Across the 308 memory slots of all our audits, 98% of the slots that store content have a value ratio at or above 0.65. Among the content-free slots, those at or below 0.45 receive a median attention mass of 6.0%, and none at or above 0.65 receives more than 2.8%. Attention does not enter the rule. We place τ_v=0.55 in the middle of the gap with δ=0.10, following the rule of prior work for a distribution of norms (Darcet et al., 2024). For the sink rate, we adopt ε=0.3, the strict choice of prior work (Gu et al., 2025).

4.1 Swapping sliding-window slots has no effect, but removing them raises the loss

We first apply the audit to the memory slots of existing designs under a sliding window that keeps the raw tokens visible, W=C (Kuaishou, 2026). Figure 2 shows the presence cost and the content effect of each memory slot and of the register set. We observe that the content effect is negligible, at most +0.0001 for each memory slot and for all four memory slots swapped at once. In contrast, the presence cost is at most +0.006 for a single memory slot and +0.05 for all four dropped together. This indicates that the loss does not depend on the content of the memory slots, although removing a memory slot raises the loss.

4.2 Masking creates content, but half of the memory slots are annexed

In Figure 2, the joint content effect under chunk-local masking is +0.075, compared to +0.0001 under the sliding window. Two of the four memory slots store content, with content effects of +0.020 and +0.033. The other two have content effects of at most +0.004 and value ratios of 0.33 and 0.21, below τ_c and $\tau_v - \delta$. Thus, both slots are annexed. The presence cost of the annexed slot that takes the most attention is +0.11, while the presence cost of a content slot is at most +0.013. The removal test would therefore identify an

Table 1: Slot audit of three official checkpoints with the official code. Attention denotes the share of the attention of the raw-token queries. With the raw tokens evicted, the sink is on a memory token that receives far more attention than a raw token and more than all other memory tokens together.

			attention (%)	
checkpoint	raw tokens	sink position	sink	other memory
AutoCompressor-Llama-2-7B	evicted	two summary vectors	48.6	3.3
RMT-GPT-2	evicted	first memory token	5.8	4.9
Beacon-Qwen-2-7B	sliding window	first raw token	36.6	5.8

Table 2: Audit of four passive sink candidates and three candidates with a value fixed at zero, under chunk-local masking at 124M. A zero value with a key from the key projection takes the sink duty. Bold marks the best value of a column within a block and underline the second best.

	attention mass (%)		memory slots by verdict		register set	
sink handling	candidate ↑	memory slots ↓	annexed ↓	content ↑	presence cost	content effect
none, Accumulating ICM	0.0	30.4	<u>2</u>	<u>2</u>	+0.390	+0.075
scalar logit	1.7	29.5	<u>2</u>	<u>2</u>	+0.335	+0.075
KV bias	4.7	27.8	3	1	+0.288	+0.071
preserved BOS	8.0	25.1	<u>2</u>	<u>2</u>	+0.119	+0.075
learnable sink token	5.7	28.8	3	1	+0.170	+0.074
KV bias, zero value	4.4	27.9	3	1	+0.297	+0.070
learnable sink token, zero value	<u>49.7</u>	<u>7.9</u>	**0**	<u>2</u>	+0.059	+0.077
KV bias, zero value, projected key	**52.8**	**5.3**	**0**	**3**	+0.072	+0.080

annexed slot as the most important slot. The sink tax of Accumulating ICM is 0.5, which means that the sink takes two of the four memory slots.

4.3 The sink duty shifts to the memory slots when raw tokens are evicted

When the raw tokens are evicted, the annexed slots take the sink duty. Figure 3 shows the attention of the last-chunk queries and the value ratio of every memory slot per layer. We observe that a register set takes an attention mass of 18.3% on average over the visible sets. Specifically, one annexed slot of the set alone takes 12.5% with a value ratio of 0.21. Under the sliding window of §4.1, where earlier raw tokens remain visible, the sink appears on delimiter tokens (Sun et al., 2024). Therefore, the sink tax is a cost of eviction.

Official checkpoints. We also audit the official checkpoints of three prior designs in Table 1. AutoCompressor (Chevalier et al., 2023) and the recurrent memory transformer (Bulatov et al., 2022) evict the raw tokens. In both models, the memory token with the most attention holds the sink. The two summary vectors of AutoCompressor that receive the most attention take 48.6% of the attention, compared to 3.3% for the other 48 together. The first memory token of the recurrent memory transformer receives 65 times the mean attention of a raw token. Swapping either position has no effect on the loss. Activation Beacon (Zhang et al., 2025) keeps a sliding window, where the sink remains on a raw token (Appendix C.1).

4.4 Passive sink candidates lose the sink to the memory slots

We audit four remedies of §2 in Table 2, namely a scalar logit, a KV bias, a preserved BOS token, and a learnable sink token. We call the component that a remedy adds for the sink a passive sink candidate. The scalar logit (OpenAI, 2025) and the preserved BOS token (Xiao et al., 2024) receive 1.7% and 8.0% of the attention, and two memory slots remain annexed as with no sink handling. Even with the KV bias (Gu et al., 2025; Sun et al., 2024) or the learnable sink token (Xiao et al., 2024; Gu et al., 2025; Deng et al., 2025a), three of the four memory slots are annexed. With no sink handling or a passive sink candidate, the memory slot that receives the most attention stores no content that the model uses.

4.5 A zero value and a token-like key take the sink

In every model we train in this section, the position that takes the sink has a value ratio of at most 0.33. A value fixed at zero is the limit of this property. We hypothesize that a candidate with a value fixed at zero takes the sink duty from the memory slots. To verify this, we train the KV bias and the learnable sink token with the values fixed at zero. In the last three rows of Table 2, the zero-valued KV bias takes 4.4% of the attention, and three

memory slots remain annexed. This bias is the key bias of prior work (Gu et al., 2025). In contrast, the learnable sink token with a zero value takes 49.7% of the attention, and no memory slot remains annexed. The presence cost of the register set then decreases from +0.390 to +0.059, because the register set no longer holds the sink. The key of the KV bias is a learnable parameter per head, while the key of the learnable sink token is computed by the key projection. To isolate the key, we produce the key of the zero-valued KV bias by the key projection. This bias takes 52.8% of the attention, and no memory slot is annexed.

Implications for design. Based on the above analysis, we derive two insights for the design of in-context memory. First, a cache position with a value fixed at zero and a token-like key takes the sink duty (§4.5). Second, the memory slots store content only when the raw tokens are evicted (§4.2). An in-context memory design therefore requires the memory slots as the only visible positions of earlier chunks.

5 Assigned-Role Memory

Every design audited in §4 provides memory slots for every chunk and at most one passive sink candidate (§4.4). In these designs, the role of each slot is implicit and emerges during training. When the raw tokens are evicted, the memory slots are the only positions that every query sees (Xiao et al., 2024; Gu et al., 2025). With no sink handling, the sink therefore takes two to three of these slots. Given the insights of §4, we introduce Assigned-Role Memory (ARM), an in-context memory design that assigns one role to every slot in two steps. First, ARM divides a register set into one sink slot with a value fixed at zero and the memory slots (§5.1). Second, with every role fixed, ARM keeps one register set for the entire context and updates the register set after every chunk. Therefore, a context of n chunks needs $G{+}1$ slots instead of nR. The cache size remains constant (§5.2).

5.1 Role assignment: a zero-valued sink slot and the memory slots

Sink slot. We reserve one slot of the register set for the sink and call this slot the sink slot. The sink slot is shared across all chunks (§5.2). The sink slot stores the key and the value of one learnable token, indexed by s. The key is computed like the key of any token, and the value is fixed at zero at every layer l, $v_s^l{=}0$. For a query with attention weights a_k over the visible cache positions k, the head output is $\sum_k a_k v_k$. Thus, the weight a_s on the sink slot contributes $a_s v_s{=}0$. The attention computation is unchanged, and the sink slot is compatible with existing FlashAttention kernels (Dao, 2024). The output gate of prior work (Qiu et al., 2025) removes the sink duty by scaling the head output after the softmax. Moreover, the gate combines with the sink slot (Appendix E.4).

Memory slots. A query in chunk c attends only to the raw tokens of the chunk within its window of W tokens, the memory slots, and the sink slot. The memory slots are therefore the only positions that store the context evicted from the window. The sink slot takes the sink duty (§4.5). A memory slot then lowers the loss only by storing the information of its chunk (§4.2). The sink slot takes the sink from the memory slots, and no memory slot is annexed under either attention mask (§6.1).

5.2 Rewritten set: one set of memory slots for all chunks

ARM keeps one set of G memory slots for all chunks and updates the memory slots after every chunk. We call this set of memory slots the rewritten set. Let x_c denote the C raw tokens of chunk c. Let m denote the $G{+}1$ learnable tokens that the model appends after every chunk with the same embeddings, namely the G memory tokens and the token of the sink slot. Let S_c denote the $G{+}1$ slots that store the keys and values of m at every layer after chunk c. For each chunk, the model writes the slots from the previous slots and the current chunk:

$$S_c = f_\theta\left(S_{c-1}, x_c, m\right), \qquad S_0 = \emptyset. \tag{1}$$

Here f_θ denotes the forward pass with parameters θ. In this pass, the tokens of m attend to x_c and S_{c-1}. Similar to recurrent memory transformers (Bulatov et al., 2022), Eq. 1 is a recurrence over chunks. Let $x_{\mathcal{W}(t)}$ denote the raw tokens of chunk c inside the window of W tokens before token t. The model predicts the next raw token from these tokens and

Table 3: Main results and ablation on PG19 at 124M with C=1024 and W=256. Cache size counts the cache positions that a query reads beyond the window of raw tokens, throughput is in thousand tokens per second at 128k tokens, and + adds one component to the row above. Recall is over all queries and over the queries two updates back, and the 128k column uses 16 test sequences. Bold marks the best value of a column among the windowed designs and underline the second best.

	cache size ↓		throughput	perplexity ↓				long-range perplexity ↓	recall (%) ↑	
design	4k	128k	128k	4k	16k	32k	128k	4k	all	2 updates
Full attention	3840	130816	0.14	82.7	2779.0	6339.4	9079.8	373	100.0	100.0
Sliding window	**0**	**0**	38.7	83.3	83.8	85.8	80.3	745	0.2	0.0
StreamingLLM	4	4	41.5	82.5	83.4	85.3	80.0	738	4.5	12.0
RMT	4	4	40.6	82.8	82.9	84.9	79.8	675	0.3	0.2
Accumulating ICM R=128	384	16256	1.04	83.0	84.6	90.5	252.1	529	53.5	52.2
Accumulating ICM R=4	12	508	20.4	82.4	85.8	93.0	117.0	621	52.9	50.9
+ sink slot	15	635	17.7	81.3	84.6	91.7	123.2	568		
+ rewritten set, ARM	5	5	40.8	81.5	81.5	83.2	78.0	582	51.5	49.1
ARM (G=16)	17	17	40.3	**80.6**	**80.5**	**82.2**	**76.9**	**528**	**78.6**	**75.4**

Table 4: Slot audit of the PG19 models, means over three seeds. Content and annexed count the four memory slots, joint is the content effect of all four, and attention is the share on the register set. Bold marks the best value of a column within a scale and underline the second best.

design	content slots ↑	annexed slots ↓	joint content effect ↑	attention (%)
124M parameters				
Accumulating ICM R=4	1.7	0.7	+0.0508	16.9
+ sink slot	3.0	**0.0**	**+0.0632**	32.1
+ rewritten set, ARM (G=4)	**3.7**	**0.0**	+0.0587	30.1
340M parameters				
Accumulating ICM R=4	2.0	**0.0**	+0.0540	15.0
+ sink slot	2.3	**0.0**	+0.0615	36.2
+ rewritten set, ARM (G=4)	**4.0**	**0.0**	**+0.0616**	33.7

from the slots written after the preceding chunk:

$$p_\theta\left(x_{t+1} \mid x_{\mathcal{W}(t)}, S_{c-1}\right). \tag{2}$$

After S_c is written, S_{c-1} is evicted. A query in chunk c therefore reads G+1 slots beyond the visible raw tokens for every c. In contrast, a query of Accumulating ICM reads the $(c-1)R$ slots of all earlier chunks. Unlike the recurrent memory transformer, which feeds the last-layer output of one segment into the next, ARM carries the rewritten set in the key-value cache of every layer. One attention mask implements Eq. 1 and Eq. 2 for all chunks in a single forward pass, as in Gisting (Mu et al., 2023), at the cost of one standard forward pass. The number of memory slots G is set by the task (§6.3). The mask and the algorithm are provided in Appendix D.

6 Experiments

Setup. We evaluate ARM on Llama-style models with 124M, 340M, and 0.8B parameters, each pre-trained from scratch on PG19 (Rae et al., 2020) under the recipe of Appendix B.1. We compare ARM with a sliding window, StreamingLLM (Xiao et al., 2024), full attention, and the recurrent memory transformer (RMT) (Bulatov et al., 2022). We also compare with Accumulating ICM at R=4 and R=128 memory slots per chunk, the ratio of Activation Beacon (Zhang et al., 2025). ARM keeps G=4 memory slots and one sink slot by default. Every design is pre-trained under identical settings on 4 chunks of 1024 tokens, with a sliding window of 256 raw tokens, as in summary attention (Kuaishou, 2026). **Metrics.** We report the word-level perplexity on the PG19 test set from 4k to 128k tokens. The long-range perplexity is the perplexity on rare tokens with a previous occurrence in an earlier chunk, beyond the window of raw tokens of every design. Recall is the exact-match accuracy on key-value pairs written into one chunk and queried at the end of a later chunk (Arora et al., 2024; Hsieh et al., 2024). The models are trained with such pairs injected into the text.

6.1 Slot audit under assigned roles

In Table 4, no memory slot of ARM is annexed at 124M and 340M, and none at 0.8B (Appendix F.4). On average 1.7 of the four memory slots of Accumulating ICM store content at 124M and 2.0 at 340M, compared to 3.7 and 4.0 for ARM. The joint content effect of the four memory slots of ARM is +0.059 at 124M compared to +0.051 for Accumulating ICM with 12 slots. At 340M the joint content effects are +0.062 and

+0.054. The sink slot keeps every memory slot of the rewritten set free of the sink, but without the sink slot one slot is annexed. The sink slot therefore frees every memory slot of the sink at every scale. With the same cache, the sink slot raises the content slots of Accumulating ICM from 1.7 to 3.0 and lowers its long-range perplexity (Tables 4 and 3).

6.2 Main results on long documents

Evaluation length. In Table 3, ARM achieves a lower perplexity than the sliding window, StreamingLLM, and the recurrent memory transformer at every length up to 128k tokens. With 5 cache positions, ARM maintains a perplexity of 78.0 at 128k tokens compared to 80.3 for the sliding window. Accumulating ICM rises to 117.0 at R=4 and to 252.1 at R=128 while reading 508 and 16,256 memory slots at 128k tokens instead of 12 and 384. Therefore, the accumulated memory degrades beyond the training length, while the rewritten set maintains its perplexity with a constant cache. At 340M and 0.8B, ARM achieves a lower perplexity than the sliding window at every length (Table 26 in Appendix F.4).

Long-range tokens. ARM achieves a long-range perplexity of 582 with 4 memory slots, compared to 745 for the sliding window and 675 for the recurrent memory transformer. With 17 cache positions, ARM (G=16) achieves 528, comparable to Accumulating ICM at R=128 with 384 positions and at R=16 with 48 (Appendix F.8). At 340M and 0.8B, ARM lowers the long-range perplexity of the sliding window as well (Table 26). The advantage holds consistently at every distance up to 15 chunks and compared to a window of 1024 tokens with a matched cache (Appendices F and E.3). Intervals for both metrics are provided in Appendix F.6.

Recall of placed facts. With 17 cache positions, ARM (G=16) recalls 78.6% of the facts, compared to 53.5% for Accumulating ICM at R=128 with 384 positions. With 5 positions, ARM (G=4) recalls 51.5%, comparable to 52.9% for Accumulating ICM at R=4 with 12.

Decoding throughput. On one A6000 GPU with a 40 GB budget, ARM matches the throughput of a sliding window, 40.8 compared to 38.7 thousand tokens per second (Table 3). ARM achieves 39× the throughput of Accumulating ICM at R=128 and 2.0× at R=4. With the same held cache, ARM lowers the long-range loss of a 1024-token sliding window at 2.7× the throughput (Appendix E.3).

6.3 Ablation

Effect of each component. In Table 3, we add the sink slot and then the rewritten set to Accumulating ICM at R=4. The sink slot reduces the long-range perplexity from 621 to 568 and frees every memory slot of the sink (Table 4). The rewritten set then stores every earlier chunk in 4 memory slots instead of 12 at a long-range perplexity of 582. The output gate lowers the loss of the rewritten set further (Appendix E.4). At 128k tokens, the rewritten set reduces the perplexity from 123.2 to 78.0 with 5 positions instead of 635.

Number of memory slots. As G increases from 1 to 16, the long-range perplexity falls to 528 while the cache grows from 2 to 17 positions (Appendix F.8).

7 Conclusion

With a slot audit that swaps the content of every slot, we revealed the sink tax of in-context memory. In existing designs that evict the raw tokens, the attention sink shifts to the memory slots. The memory slot that receives the most attention stores no content that the model uses at 124M, 340M, and 0.8B parameters. Furthermore, under the mask of compression training, the sink takes two to three of the four memory slots per chunk. We proposed ARM, which assigns the sink of the memory slots to one zero-valued slot and one role to every memory slot. With 17 cache positions, ARM matches the long-range perplexity of per-chunk accumulation with 384 positions, recalls 79% of the facts compared to 54%, and maintains its perplexity at 128k tokens. Moreover, ARM keeps the decoding throughput of a sliding window at 128k tokens and achieves 39× that of per-chunk accumulation.

References

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *ICLR*, 2024.

Federico Barbero et al. Why do llms attend to the first token? In *COLM*, 2025.

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

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *NeurIPS*, 2022.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. Sepllm: Accelerate large language models by compressing one segment into one separator. In *ICML*, 2025.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *EMNLP*, 2023.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. Unigist: Towards general and hardware-aligned sequence-level long context compression. *arXiv:2509.15763*, 2025a.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Xinting Huang, Dong Yu, and Zhicheng Dou. A silver bullet or a compromise for full attention? a comprehensive study of gist token-based context compression. In *ACL*, 2025b.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2024. URL https://zenodo.org/records/12608602.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *NeurIPS*, 2021.

Aaron Grattafiori et al. The llama 3 herd of models. *arXiv:2407.21783*, 2024.

Xiangming Gu et al. When attention sink emerges in language models: An empirical view. In *ICLR*, 2025.

Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I. Jordan, and Song Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in llms. *arXiv:2410.13835*, 2024.

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 *COLM*, 2024.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. In *NeurIPS*, 2022.

Dongseong Hwang, Weiran Wang, Zhuoyuan Huo, Khe Chai Sim, and Pedro Moreno Mengibar. Transformerfam: Feedback attention is working memory. *arXiv:2404.09173*, 2024.

Peng Jin et al. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. In *ICLR*, 2025.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *ICLR*, 2024.

Kuaishou. Kwai summary attention technical report. *arXiv:2604.24432*, 2026.

Woosuk Kwon et al. Efficient memory management for large language model serving with pagedattention. In *SOSP*, 2023.

Yuhong Li et al. Snapkv: Llm knows what you are looking for before generation. In *NeurIPS*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *NeurIPS*, 2022.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *NeurIPS*, 2023.

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

OpenAI. gpt-oss-120b and gpt-oss-20b model card. *arXiv:2508.10925*, 2025.

Zihan Qiu et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *NeurIPS*, 2025.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *COLM*, 2024.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. Do long-range language models actually use long-range context? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Jesse Vig et al. Investigating gender bias in language models using causal mediation analysis. In *NeurIPS*, 2020.

Jeffrey T. H. Wong, Cheng Zhang, Louis Mahon, Wayne Luk, Anton Isopoussu, and Yiren Zhao. On the existence and behavior of secondary attention sinks. *arXiv:2512.22213*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

Jingyang Yuan et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *ACL*, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *ICLR*, 2024.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *ICLR*, 2025.

Zhenyu Zhang et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023.

Zayd M. K. Zuhri, Erland H. Fuadi, and Alham F. Aji. Softpick: No attention sink, no massive activations with rectified softmax. *arXiv:2504.20966*, 2025.

Reproducibility statement

The training recipe, the data, the hyperparameters, and the hardware of every model are given in Appendix B.1, and the attention masks of both layouts with the update rule are given in Appendix D. The audit protocol, its thresholds, and the sensitivity of the verdicts to them are given in §3 and Appendix C.2. The definition of the long-range loss, the evaluation lengths, and the recall protocol are given in Appendix F.1 and §6, and the per-seed values behind the main results are given in Appendix F.5.

Use of AI assistance

The authors used a large language model assistant (Claude, Anthropic) throughout this work. It was used to draft and polish text under the authors' direction, to search related work and the official implementations of the baselines, and to produce internal reviews of drafts. One schematic of the appendix was produced with an image generation model (gpt-image-2, OpenAI) from a description written by the authors. All research questions, design decisions, claims, and final wording were set and verified by the authors, who take full responsibility for the content.

Contents of the Appendix

A Extended related work 13
A.1 In-context memory tokens 13
A.2 Recurrent memory 13
A.3 Attention sinks and their remedies 13
A.4 Diagnosing memory tokens 14
B Experimental details 14
B.1 Models, data, and training 14
C Validity of the slot audit and further audits 16
C.1 Verdicts confirmed by key geometry 16
C.2 Soundness of the verdict rule and the thresholds 16
C.3 Sensitivity to the sink threshold and the sequence length 18
C.4 Slot audit under chunk-local masking at 124M, 340M, and 0.8B 19
C.5 Per-slot audit of Accumulating ICM at 124M and 340M 19
C.6 Sink handling under chunk-local masking at 124M 19
C.7 Training dynamics of the audit quantities 20
C.8 Bootstrap intervals of the loss differences between memory designs 21
C.9 Attention maps of the audited 124M models 21
C.10 Further audits of sink positions, attention variants, seeds, and budgets 21
C.11 Replication of the zero-value test with a second seed 25
C.12 Distance decomposition, key geometry, and routing of the memory slots 26
C.13 Training traces of the passive sink candidates 27
C.14 Failure modes of the passive sink candidates 28
D Mechanism of the rewritten set and the sink slot 29

D.1 Attention masks of accumulating and rewritten register sets 29
D.2 Behavior of the sink-slot key across documents 30
E Additional results 30
E.1 Length extrapolation under chunk-local masking 30
E.2 Zero-shot accuracy on downstream tasks 31
E.3 Cache, throughput, and the equal-cache comparison 31
E.4 Comparison with the output gate 34
F Long-document analyses 35
F.1 Definition of the long-range loss 35
F.2 Loss at 128k tokens 35
F.3 Training on sixteen chunks 36
F.4 Results at 340M and 0.8B 36
F.5 Results with three seeds 37
F.6 Intervals of the PG19 comparisons 37
F.7 Slot audit of the PG19 models with three seeds 38
F.8 Set size of the rewritten set on PG19 38
F.9 Recall by the number of facts and by distance 39

A Extended related work

A.1 In-context memory tokens

In-context memory bounds the cache by training a few tokens to replace evicted raw tokens. Gisting compresses a prompt into gist tokens by masking the prompt from every token that follows the gist tokens, and the gist tokens are the only positions of the prompt that later tokens can attend to (Mu et al., 2023). Activation Beacon interleaves beacon tokens with each chunk at a chosen ratio and keeps the beacon activations instead of the raw KV states (Zhang et al., 2025). Summary attention appends one summary token per chunk, and later tokens read the summary once the chunk leaves the sliding window (Kuaishou, 2026). UniGist inserts gist tokens at a fixed ratio and prepends learnable sink tokens with unconstrained values to keep training stable without the initial tokens (Deng et al., 2025a). Titans prepends persistent tokens meant both to hold task knowledge and to absorb the attention the causal mask directs at the initial positions, two roles in one token (Behrouz et al., 2025). In all of these methods the roles of the tokens emerge during training. The methods with prepended tokens expect the sink to appear on the prepended tokens and the content to stay in the memory tokens. We refer to all such tokens as memory tokens.

A.2 Recurrent memory

Recurrent and compressive memory transformers carry a fixed-size memory state across segments (Bulatov et al., 2022; Rae et al., 2020; Munkhdalai et al., 2024). The rewritten set of §5.2 belongs to this family and differs in two respects. First, the sink role is held by a separate zero-valued slot, and without that slot one of the memory slots of the rewritten set is taken by the sink (Appendix C.10). Second, the recurrent memory transformer feeds the last-layer output of its memory tokens to the input of the next segment, whereas the rewritten set is carried inside the key-value cache of every layer, and training therefore remains one forward pass over all chunks under the mask of Appendix D. A released checkpoint of the recurrent memory transformer places the sink on its first memory token (Appendix C.1). Within in-context memory, compressed context memory merges each newly compressed key and value into a memory of constant size by a running average (Kim et al., 2024). The merged memory trails concatenation there as the number of compression steps grows, whereas the rewritten set is rewritten by the model and matches the validation loss of Accumulating ICM (Appendix C.4). Transformer-XL caches the hidden states of

the preceding segment at every layer and reads the cached states without a gradient (Dai et al., 2019), and block-recurrent and feedback transformers carry a fixed set of state vectors across blocks (Hutchins et al., 2022; Hwang et al., 2024). The rewritten set shares the carry at every layer with these designs and differs in the zero-valued sink slot and in the audit of what the carried positions store.

A.3 Attention sinks and their remedies

Attention runs in several heads per layer, and for every query each head spreads one unit of softmax weight over the positions in the cache. The sink arises because a head with nothing to attend to must still place its weight somewhere, and it is measured head by head (Xiao et al., 2024; Gu et al., 2025). The chosen position carries a small or uninformative value, and the allocation therefore changes the output little (Gu et al., 2025; Guo et al., 2024). The sink leaves massive activations in the residual stream of that position (Sun et al., 2024). The sink appears on any fixed position visible to every query, including delimiters and tokens at a fixed offset (Gu et al., 2025; Sun et al., 2024; Wong et al., 2025). Proposed remedies give the sink a dedicated position: retained initial tokens (Xiao et al., 2024), a prepended learnable sink token (Xiao et al., 2024; Gu et al., 2025; Deng et al., 2025a), a learnable key and value per head (Gu et al., 2025; Sun et al., 2024), or a scalar logit in the softmax denominator (OpenAI, 2025). Attention alone therefore cannot identify a sink, because a token that many later queries genuinely need also receives high attention, the heavy hitters that eviction methods keep (Zhang et al., 2023). Separator tokens act as sinks as well, and SepLLM compresses each segment into its separator (Chen et al., 2025), while a rectified softmax removes the sink without a gate (Zuhri et al., 2025).

A.4 Diagnosing memory tokens

Prior work validates memory tokens by task performance with and without the memory tokens (Mu et al., 2023; Zhang et al., 2025; Kuaishou, 2026). This comparison measures dependence rather than content. The one prior content probe trains a separate decoder to reconstruct each snippet from its gist token and reports aggregate accuracy (Deng et al., 2025b). That probe tests recoverability by another network rather than use by the model. A replacement test separates presence from content: Sun et al. (2024) show that massive activations at the sink act as a fixed bias, since substituting their mean over inputs leaves perplexity unchanged while zeroing them destroys it. The interchange intervention, developed to trace facts and biases through heads and layers, swaps an activation for one computed on a different input and measures the output change (Geiger et al., 2021; Vig et al., 2020; Meng et al., 2022; Zhang & Nanda, 2024). No prior work applies the interchange intervention to memory slots together with removal, and §3 builds on this pairing.

B Experimental details

B.1 Models, data, and training

Unless a caption says perplexity, the tables of this appendix report the loss in nats per token. The perplexity of the main text is the exponential of the same loss, word-level where the table says so.

Architecture. Pre-norm transformers with RMSNorm, rotary position embeddings with θ=10^4, SwiGLU MLPs, no bias terms, and tied input/output embeddings over the GPT-2 BPE vocabulary padded to 50,304. Each register slot index has its own learned token embedding appended to the vocabulary, and the per-scale settings are shown in Table 5. Training uses block layout masks, and every audit runs eager attention with explicit dense masks so interventions are exact.

Optimization. AdamW with β_1=0.9, β_2=0.95, weight decay 0.1 on matrix parameters only, gradient clipping at 1.0, batch size 524,288 tokens, and a cosine schedule that decays to 10% of the peak rate after 500 warmup steps, in bf16 autocast.

Data. FineWeb-Edu, the sample-10BT release, tokenized with GPT-2 BPE and one end-of-text token per document, stored as uint16 memmaps. Validation uses a held-out shard of

Table 5: Model scales and the training settings that vary with the scale. All other settings are shared across scales. The 124M and 340M runs use one NVIDIA RTX A6000 with 48 GB, and the 0.8B runs use one NVIDIA A100.

	124M	340M	0.8B
layers	12	24	36
heads	12	16	20
d_{model}	768	1024	1280
d_{ff}	2048	2816	3584
training tokens	2.5B	2.5B	2.5B (5B on FineWeb-Edu)
peak learning rate	6×10^{-4}	3×10^{-4}	2.5×10^{-4}
micro-batch (sequences)	8	4 (full attention: 2)	2
batch (tokens)	524,288	524,288	524,288

60k documents. The PG19 control trains the same recipe on book data (Rae et al., 2020). The validation stream is a contiguous slice of the validation shard. The shard concatenates 60,000 documents of mean length 1050 tokens and median length 635. In this stream 66% of the tokens belong to documents longer than W=1024 tokens, 39% lie more than W tokens into their document, and 13% lie more than 4096 tokens in.

Layout. 4,096 raw tokens per sequence in 4 chunks of C=1024, a sliding window of W=1024, R=4 registers per chunk unless a sweep varies it, and chunk-local masking for the runs labeled chunk-local. The designs with the rewritten set maintain one register set of G slots plus the sink slot. Length extrapolation evaluates the same checkpoints at 8, 16, and 32 chunks. StreamingLLM keeps the first four raw tokens visible to every query, and once its cache is full the evaluation assigns them the positions immediately before the window, as the released implementation does. The recurrent memory transformer keeps 4 memory tokens, feeds their last-layer states after each chunk as the input embeddings of the memory tokens of the next chunk (Bulatov et al., 2022), and is trained under the shared recipe of every design on the same 4 chunks. This assignment leaves the loss at the training length unchanged within 0.0001.

Table 6: Loss of the sliding-window models trained with W=1024 and evaluated with the window restricted to 1024, 512, 256, and 128 raw tokens, on 64 validation sequences at the training length. The increase over 1024 tokens is the loss carried by the tokens beyond the restricted window, and the long-document regime uses the 256-token window.

evaluation window W	1024	512	256	128
FineWeb-Edu, 124M	3.237	3.255 (+0.018)	3.309 (+0.072)	3.421 (+0.184)
PG19, 124M	2.909	2.934 (+0.025)	2.979 (+0.069)	3.046 (+0.137)

Choice of the sliding window for the long-document regime. To choose W for §6.2 we follow the context-restriction protocol of Khandelwal et al. (2018) and Sun et al. (2021): a model trained with W=1024 is evaluated with its window restricted to 512, 256, and 128 tokens, and the increase in the loss measures how much the model uses the tokens that are no longer within the restricted window. Table 6 shows the increase for the sliding-window models on both corpora. Restricting the window to 256 tokens raises the loss by 0.07 on both corpora, an order of magnitude above the seed spans of Appendix C.11, whereas 512 tokens leaves an increase of at most 0.025. The window of 256 tokens therefore places a measurable amount of information beyond the window of raw tokens. We set W=256 for the long-document regime.

Recall task. The recall models of §6.2 are trained on the PG19 sequences with one fact span per sequence. Keys and values are two disjoint sets of 256 vocabulary items sampled once at random, and a fact is one key followed by one value. The fact span is placed at the start of a chunk sampled uniformly from the first three chunks and holds eight facts after a marker token. A first query span follows 64 tokens later, inside the sliding window, and repeats four of the keys. A second query span closes a later chunk sampled uniformly and repeats the other four keys, where the facts are no longer within the sliding window. The loss is the language-modeling loss with a weight of 20 on the value positions of the query spans, and the recipe is unchanged. Recall is the top-1 accuracy of the value that follows a repeated key, with a chance level of 1/256 among the values. We evaluate 512 test sequences, 2048 queries whose facts are no longer within the sliding window, of which 1264,

560, and 224 lie one, two, and three chunks after the fact span. Every recall entry of Table 3 follows this protocol.

Audit protocol. The audit runs forward passes only, with no gradient step, on the trained model. Every loss is the next-token cross-entropy on validation FineWeb-Edu sequences of the training length, the matched-length rule, averaged over the raw tokens of every chunk after the first, because the queries of the first chunk cannot read any register. An increase in the loss Δ is that loss under an intervention minus the same loss of the original model on the same sequences, and absolute loss offsets between samples therefore cancel. Final checkpoints are audited on 64 validation sequences at batch size 1, and intermediate checkpoints, taken at 10/25/50/75/100% of training, on 16. The sequences that supply the swapped keys and values come from a disjoint half of the evaluation sample under a fixed seed, and we report the sensitivity to these sequences and to the sample size in Appendix C.2. The sink threshold is ε=0.3 with sweeps in Appendix C.3.

Per-run configurations. Every run serializes its complete configuration next to its checkpoints, and runs vary only in the following flags, namely layout, sink handling, R, G, seed, and scale. The elementwise gate follows the released implementation (Qiu et al., 2025).

Design labels. The gate variant adds the per-head output gate of prior work (Qiu et al., 2025) to every attention block. Window dropout denotes a sliding-window model that is trained under chunk-local masking on half of the batches. In the tables of verdicts, w denotes a weak slot, an idle slot with a content effect between 0.001 and 0.005.

C Validity of the slot audit and further audits

C.1 Verdicts confirmed by key geometry

Key geometry and delimiter sinks. The key geometry of every audited slot separates the two verdicts, with annexed keys aligned to the query mean and content keys inside the content cone, as Appendix C.10 shows. Even sliding-window models with no registers place their sink duty on document delimiters, and audits that fix k=1 would therefore miss the relocated duty. A public 7B compression model (Zhang et al., 2025) keeps its beacons intact while the sink appears on the first raw token of every window, with an attention share of 0.37 and a value ratio of 0.34. This model keeps a sliding window in addition to the beacons, and the outcome matches §4.1, where the raw tokens hold the sink duty. The sink tax appears once the raw tokens are evicted, as in §4.2.

AutoCompressor. Two further released models were audited with the official code of each model. AutoCompressor (Chevalier et al., 2023) compresses a context of 1536 tokens into 50 summary vectors, evicts the context, and scores a window of 512 raw tokens with the summary vectors in its cache, on 8 documents. The 50 vectors as a group store content, since swapping all of them for the vectors of another document raises the loss by +0.110 and dropping them by +0.085. On 32 documents, the window queries place 51.9% of their attention on the 50 vectors, and 48.6% of it on two of them, 0.286 on one vector and 0.200 on the other, while the remaining 48 vectors share 3.3%. Swapping either of the two vectors alone changes the loss by +0.0006 and +0.0004, with bootstrap intervals inside τ_c, and the value vector of each is the same across documents, with a mean cosine of 0.99 and 0.98 between documents. A position that takes 29% of the attention would show its content in the swap if it stored any, and the two vectors therefore hold the sink duty once the context is evicted. The two vectors keep a value that does not change with the document, the fixed-bias behavior reported for massive activations (Sun et al., 2024).

Recurrent memory transformer. The released GPT-2 checkpoint of the recurrent memory transformer (Bulatov et al., 2022) prepends 16 memory tokens to every segment, and we run the official code on 32 documents of four segments. The first memory token takes 54% of the attention that the segment queries place on the memory, 35 times the median memory token, and has a value ratio of 0.40 compared to 0.92 to 1.19 for the other 15 tokens. The sink therefore appears on a memory token of this design as well. We repeat the audit on the task of the checkpoint, BABILong QA1 with 100 samples, where the memory carries the fact. Replacing the whole memory lowers the accuracy from 100% to 53%. The first memory token again receives the most attention, 12 times the median memory token, and

has the lowest value ratio, 0.50 compared to 1.11 to 1.21. Replacing this token alone has no effect on the accuracy or on the answer loss.

C.2 Soundness of the verdict rule and the thresholds

Reading the audit. The content effect rises with the information a slot stores from its chunk, and a content effect near zero marks a content-free slot. The presence cost rises whenever deleting the slot disturbs the model. A large presence cost together with a near-zero content effect is therefore the signature of a sink, and a large presence cost together with a large content effect is the signature of memory. The value ratio falls as a slot drains toward a fixed bias. A value ratio below τ_v therefore marks a sink, and a value ratio near one marks a slot whose values still resemble raw tokens. Attention mass and sink rate both rise with sink strength. A large value of either therefore marks a strongly adopted sink, and a sink rate of zero at ε=0.3 leaves open a diffuse sink that the attention mass still exposes.

Across the 90 memory slots of 23 final audits at 124M, the attention criterion of §3 and the two interventions of §3, read with a slot counted as a sink when the swap leaves the loss unchanged and the presence cost is large, agree on 95.6% of sink-vs-not calls, and representation signatures agree with the interventions on 85.6%. The disagreements are systematic rather than random. They are drained slots, with a value ratio of 0.2 to 0.4, mass of 3 to 8%, and swap ≈ 0, that signatures flag, that the attention criterion leaves below its per-head threshold (Wong et al., 2025), and that carry a small presence cost. Every one of these slots is housekeeping under the verdict rule of §3, because the swap has no effect on the loss and the value ratio lies at or below $\tau_v - \delta$. The verdict therefore agrees with the signatures, and the low mass and low presence cost of the same slots are the reason that neither the attention mass nor the presence cost enters the verdict. **Simpler diagnostics fail.** Two worked cases from §4 illustrate the failure. A mass-and-loss reading calls the model with the KV bias the best configuration, because the learned column absorbs 4.7% of attention on average and 19% in its peak layer, and the loss improves over the Accumulating ICM baseline, at 3.245 compared to 3.249. The audit reads the same model differently. Content has collapsed into one slot at +0.069 while the other three slots are empty at swap ≤ 0.0001. A drop-only reading fails in the opposite direction. The drop-only reading calls the content-free slots of the sliding setting essential, with drop +0.05 at swap ≈ 0, and it cannot clear popular content tokens, because heavy hitters also hurt when dropped (Zhang et al., 2023). Each single-family diagnostic misclassifies a case the joint rule gets right. A mass-only reading also cannot separate a sink from a heavy hitter.

Why the decision rule is sound. Under softmax the attention a query spends must land somewhere. Deleting an adopted sink therefore redistributes its mass onto whatever remains, and the loss effect of that redistribution is unbounded. Drop therefore measures presence rather than content. Swap holds position, mask, and mass structure fixed and varies only the information the slot computed from its document. Its null therefore has one reading, that the slot's output does not depend on the input. This independence is the defining property of a housekeeping structure, the constant-bias role of the dormant phase in Guo et al. (2024) and Barbero et al. (2025). The verdict rule of §3 therefore admits the swap and the value ratio and excludes the presence cost.

Choice of thresholds. We re-audit the final checkpoints of Accumulating ICM R=4 and of Accumulating ICM R=4, gate variant, with twice the evaluation sample and, separately, with different swap documents, and define $\tau_c = \max_k |\Delta(k) - \Delta'(k)|$ over both interventions and both repetitions, where $\Delta'(k)$ is the increase in the loss measured in the repetition. The largest change is 0.0048, on one content slot under the change of swap documents, and we therefore set τ_c=0.005. Neither repetition changes a verdict or a per-slot swap ordering. A content above τ_c therefore exceeds every resampling change. Across the 11 Accumulating ICM models with no sink handling or with a passive sink candidate at 124M, 340M, and 0.8B, no memory slot has a content effect between 0.005 and 0.01. Every verdict of these 11 models is therefore unchanged for any τ_c in this range. The value threshold follows the rule of cutting a histogram of norms in the gap between its two groups (Darcet et al., 2024). Across the 82 final audits with 308 memory slots, 139 of the 142 slots that store content have a value ratio at or above 0.65, and none lies at or below 0.45. Under chunk-local masking, 39 of the 68 content-free slots lie at or below 0.45, 24 lie at or above

0.65, and 5 lie between. A further 36 content-free slots belong to the sliding-window models, and the remaining 62 belong to the models with a sliding window shorter than the chunk, where 57 lie at or above 0.65. We therefore take τ_v=0.55, the middle of the gap, with a band of δ=0.10 that spans the gap, and a content-free slot inside the band is undetermined and receives no verdict. The four outcomes, content, housekeeping, idle, and undetermined, are disjoint and cover every slot. Two sinks of released models lie below the band as well, the first raw token of the window of the released 7B compression model at 0.34 and the first memory token of the released recurrent memory transformer at 0.40, whose remaining memory tokens lie at 0.92 to 1.19 (Appendix C.1). Under the sliding window 19 of the 36 content-free slots fall inside the band, at 0.46 to 0.65, and those slots remain undetermined. We therefore report no annexed count for the sliding window in §4.1.

Table 7: Threshold-free signature of the verdict groups. Swap, attention, and cross-document cosine are medians over the positions of each group in our models, positions counts the slots of the group, and attention is the share of the query attention on the position. The two sink vectors of AutoCompressor match the annexed group on every quantity.

position	positions	swap	attention (%)	cross-document cosine
annexed	19	+0.0000	6.5	0.997
undetermined	6	+0.0000	2.1	0.998
idle	16	+0.0004	1.0	0.979
content	64	+0.0198	1.0	0.218
AutoCompressor, summary vector 2	1	+0.0006	28.6	0.99
AutoCompressor, summary vector 3	1	+0.0004	20.0	0.98

A signature without the value threshold. A position contributes $a_k v_k$ to the output of a head, and the contribution carries nothing about the document when v_k does not change with the document, whether v_k has drained toward zero or stays a constant vector that acts as a bias (Sun et al., 2024). Table 7 lists three quantities that need no threshold on the norm. A slot that the sink takes is content-free under swap, receives more attention than the other slots, and keeps a state that does not change with the document. In our models the annexed slots have a cross-document cosine of 0.997 and a median attention of 6.5%, compared to 0.218 and 1.0% for the content slots. The two summary vectors of the released AutoCompressor that receive 48.6% of the attention are content-free under swap, receive 539 and 378 times the median attention, and have a cross-document value cosine of 0.99 and 0.98. Under the three quantities both vectors are sinks. In Table 7, the entries of our groups are medians over the slots of the 32 final audits with saved slot states. Attention is the share of the last-chunk queries, and the cosine compares the state of the same slot across documents. For the released models the cosine compares the value vectors across documents, and attention is the share of the attention of the window queries that the vector receives.

Choice of ε. The per-head threshold enters no verdict and sets only the strictness of the sink rate s_k. Gu et al. (2025) note that no principled optimum exists for this threshold, report the metric across a range, and adopt 0.3. We keep this value for comparability. The value is also strict in our setting. A query in the last chunk under chunk-local masking sees at most 1024 raw tokens and 12 registers, and the uniform share of one position is therefore below 0.001. A head that places a mean attention of 0.3 on one position allocates that position three hundred times the uniform share, far above the few percent that a heavy hitter receives. The attention mass therefore captures diffuse absorption and the sink rate captures concentrated adoption, and we report both across ε in Appendix C.3.

C.3 Sensitivity to the sink threshold and the sequence length

Threshold. Sweeping the sink threshold from ε=0.1 to 0.5 changes no verdict, as Table 8 shows. The sink slot is detected at every threshold, with a share of heads that falls smoothly from 90% to 35%, and content slots trigger at none, with a share of heads of at most 0.5% at ε=0.1 and zero above it. The annexed sinks of Accumulating ICM models are the fragile case. Their duty is spread across heads, and they therefore cross the per-head bar only at $\varepsilon \leq 0.2$, with a share of up to 66% of the heads at ε=0.1 that falls to zero by ε=0.3. Detection alone under-counts diffuse sinks. The swap and the value ratio therefore decide the verdict in §3. Across every final audit, the annexed slots carry an attention mass between 1.2% and 21.1% with a median of 6.0%, while no idle slot exceeds 2.8%, and the annexed slots are

Table 8: Share of heads whose attention on a register slot exceeds the sink threshold ε, in percent, for the six audited models of the threshold sweep. The sink slot of ARM is detected at every threshold, the annexed slots of Accumulating ICM at most at $\varepsilon \leq 0.2$, and no content or idle slot at any threshold.

	sink threshold ε				
slot	0.1	0.2	0.3	0.4	0.5
ARM (G=4), 124M, sink slot	90.3	75.0	63.9	45.8	35.4
ARM (G=8), 124M, sink slot	89.6	73.6	57.6	45.8	32.6
Accumulating ICM R=4, 124M, annexed slot 2	0.0	0.0	0.0	0.0	0.0
Accumulating ICM R=4, 124M, annexed slot 3	66.0	15.3	0.0	0.0	0.0
Accumulating ICM R=4, 340M, annexed slot 0	8.9	0.0	0.0	0.0	0.0
Accumulating ICM R=4, 340M, annexed slot 3	45.1	0.8	0.0	0.0	0.0
content slots of all six models, maximum	0.5	0.0	0.0	0.0	0.0
idle slots of all six models, maximum	0.7	0.0	0.0	0.0	0.0

therefore active sinks rather than unattended slots even though attention does not enter the rule.

Table 8 covers 144 layer-head pairs at 124M and 384 at 340M, from the final checkpoints on 64 validation sequences. The six models are Accumulating ICM R=4, its gate variant, ARM (G=4), and ARM (G=8) at 124M, and Accumulating ICM R=4 and its gate variant at 340M, and the working threshold is ε=0.3. The share of heads with a sink on the sink slot falls from 90% to 35% for G=4 and from 90% to 33% for G=8. The annexed slots of the Accumulating ICM models cross the per-head threshold only at $\varepsilon \leq 0.2$ and read as 0 at the working threshold. Slot 3 reaches 66% at ε=0.1 and 15% at 0.2 at 124M, and slots 3 and 0 reach 45% and 9% at ε=0.1 at 340M. Detection alone therefore under-counts diffuse sinks, and the swap and the value ratio decide the verdict.

Length. At twice the training length the audit repeats its verdicts on 4 validation sequences. Per-slot masses and swap orderings are unchanged in both re-audited models. The aggregate register mass of Accumulating ICM R=4, gate variant, grows from 6% to 11% because more register sets become visible, while the sink slot of the rewritten set stays at 42%. The sink metric is length-stable for the rewritten set, and the growth in the accumulating layout is the dilution that the matched-length rule of Appendix B.1 guards against.

C.4 Slot audit under chunk-local masking at 124M, 340M, and 0.8B

Table 9: Slot audit of every design at 124M, 340M, and 0.8B under chunk-local masking, with the cache size of every design. All models of a scale share the recipe, the data, and the token budget. Retained and read denote the cache positions, in addition to the sliding window, that are kept at the training length and that the last chunk reads. Bold marks the best value of a column within a scale among the windowed designs and underline the second best.

		cache size		slot audit	
method	sink	retained ↓	read ↓	annexed ↓	content effect ↑
124M parameters, 2.5B tokens					
Sliding window	none	**0**	**0**	**0**	0
StreamingLLM	kept tokens	4	4	**0**	0
Accumulating ICM R=4	none	16	12	<u>2</u>	**+0.0753**
Full attention	none	3072	3072	0	0
ARM (G=1)	sink slot	<u>2</u>	<u>2</u>	**0**	<u>+0.0702</u>
340M parameters, 2.5B tokens					
Sliding window	none	**0**	**0**	**0**	0
StreamingLLM	kept tokens	4	4	**0**	0
Accumulating ICM R=4	none	16	12	<u>2</u>	**+0.0793**
Full attention	none	3072	3072	0	0
ARM (G=1)	sink slot	<u>2</u>	<u>2</u>	**0**	<u>+0.0767</u>
0.8B parameters, 5B tokens					
Accumulating ICM R=4	none	<u>16</u>	<u>12</u>	<u>2</u>	**+0.1001**
ARM (G=1)	sink slot	**2**	**2**	**0**	<u>+0.0971</u>

Table 9 lists the audit of every design at 124M, 340M, and 0.8B under chunk-local masking, on the same 64 validation sequences of 4096 tokens, with annexed and content effect as defined in §3. At every scale two of the four memory slots of Accumulating ICM are annexed, while no memory slot of ARM (G=1) is annexed. ARM (G=1) reaches this verdict with 2 cache positions instead of 12. One memory slot of ARM carries a content effect of +0.0702,

+0.0767, and +0.0971 at the three scales, close to the +0.0753, +0.0793, and +0.1001 of the four memory slots of Accumulating ICM. A single assigned memory slot therefore stores as much of the context as the four slots per chunk of the accumulating design, with none of them lost to the sink.

C.5 Per-slot audit of Accumulating ICM at 124M and 340M

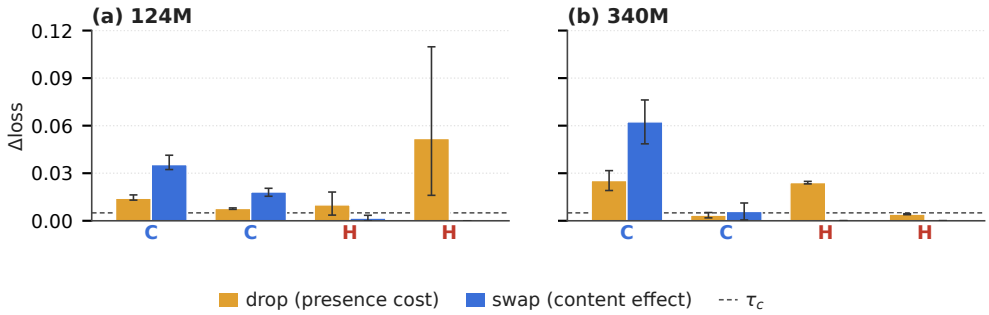


Figure 4: Per-slot audit of Accumulating ICM R=4 at (a) 124M and (b) 340M. Bars are means over three seeds at 124M and two seeds at 340M with the range over seeds, and the slots are sorted by verdict. The letters below the bars denote the verdict of the slot, C for content and H for housekeeping, and the dashed line indicates τ_c. The sink takes two to three of the four memory slots at 124M and 340M.

Figure 4 shows the per-slot audit of Accumulating ICM R=4 at 124M and 340M under chunk-local masking. Bars show the mean over seeds on 64 validation sequences with the range over seeds, and the verdicts are CCHH in every seed at 124M and CCHH and CHHH at 340M. Within a panel the slots are sorted by verdict and then by content effect, whereas Appendix C.10 lists the slots in index order. At both scales the sink takes two to three of the four memory slots with every seed. The sink tax therefore does not shrink with scale, and the 0.8B model of Table 9, trained on 5B tokens, shows the same two annexed slots.

C.6 Sink handling under chunk-local masking at 124M

Table 10: Sink handling under chunk-local masking at 124M. Annexed counts the four memory slots, register set and candidate are attention shares, where the register set of the sink-slot row includes the sink slot, and $\Delta\mathcal{L}$ is the loss relative to no sink handling, whose three-seed span is 3.2423 to 3.2576. A dash denotes a quantity that the design does not have. Bold marks the best value of a column within a block and underline the second best.

			attention (%)			
sink handling	annexed ↓	content effect ↑	register set	candidate ↑	loss ↓	$\Delta\mathcal{L}$ ↓
no sink handling	<u>2</u>	+0.0753	30.4	–	3.2492	0
scalar logit	<u>2</u>	<u>+0.0754</u>	29.5	1.7	3.2485	−0.0007
KV bias	3	+0.0708	27.8	4.7	<u>3.2454</u>	<u>−0.0038</u>
preserved BOS	<u>2</u>	+0.0747	25.1	<u>8.0</u>	3.2485	−0.0007
learnable sink token	3	+0.0735	28.8	5.7	3.2465	−0.0027
sink slot	**0**	**+0.0775**	45.8	**25.3**	**3.2400**	**−0.0092**

Table 10 repeats the audit of §4.4 with the validation loss of every remedy. All rows use chunk-local masking with R=4 accumulating registers, the same recipe and 2.5B tokens, and the audit on 64 validation sequences, with annexed and content effect as in Table 9 and attention in percent. The passive sink candidates receive 1.7% to 8.0% of the attention and leave two or three memory slots annexed, and their loss stays within 0.004 of the design with no sink handling, inside its seed span of 0.0153. The sink slot receives 25.3% of the attention, leaves no memory slot annexed, and reduces the loss by 0.0092, the largest change in the table.

C.7 Training dynamics of the audit quantities

Figure 5 tracks the audit quantities of Accumulating ICM R=4 across training at 124M under chunk-local masking. Panel (a) shows the validation loss of Accumulating ICM and ARM (G=4) against training tokens, and panels (b) to (d) the content effect, value ratio,

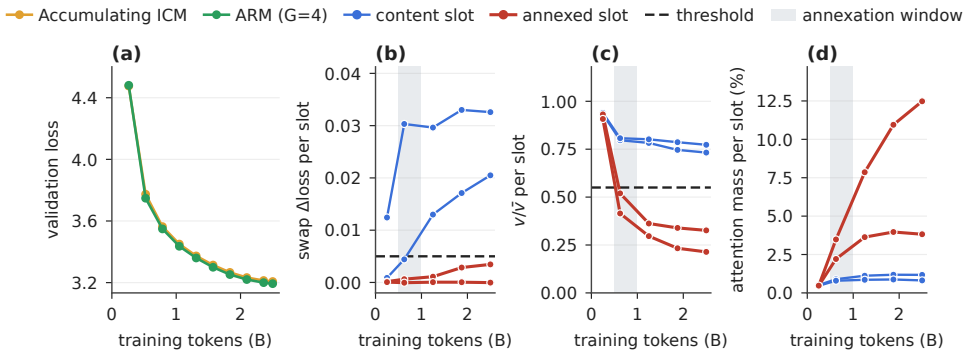


Figure 5: Training dynamics at 124M under chunk-local masking. (a) Validation loss against training tokens. (b) to (d) Content effect, value ratio, and attention mass of every memory slot of Accumulating ICM R=4 at the five audited checkpoints, seed 0. Dashed lines indicate τ_c in (b) and τ_v in (c), and the gray area indicates the annexation window. The annexed slots never form content.

and attention mass of every slot at the five audited checkpoints for seed 0, with τ_c=0.005 and τ_v=0.55. Intermediate audits use 16 validation sequences and the final audit 64. Content forms in the surviving slots from the first audited checkpoint, and inside the annexation window from 0.5B to 1B tokens the value ratio of the annexed slots falls below τ_v as their attention mass rises. The annexed slots never form content afterwards, and the verdicts are therefore fixed early in training. The two further seeds follow the same course and end with the verdicts CCHH and HCHC.

C.8 Bootstrap intervals of the loss differences between memory designs

We compute paired bootstrap intervals for the differences of validation loss between designs at the training length, over the three chunks after the first, where the memory slots of an earlier chunk are readable. ARM (G=1) reaches a loss of 3.2287 compared to 3.2442 for Accumulating ICM at 124M, and 3.1212 compared to 3.1248 at 340M. Every interval is the 95% interval of the mean difference over 10,000 resamples of the 64 validation sequences of one seed and therefore covers sequence sampling, while the seed spans of Appendix C.11 cover training randomness. The lead of ARM (G=1) over Accumulating ICM excludes zero at both scales, from −0.0194 to −0.0114 at 124M with a lower loss on 83% of the sequences, and from −0.0067 to −0.0004 at 340M with a lower loss on 62%.

C.9 Attention maps of the audited 124M models

Figure 6 shows the attention maps of Figure 3 at a second layer and for the gate variant and the rewritten set, with every register column of every visible register set. The maps come from the final checkpoints on 16 validation sequences, averaged over heads, and layers are counted from 1. Panel (a) uses a log scale for every register set visible to the queries. The accumulating rows show the 3 sets written after the earlier chunks, 12 positions, and the row of ARM shows the single rewritten set of 5 positions. Color is the verdict of the slot under the rule of §3, and the sink slot, whose value is fixed at zero, shares the annexed color with a diamond marker. The dotted lines are the mean over the 1024 raw-token positions of the last chunk at the two layers, 4 to 9×10^{-4}. In panels (b) and (c), the raw tokens are mean-pooled 4×4, and all maps share one log color scale. The register set is a strip with one position per slot and verdict-colored caps. The text in each map names the register position with the largest mean and gives its value. The accumulating maps omit the two earliest register sets, which receive 18% of the query mass at layer 4 and 8% at layer 8. The maps of the gate variant and of the rewritten set contain 98 to 100%. The annexed slot 3 of the Accumulating ICM model receives 0.19 of all attention at layer 4 and 0.13 at layer 8, roughly 190 to 370 times a raw-token position. Under the gate variant the largest register position receives 0.02 and 0.03. With the rewritten set the sink slot receives 0.58 and 0.20 while the content slots stay between 10^{-3} and 2×10^{-2}.

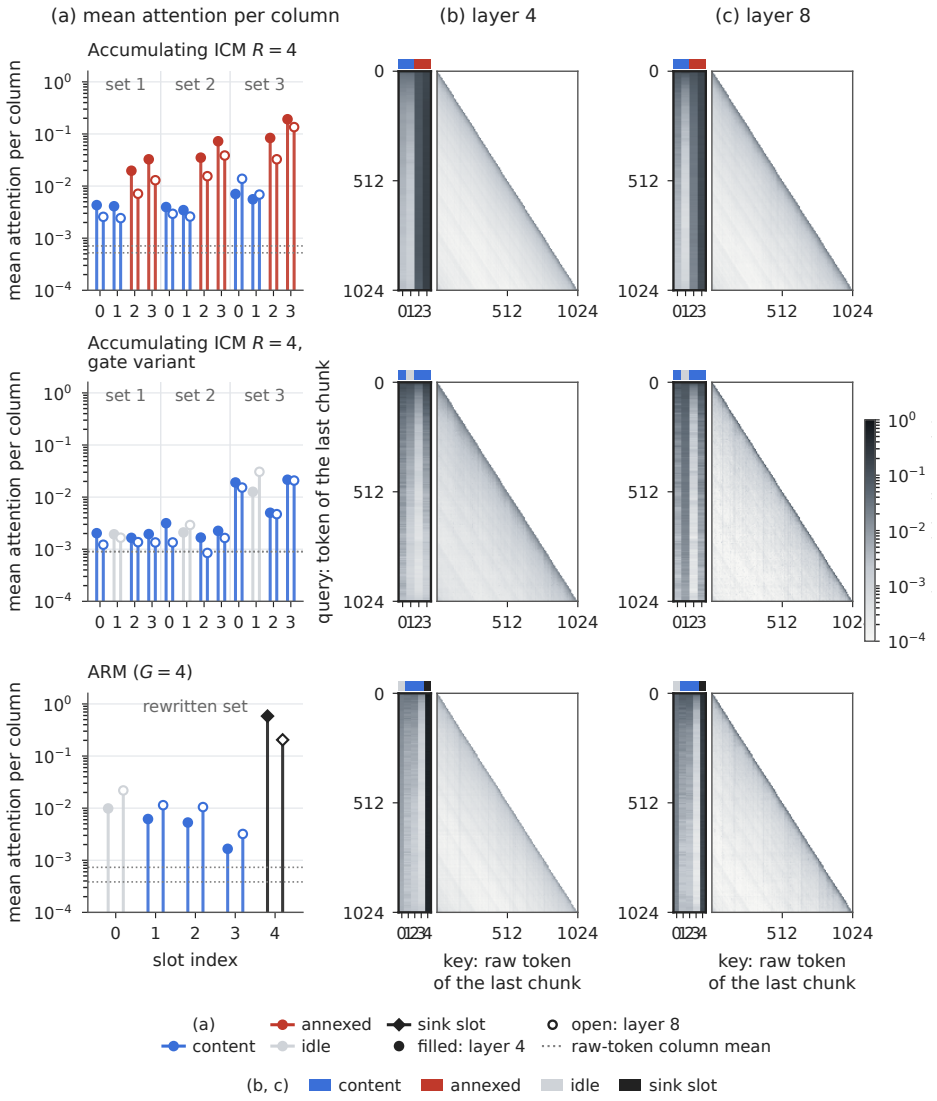


Figure 6: Attention of the last-chunk queries in three 124M designs at two layers, averaged over heads and over 16 validation sequences, at layer 4 where the attention mass on the register set peaks and at layer 8 in the later half. Rows are Accumulating ICM R=4, its gate variant, and ARM (G=4). (a) Mean attention per register position, grouped by the register set written after each chunk. (b, c) Attention maps over the register set of the preceding chunk, whose verdict colors are marked above the strip, and the raw tokens of the last chunk. With the rewritten set, the sink slot receives the attention that an annexed slot receives in Accumulating ICM.

C.10 Further audits of sink positions, attention variants, seeds, and budgets

Delimiter and EOT sink duty. A full-position mass scan, the mean attention per cache column over token queries on 8 validation sequences, finds that the top non-register columns are document-boundary tokens in every model with no sink handling, at 5 to 9× the uniform share. As a group, EOT columns carry 2.2% of all attention in the Accumulating ICM R=4 models, identical at 124M and 340M, and 4.6% in the sliding window. When the registers are removed, the delimiters absorb twice the sink duty, as Table 11 shows. The duty therefore relocates rather than disappears.

The three designs of Table 11 are Accumulating ICM R=4 at 124M and at 340M and the sliding window at 124M. Masses come from the final checkpoints on 8 validation sequences.

Table 11: Attention on the positions of three designs with no sink handling, from the final checkpoints on 8 validation sequences, averaged over layers, heads, and the token queries of every chunk but the last. The first two columns give the highest and the tenth-highest non-register position as a multiple of the uniform share $1/T$, where every such position coincides with an end-of-text token in one of the eight sequences, and the last two columns give the share of attention on all register positions and on all end-of-text tokens. A dash denotes a quantity that the design does not have. Removing the registers doubles the sink duty that the end-of-text tokens take, and no raw position takes more than ten times the uniform share.

	raw position, multiple of the uniform share		share of attention (%)	
design	highest	tenth	registers	end-of-text
Accumulating ICM R=4, 124M	5.0	4.6	30.0	2.2
Accumulating ICM R=4, 340M	5.2	4.6	29.4	2.2
Sliding window, 124M	9.1	7.7	–	4.6

The uniform share is $1/T$, and each of the highest positions coincides with an end-of-text token in one of the eight sequences. The top delimiter positions receive 4.6 to 5.2× the uniform share when registers are present and 7.7 to 9.1× without them. End-of-text tokens receive 2.2% of attention in both Accumulating ICM models compared to 4.6% in the model with no registers.

Grouped-query attention. We repeat the audit with 12 query heads and 4 key-value heads at 124M under the setting of §4. Under Accumulating ICM, two of the four memory slots are annexed, with value ratios of 0.17 and 0.16 and 10.4% and 7.1% of the attention, and the joint content effect is +0.0729. Under ARM (G=4), the sink slot receives 42.1% of the attention with a content effect of 0.0000, all four memory slots store content with content effects of +0.0087 to +0.0160, and the joint content effect is +0.0782.

Table 12: The memory slot that receives the most attention in every model with no sink handling or with a passive sink candidate. Values are means over seeds. The content effect of this memory slot lies below τ_c in every model. Bold marks the best value of a column within a block and underline the second best.

design	attention (%)	content effect ↑	presence cost
Accumulating ICM, no sink handling			
124M	9.7	0.0000	+0.0519
340M	9.1	0.0000	+0.0300
340M, 7B tokens	14.6	+0.0002	+0.0551
0.8B	13.0	+0.0001	+0.0527
124M, R=5	6.5	+0.0002	+0.0155
124M, R=16	4.8	0.0000	+0.0133
124M, R=64	4.2	**+0.0004**	+0.0156
124M, PG19	3.4	+0.0001	+0.0030
124M, PG19, R=128	0.4	−0.0001	+0.0002
340M, PG19	3.0	<u>+0.0003</u>	+0.0021
124M, grouped-query attention	10.4	0.0000	+0.0299
Accumulating ICM with a passive sink candidate, 124M			
scalar logit	11.1	**+0.0002**	+0.0793
KV bias	9.2	<u>+0.0001</u>	+0.0347
preserved BOS	6.9	<u>+0.0001</u>	+0.0250
learnable sink token	6.8	0.0000	+0.0166
KV bias, zero value	9.1	<u>+0.0001</u>	+0.0354
Rewritten set (G=4) without the sink slot, 124M			
FineWeb-Edu	21.1	<u>+0.0001</u>	+0.2422
PG19	6.2	**+0.0009**	+0.0026

Memory slot with the most attention. Table 12 shows the memory slot that receives the most attention in every model of Accumulating ICM with no sink handling or with a passive sink candidate, and in the rewritten set without the sink slot. Values are means over three seeds for Accumulating ICM at 124M and 340M on FineWeb-Edu and PG19, over two seeds for the passive candidates, and over one seed otherwise. The content effect of this memory slot is at most +0.0008 in all 29 models of Accumulating ICM at 124M, 340M, and 0.8B parameters, from 4 to 128 memory slots per chunk, with a second seed of every passive sink candidate, and under grouped-query attention. Without the sink slot, the content effect of this memory slot in the rewritten set is at most +0.0009. The presence cost of the same memory slot is positive in every model.

Seed-wise slot selection. The annexation is deterministic in count and random in identity. Across three seeds of the Accumulating ICM model, two of four slots are annexed every time,

but which two varies: slots $\{2, 3\}$, $\{2, 3\}$, and $\{0, 2\}$. Annexed values drain to 0.21 to 0.33× the token mean, and the complementary slots store the content, +0.016 to +0.041 each.

Per-slot audits across the budget sweep. Per-slot swap at every size of the rewritten set, with the sink slot always at swap ≈ 0 and never annexed, is as follows. At G=1 the single slot stores content at +0.070. At G=2 both slots store content, at +0.023 and +0.036. At G=4 three slots store content, at +0.018, +0.024, and +0.011. No memory slot is annexed at any set size.

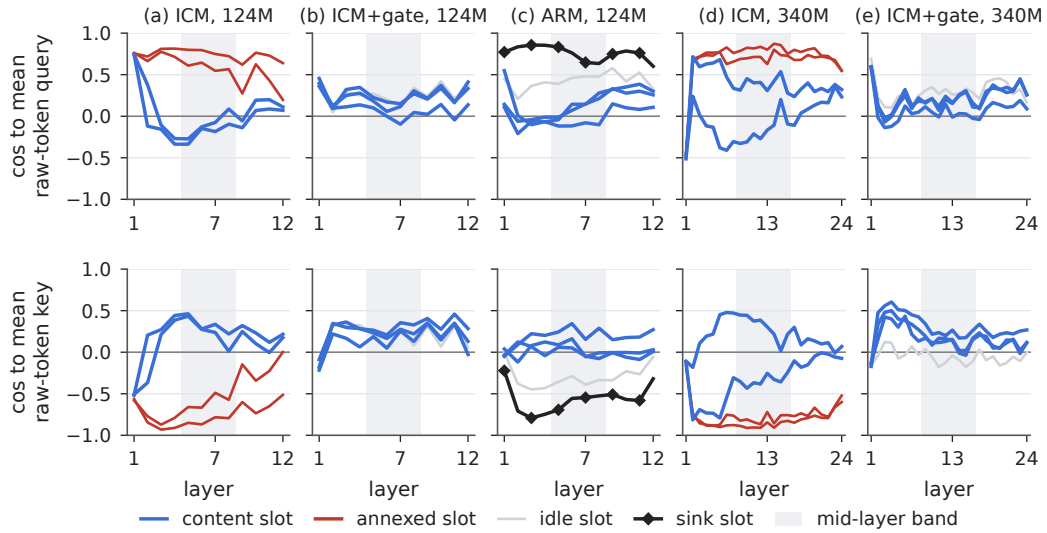


Figure 7: Per-layer key geometry of every register slot in five audited models, where ICM denotes Accumulating ICM R=4, ICM+gate its gate variant, and ARM the rewritten set with G=4. The top and bottom rows show the cosine of the slot key to the mean raw-token query direction and to the mean raw-token key direction. The gray area indicates the middle third of layers. The annexed slots align with the query direction, whereas the content slots keep the geometry of raw-token keys.

Per-layer key geometry by the query-mean method. For every slot we measure the mid-layer cosine between its key and two references. The first is the mean raw-token query direction, a direction that attracts attention regardless of content. The second is the mean raw-token key direction, the content cone. The two verdicts separate cleanly. Slots that are housekeeping under the verdict rule align with the query mean at +0.59 to +0.82 and leave the content cone at −0.60 to −0.88. The content slots of the Accumulating ICM 124M model do the opposite, at −0.19 to −0.10 against the query mean and +0.24 to +0.32 against the cone, and one 340M content slot with a borderline value ratio of 0.56 lies between the two groups. The sink slot, whose value is fixed at zero, adopts the sink geometry at +0.72 against the query mean and −0.58 against the cone, and the alignment sharpens from 124M to 340M, from +0.77 to +0.82 for the most aligned sink. The audit's causal verdicts and the representation geometry give the same slot-by-slot answer, as Figure 7 shows, in line with the reported key geometry of sinks (Gu et al., 2025).

Figure 7 uses the final checkpoints on 8 validation sequences, averaged over heads. Columns (a) to (e) are titled by design and scale, and layers are counted from 1, with 12 layers at 124M and 24 at 340M. The values in the text are averaged over the middle third of layers. The end labels give the slot index, and the sink slot is shown from ARM (G=4).

Per-slot values and readout. Table 13 lists the per-slot values of Figure 2, and Table 14 shows the cosine readout of every slot.

The values come from the final checkpoints on 64 validation sequences, with one verdict letter per slot. C is content with swap > 0.005, H is housekeeping with swap ≤ 0.005 and $\rho \leq 0.45$, w is weak, an idle slot with swap between 0.001 and 0.005, u is undetermined, a content-free slot with ρ between 0.45 and 0.65, i is idle with swap ≤ 0.005 and $\rho \geq 0.65$, and D is the sink slot. Under the sliding window every swap is within the resolution of the audit,

Table 13: Loss increase of each memory slot under the drop and under the swap of the memory test at 124M, with its value ratio. Drop denotes the presence cost, swap the content effect, and ρ the value ratio, listed per slot in index order. Under chunk-local masking two memory slots store all of the content. The verdict letters are C for content, H for housekeeping, i for idle, u for undetermined.

setting	verdict	slot	drop $\Delta\mathcal{L}$	swap $\Delta\mathcal{L}$	ρ
Sliding window, R=4	u	0	+0.0054	+0.0001	0.56
	u	1	+0.0057	+0.0000	0.46
	u	2	+0.0005	+0.0000	0.62
	i	3	+0.0009	+0.0001	0.82
Accumulating ICM R=4	C	0	+0.0081	+0.0205	0.73
	C	1	+0.0132	+0.0326	0.77
	H	2	+0.0036	+0.0035	0.33
	H	3	+0.1098	−0.0000	0.21

Table 14: Readout of the slot states on the final checkpoints without any trained probe. Slots are numbered from 0, the last slot of a design with the sink slot is the sink slot, and the verdict letters are C for content, H for housekeeping, i for idle, u for undetermined, w for a weak slot, and D for the sink slot. Top-1 denotes how often the chunk of the slot is the nearest chunk, in percent, and self-cos denotes the mean cosine between the states of the slot across sequences. Content slots identify their chunk, whereas the annexed slot with the largest attention mass and the sink slot do not. Rows show seed 0. A dash denotes a quantity that the design does not have.

	verdict					top-1 (%)					self-cos				
design	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
124M															
Sliding window, R=4	u	u	u	i	–	4	0	9	56	–	1.00	1.00	1.00	0.98	–
Accumulating ICM R=4	C	C	H	H	–	57	59	60	4	–	0.25	0.23	0.43	1.00	–
ARM (G=4)	w	C	C	C	D	57	56	62	43	1	0.60	0.26	0.30	0.22	0.99
ARM (G=1)	C	D	–	–	–	55	5	–	–	–	0.22	1.00	–	–	–
Accumulating ICM R=4 with the sink slot	w	C	C	i	D	56	57	59	32	1	0.34	0.25	0.29	0.98	0.99
Accumulating ICM R=4 + learnable sink token	H	H	C	H	–	6	57	63	4	–	1.00	0.40	0.26	1.00	–
Accumulating ICM R=4 + KV bias	H	H	C	H	–	4	15	57	3	–	1.00	0.99	0.24	1.00	–
340M															
Accumulating ICM R=4	H	C	C	H	–	9	53	53	2	–	1.00	0.19	0.16	1.00	–
ARM (G=4)	i	i	C	C	D	13	30	52	51	3	0.99	0.88	0.17	0.18	1.00
ARM (G=1)	C	D	–	–	–	53	5	–	–	–	0.17	1.00	–	–	–

and under chunk-local masking the other two memory slots are annexed. In Table 14, the final-layer state of each memory slot on 64 validation sequences is matched by cosine against the mean token state of all 256 validation chunks, with a chance level of 0.4%. A self-cos of 1.00 means an input-independent state. Verdict letters follow Appendix C.10, with D the sink slot and u a content-free slot whose value ratio lies between the two thresholds of §3. The readout uses the final layer while swap covers every layer, and a slot can therefore be content by swap and constant at the final layer. The sliding-window slots carry near-identical states across documents. The sink tax follows directly from the table. Accumulating ICM, sliding-window, and passive-candidate rows never exceed half their slots as content, and the sink slot is housekeeping in every row of ARM, while no memory slot is annexed.

C.11 Replication of the zero-value test with a second seed

Zero-value test with a second seed. We repeat the zero-value test of §4.5 with a second seed. The zero-valued KV bias receives 3.8% of the attention and two memory slots are annexed, compared to 4.4% and three annexed memory slots with the first seed. The zero-valued learnable sink token receives 50.4% of the attention, compared to 49.7%, and no memory slot is annexed with either seed. The joint content effect of the register set is +0.0767, compared to +0.0765. The validation losses of the two seeds are 3.2386 and 3.2419 with the zero-valued learnable sink token and 3.2458 and 3.2469 with the zero-valued KV bias. With a second seed of the four passive sink candidates, the candidate receives 1.5%, 3.7%, 5.9%, and 7.4% of the attention for the scalar logit, the KV bias, the preserved BOS token, and the learnable sink token, and two, two, one, and two memory slots are annexed, with joint content effects of +0.0751, +0.0760, +0.0753, and +0.0768.

Table 15 shows the 124M designs that are absent from Table 9, with the protocol and the columns of that table. The chunk-local model without registers attends only to the raw

Table 15: Validation loss of ARM variants and additional designs at 124M and 340M. No registers denotes the chunk-local mask without register slots, and the learned fifth slot is a fifth register whose value is a learned constant. Every rewritten set from G=1 to G=8 obtains a comparable validation loss. Bold marks the best value of a column within a block and underline the second best.

			cache size		slot audit		validation
method	sink handling	slots	retained ↓	read ↓	annexed ↓	content effect ↑	loss ↓
124M parameters, 2.5B tokens							
No registers	none	0	**0**	**0**	**0**	0	3.2616
Accumulating ICM R=5, learned fifth slot	learned value	5	20	15	4	+0.0711	3.2511
Rewritten set, no sink slot	none	4	4	4	1	+0.0754	3.2526
Accumulating ICM with the sink slot	sink slot	4	20	15	**0**	+0.0775	3.2400
ARM (G=2)	sink slot	2	3	3	**0**	+0.0745	3.2377
ARM (G=4)	sink slot	4	5	5	**0**	+0.0797	**3.2376**
ARM (G=8)	sink slot	8	9	9	**0**	**+0.0816**	3.2395
340M parameters, 2.5B tokens							
ARM (G=4)	sink slot	4	5	5	0	+0.0839	3.1255
340M parameters, 7B tokens							
ARM (G=4)	sink slot	4	5	5	0	+0.0989	2.8946

tokens of the current chunk. The accumulating ARM rows keep one register set per chunk with the sink slot, the set-size rows vary G of the rewritten set with the sink slot.

Table 16: Seed spans of the audit quantities for every design trained with more than one seed. Loss and swap-all are means over the seeds, each followed by its span over seeds, three seeds for Accumulating ICM, for ARM (G=1) at 124M, and for ARM (G=4) at 340M, and two otherwise, and annexed is the mean number of annexed memory slots on each seed. Rows with + add a passive sink candidate to Accumulating ICM. The joint content effect is stable across seeds at 124M under chunk-local masking. A dash denotes a quantity that the design does not have. Bold marks the best value of a column within a block and underline the second best.

	loss ↓		swap-all ↑		annexed ↓
design	mean	span	mean	span	mean
124M					
Accumulating ICM R=4	3.2497	0.0153	+0.0756	0.0012	2.0
+ scalar logit	3.2467	0.0036	+0.0752	0.0003	2.0
+ KV bias	3.2477	0.0045	+0.0734	**0.0052**	2.5
+ preserved BOS	3.2416	0.0139	+0.0750	0.0006	1.5
+ learnable sink token	3.2475	**0.0020**	+0.0751	0.0033	2.5
ARM (G=4)	3.2403	0.0054	**+0.0796**	0.0001	**0.0**
ARM (G=2)	3.2389	0.0023	+0.0748	0.0007	**0.0**
ARM (G=1)	3.2388	0.0110	+0.0692	0.0025	**0.0**
Sliding window, R=4	**3.2378**	0.0098	+0.0001	0.0001	**0.0**
Sliding window	3.2384	0.0028	–	–	**0.0**
340M					
Accumulating ICM R=4	3.1313	0.0037	+0.0785	0.0013	2.3
ARM (G=4)	**3.1277**	0.0089	**+0.0840**	0.0020	**0.0**
ARM (G=1)	**3.1277**	**0.0030**	+0.0779	**0.0023**	**0.0**

Table 16 shows the per-seed loss, joint content effect and annexed-slot count of every design trained with more than one seed, at 124M and 340M. The joint content effect is seed-stable, with spans of at most 0.0052. Table 16 uses the final audits on 64 validation sequences, and the span is the maximum minus the minimum. The spans ground the single-model values of Table 9. The annexed counts vary by one slot at 340M, and under Accumulating ICM the sink takes at least two of the four memory slots with every seed.

C.12 Distance decomposition, key geometry, and routing of the memory slots

Three post-hoc analyses on three 124M checkpoints, namely Accumulating ICM R=4, gate variant, Rewritten set (G=4), gate variant, and ARM (G=8), run on 16 validation sequences under the audit protocol of §3.

Distance decomposition of content. We swap only chunk j's registers with the KV of a different document and read the loss change per query chunk. In all designs the content effect comes from the register set of the preceding chunk, at +0.067 to +0.092. The accumulating layouts keep the older register sets visible, and these sets still receive attention, but the model uses no content from them. The pattern replicates on four further seeds, at +0.064 to +0.096. The rewritten set therefore carries all the content that the accumulating layouts

use. A second intervention replaces the raw tokens of the preceding chunk by a different document and measures what the memory slots carry from that chunk, on 64 validation sequences. The loss on the last chunk increases by 0.070 for Accumulating ICM, by 0.073 with the sink slot, and by 0.078 for ARM (G=4), the largest increase of the designs. With recall queries in the training data, the same rewritten set retains facts across two updates, as shown in §6.2.

Inter-slot key geometry. We measure cosine similarity between register keys, per layer. In the G=8 layout the sink slot and the two idle slots form a block apart from the six remaining slots, three content and three weak, at a mean off-diagonal cosine of 0.30. Across chunks, a content slot's key is largely new each time, with a same-slot cosine of 0.0 to 0.2, while the keys of the sink slot stay at ≈0.5. The role is stable and the content is not, and the audit gives exactly this division.

Head-to-slot routing. We resolve each slot's attention mass by layer and head. No slot is owned by a head, since the top head carries at most 8% of any slot's mass, and the normalized entropy of the head distribution is 0.86 to 0.96 for every slot in all designs. The annexed slots are read most diffusely of all, with an entropy of 0.95 to 0.96 and a top head at 2%. Slots specialize by role and by chunk, yet every head reads every slot.

All three patterns replicate at 340M. The content effect of the preceding chunk grows with scale to +0.066 to +0.102. The keys of the annexed slots stay across chunks at 0.34 to 0.48 while content keys are rewritten at 0.03 to 0.16. **Content follows eviction.** A memory slot learns content only from the queries whose window no longer reaches the context that the slot could summarize. When a query still sees the raw tokens of the previous chunk, the slot adds no information that lowers the loss, and the loss gives the slot no content gradient. The sink role receives a gradient from every query, since the attention weights of every query sum to one. Gisting and Activation Beacon train under masks that hide the compressed text for this reason (Mu et al., 2023; Zhang et al., 2025). The audits follow this rule. With a window of 1024 raw tokens, the width of a chunk, no memory slot stores content, and the content effect of every slot is at most +0.0001 (Figure 2). With a window of 256 raw tokens on PG19, one or two of the four memory slots of Accumulating ICM store content and three or four of ARM (Appendix F.7). Under chunk-local masking, which hides every raw token of earlier chunks, two of the four store content and the sink takes the other two (Figure 2). The number of content slots is therefore set by the context beyond the window. Which slots store it is set by the seed in the accumulating designs, CCHH, CCHH, and HCHC at 124M and HCCH, HHHC, and HHCC at 340M (Table 16). ARM fixes the sink role, and every remaining slot is free for content.

C.13 Training traces of the passive sink candidates

Each account below reads the training trace of one failed candidate at 124M on seed 0 and ties the failure to the finding of §4.5, that a passive sink candidate takes the sink duty only when its value is zero and its key is computed like the key of a token.

A zero value without a token key: the scalar logit. The scalar logit adds one learned score per head to the softmax, with no value and the same score for every query. Its value is therefore zero, as for the sink slot. The score is learned and grows during training, from 0.06 to 0.28 on average over heads and up to 1.41 in one head, and its attention share rises from 0.3% to 1.7% under chunk-local masking. The memory slots rise from 3.6% to 29.5% over the same steps, and two of them stay annexed. In the sliding window, where no slot competes, the share of the scalar reaches 6.8%. A zero value alone therefore does not take the sink duty from the memory slots.

A zero value without a token key: the KV bias. The KV bias adds one learned key and value per head that every query can see. Its key is a free parameter per head, whereas the key of a token is computed by the key projection from the hidden state of the token. The learnable sink token occupies the first position, where the rotary rotation is the identity, and the rotary position therefore does not separate the two keys. To isolate the key, we train the zero-valued KV bias again with its key produced by the key projection from a learned embedding. This bias takes 52.8% of the attention, compared to 4.4% with the free key, no memory slot is annexed, three memory slots store content, and the presence

cost of the register set decreases from +0.297 to +0.072. A free key normalized to the same root mean square takes 14.9% of the attention and leaves three memory slots annexed, and the norm of the key therefore does not account for the difference. A key produced by the key projection is therefore the property that the sink requires in addition to the zero value. Its learned value drains to between 0.04 and 0.06 of the raw-token value in every setting, and fixing the value at zero leaves its share, 4.4% compared to 4.7%, and the three annexed slots unchanged, as shown in §4.5. The value that the bias supplies therefore plays no part in its failure. Under chunk-local masking three memory slots stay annexed in addition to the bias, and with window dropout two do. Under a sliding window the registers store no content with or without the bias, and no slot is annexed.

A learned value slows adoption: the learnable sink token. The learnable sink token is prepended at the first position and sees no context, and its value is therefore a learned constant that cannot store content. During training its value ratio falls from 0.74 to 0.21, while its attention mass rises only from 0.4% to 5.7%. From the first checkpoints the memory slots receive ten times more attention than the token, and at the end they keep 28.8% with three slots annexed. The gradient therefore drains the value toward the sink role, but the memory slots take the sink duty before the drain completes. The same token with its value fixed at zero needs no drain and takes 49.7% of the attention, as shown in §4.5.

C.14 Failure modes of the passive sink candidates

Table 17: Audit of every passive sink candidate in every training setting at 124M. Cand. and reg. denote the attention on the candidate and on the register slots, drop denotes the presence cost of the candidate, and ρ denotes its value ratio. No passive sink candidate keeps more than two content slots in any setting. A dash denotes a quantity that the design does not have. Bold marks the best value of a column within a block and underline the second best.

setting	loss ↓	cand. (%) ↑	drop ↑	ρ ↓	reg. (%) ↓	content ↑	sum ↑	swap-all ↑
scalar logit								
sliding window, R=4	**3.2428**	0.7	+0.002	–	**11.0**	0	0.000	+0.0001
chunk-local	3.2485	1.7	+0.004	–	29.5	**2**	**+0.055**	**+0.0754**
sliding window	3.2458	**6.8**	**+0.085**	–	–	–	–	–
KV bias								
sliding window, R=4	**3.2295**	4.0	+0.035	0.05	**9.1**	0	0.000	0.0000
chunk-local	3.2454	4.7	+0.008	**0.04**	27.8	**1**	**+0.069**	**+0.0708**
sliding window	3.2442	**5.6**	**+0.060**	0.06	–	–	–	–
window dropout	3.2300	4.3	+0.027	0.05	14.5	0	0.000	+0.0030
preserved BOS								
sliding window, R=4	**3.2403**	2.7	+0.004	0.31	**12.0**	0	0.000	+0.0001
chunk-local	3.2485	**8.0**	**+0.013**	**0.22**	25.1	**2**	**+0.058**	**+0.0747**
learnable sink token								
chunk-local	3.2465	5.7	+0.007	0.21	28.8	1	+0.060	+0.0735

Table 17 shows the full audits for every passive candidate in every setting we trained it in. Three patterns repeat. The scalar logit receives 0.7 to 1.7% of the attention where registers are present, and 6.8% in the sliding window. The KV bias takes 4.0 to 5.6% in every setting, and under chunk-local masking the content of the register set collapses to one slot. The preserved BOS and the learnable sink token are themselves drained, with value ratios of 0.21 to 0.31, while the memory slots stay annexed. The mechanistic accounts are in Appendix C.13. Figure 8 plots the candidate adoption and the per-slot verdicts of the same arms against the two designed roles. In Table 17, cand. and reg. are the candidate and the register set of Table 10. The scalar logit has no value vector, and the value of the sink slot is fixed at 0. Content counts the memory slots with swap > 0.005 among the R memory slots, and sum is the sum of their per-slot swap $\Delta\mathcal{L}$, 0.000 when no slot passes the threshold. Swap-all is the joint swap of all register slots and also counts sub-threshold and interaction terms.

The seven designs of Figure 8, named on the axis, are no sink handling, the four passive sink candidates, and the sink slot in the accumulating layout and with the rewritten set (G=4), from the final checkpoints on 64 validation sequences. In panel (a), the design with no sink handling provides no candidate. In panel (b), the verdicts follow the rule of §3, and the sink slot has a value fixed at zero and is not counted. The passive sink candidates

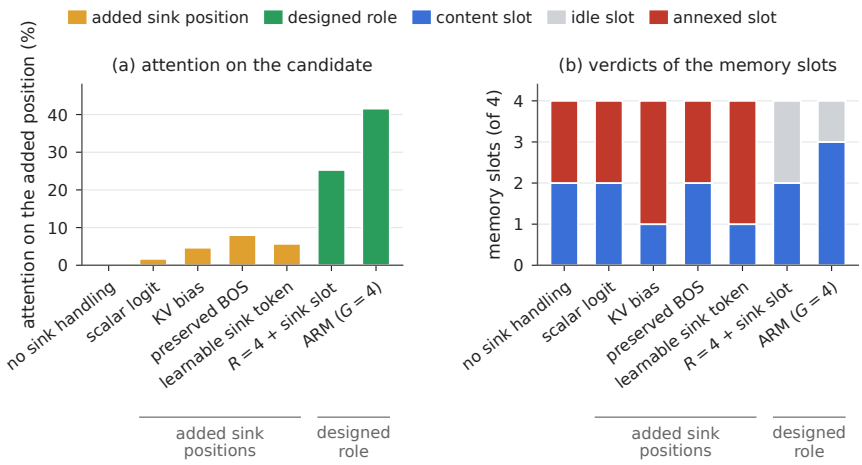


Figure 8: Passive sink candidates against the designed roles at 124M. (a) Mean attention on the passive sink candidate. (b) Verdict of each of the four memory slots, stacked, where the design with no sink handling has no candidate in (a). No memory slot is annexed with the sink slot, whereas the memory slots keep the sink under every passive sink candidate.

receive 1.7 to 8.0% of attention. The KV bias and the learnable sink token collapse the content into a single slot, at +0.069 of +0.071 and +0.060 of +0.074. The sink slot receives 25.3% of attention in the accumulating layout and 41.6% with the rewritten set, and the gate removes the sink duty. In both designs with the sink slot, every value ratio is at or above the threshold and the joint content effect is +0.078 and +0.080.

D Mechanism of the rewritten set and the sink slot

D.1 Attention masks of accumulating and rewritten register sets

Every chunk c occupies C raw positions followed by $R{=}G{+}1$ register positions, the G memory slots and the sink slot, in one sequence with consecutive rotary position indices. In the forward pass of chunk c the raw queries attend only to the raw tokens of chunk c up to their own position and to the R registers written after chunk $c{-}1$. The queries of the first chunk see no earlier set. The register queries of chunk c attend to every raw token of chunk c, to the registers of chunk $c{-}1$, and causally to one another. Their keys and values at every layer form the new set S_c. Once S_c is written, S_{c-1} and the raw tokens of chunk c are evicted, and R cache positions per layer remain in addition to the raw tokens of the chunk in progress. The recurrence gives S_c access to S_{c-1} and to chunk c, and any summary that the accumulating layout keeps in its earlier sets can therefore be folded into S_c.

Sink slot. The sink slot is the last register position of every set. Its value is set to zero at every layer. Its key at every layer is computed from its hidden state like every other register key, and that hidden state attends to the same positions as the memory slots of its set.

Training, positions, and inference. Training and the length evaluation run these dependencies over all chunks of a sequence in one forward pass with the mask of Figure 9, and the position indices continue at longer lengths, and the rewritten set therefore always takes fresh positions in chronological order. The keys of S_c are stored with the rotary positions that follow chunk c, and the content of earlier chunks enters S_c through the hidden states of the memory slots and not through the positions. The content of chunk $c{-}k$ enters S_c through k updates, and the recall experiments of Appendix F cover up to four updates. A query therefore reads the rewritten set at a relative position of at most $C{+}G{+}1$ at every context length, which lies within the relative positions seen during training. Algorithm 1 states the same procedure as a rolling cache at inference. The memory slots read every raw token of the chunk at the rewrite, and with a window shorter than the chunk one request therefore retains up to $C{+}G{+}1$ cache positions.

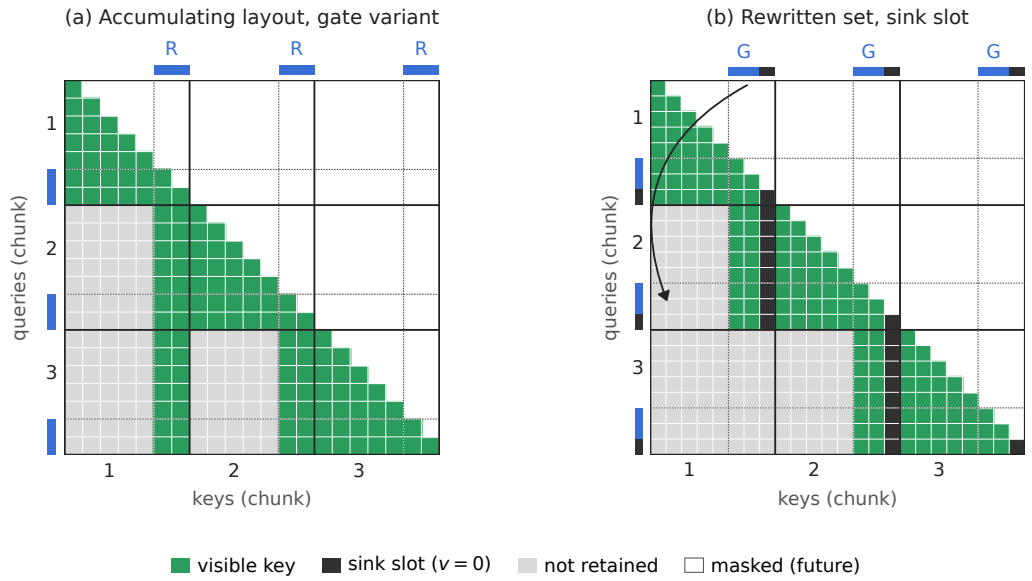


Figure 9: Attention masks with three chunks of five tokens and two memory slots each. Blue bars indicate register positions, and black bars indicate the sink slot. (a) Accumulating layout, where a query attends to every earlier register set. (b) Rewritten set, where a query attends to the latest register set and the sink slot, and the arrow indicates the update of the set from the previous set. The two grids have the same size, and the rewritten set holds one more position per chunk, the sink slot.

D.2 Behavior of the sink-slot key across documents

The value of the sink slot is zero, and the swap of §3 therefore replaces only the key of the sink slot by the key that the same slot computes on a different document. Across the 30 final audits of models with the sink slot, at 124M and 340M, on FineWeb-Edu and PG19, in both layouts, this swap changes the loss by at most 0.0003, with a median of 0.0000. The attention on the sink slot depends on the query, as the attention on any sink does, and the loss does not depend on the document behind the key. Two further controls agree. The zero-valued token of §4.5 is prepended at the first position, sees no context, and takes 49.7% of the attention. A sink slot that attends only to itself still computes its key like the key of a token, from one embedding through every layer. This sink slot reaches the same loss as the sink slot of §5.1 in a run of 0.5B tokens on PG19, 3.4268 compared to 3.4329, with 9.0% compared to 7.0% of the attention on the sink slot.

Algorithm 1 The rewritten set of ARM as a rolling cache at inference.

Require: chunks $x_1, \ldots, x_n$ of C raw tokens; learnable embeddings of G memory slots and one sink slot

$S_0 \leftarrow \emptyset$
for $c = 1, \ldots, n$ **do**
  for each raw token t of x_c in order **do**
    attend to the raw tokens of x_c up to t and to S_{c-1}; predict the next token
  end for
  place the $G{+}1$ slot embeddings at the $G{+}1$ position indices after x_c
  run the slots through every layer; slot j attends to S_{c-1}, to every raw token of x_c, and to slots $1, \ldots, j$ of its set
  set the value of the sink slot to zero at every layer and keep its key
  $S_c \leftarrow$ the keys and values of the $G{+}1$ slots at every layer
  evict S_{c-1} and the raw tokens of x_c
end for

Ensure: at every step the cache holds S_{c-1}, which is $G{+}1$ positions per layer, and the raw tokens of the chunk in progress

In Figure 9a, the accumulating layout is shown for the gate variant, whose per-head output gate replaces the sink slot that Accumulating ICM R=4 with the sink slot appends to every register set. A query attends to its own chunk and to every earlier register set, and earlier raw tokens are not retained. The accumulating layout adds one register set per chunk, whereas the rewritten set keeps the number of visible register positions constant.

E Additional results

E.1 Length extrapolation under chunk-local masking

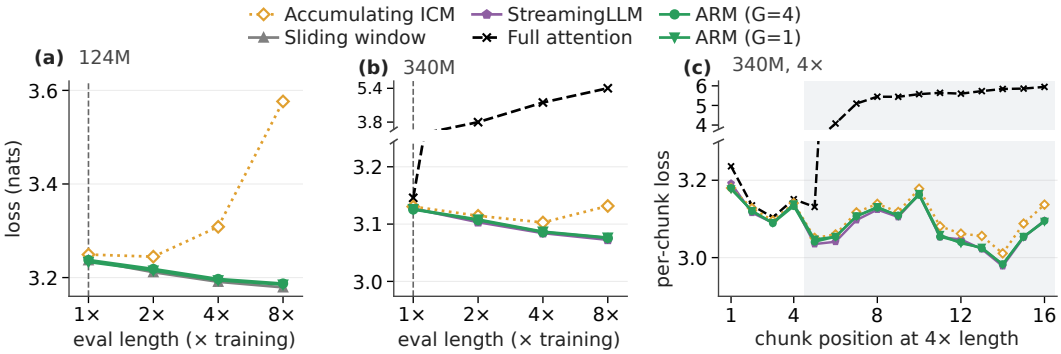


Figure 10: Loss against evaluation length for models trained at four chunks with a window that spans a chunk. (a) 124M and (b) 340M, where the dashed line indicates the training length and the sliding window is shown at 124M only. (c) Per-chunk loss at 4× the training length for 340M, where the gray area indicates the lengths beyond the training length. ARM (G=1) matches the loss of the sliding window beyond the training length with a constant cache size.

Figure 10 evaluates every model trained at four chunks with its own layout on 64 validation sequences at 1× to 4× the training length, and on 16 sequences at 8×. At 124M, ARM (G=1) goes from 3.2333 at the training length to 3.1850 at 8×, within 0.006 of the sliding window at every length, whereas Accumulating ICM rises to 3.5764 and full attention to 5.945. At 340M, ARM (G=1) reaches 3.0764 at 8× compared to 3.1316 for Accumulating ICM, and ARM (G=4) 3.0758. A second seed of ARM (G=1) at 124M follows the first within 0.011 at every length, from 3.2443 at the training length to 3.1930 at 8×. Both set sizes of ARM therefore hold the loss at every length with a constant cache, while the accumulating design trails once its cache outgrows the training length.

E.2 Zero-shot accuracy on downstream tasks

Table 18: Zero-shot accuracy of the final checkpoints on seven standard tasks, where ARC-e and ARC-c denote ARC easy and challenge, HS HellaSwag, LAMB LAMBADA, OBQA OpenBookQA, and WG WinoGrande. Every prompt of these tasks is shorter than one chunk, and the scores therefore measure short-context quality. Assigning the roles and updating a single memory slot have no effect on short-context quality.

design	ARC-e	ARC-c	HS	LAMB	OBQA	PIQA	WG	mean
Full attention	43.1	24.1	29.4	19.6	28.0	60.1	51.2	36.5
Sliding window	43.2	25.3	29.7	20.1	29.6	59.5	51.0	36.9
Sliding window, R=4	43.6	25.1	29.3	19.8	29.8	59.6	51.7	37.0
Accumulating ICM R=4	44.8	24.2	29.9	21.3	32.2	60.8	52.0	37.9
ARM (G=4)	43.4	25.5	29.7	21.1	32.0	61.0	52.8	37.9
ARM (G=1)	44.3	24.4	29.7	21.7	29.4	61.2	51.0	37.4

Table 18 reports the zero-shot accuracy of the final 124M checkpoints on seven standard tasks, evaluated with the released harness (Gao et al., 2024) under its default settings and with normalized accuracy where the harness defines it. ARM (G=4) reaches a mean of 37.9, the same as Accumulating ICM, above 36.9 for the sliding window and 36.5 for full attention, and ARM (G=1) reaches 37.4. Assigning the roles therefore leaves the short-context quality of the model unchanged.

E.3 Cache, throughput, and the equal-cache comparison

Table 19: Cache positions read per decoding step by context length, with C=1024 and a sliding window of 256 raw tokens. The last row is the ratio of Accumulating ICM R=128 to ARM (G=4).

	cache positions read per decoding step					
context length	4k	8k	16k	32k	64k	128k
Full attention	4096	8192	16384	32768	65536	131072
Accumulating ICM R=128	640	1152	2176	4224	8320	16512
Accumulating ICM R=4	268	284	316	380	508	764
Sliding window	256	256	256	256	256	256
ARM (G=4)	261	261	261	261	261	261
ratio, R=128 to ARM	2.5×	4.4×	8.3×	16.2×	31.9×	63.3×

Cache and compute cost. In Table 19, a decoding step loads the sliding window and every position retained in addition to the window. A request of a design with memory slots also retains the raw tokens of the current chunk until the set is written, up to 768 further positions that no step reads. In Table 20, the measured columns are averaged over 8 validation sequences after 2 warm-up passes, with batch size 1, bf16 autocast, and 48 GB of GPU memory. The retained column lists retained registers + sliding window from the layout constants at 1× with 4 chunks and at 4× with 16 chunks. The designs differ in the cache they retain, and the retained cache grows linearly with length for the accumulating design. Compute cost agrees within 4.7% across the five windowed designs, from 895 to 938 ms, in one session of the same script.

Table 20: Evaluation cost of the 340M designs on one NVIDIA RTX A6000, with the retained cache at 4k and 16k tokens. Wall time and peak memory are measured for one forward pass over a sequence of 4096 tokens, T is the number of positions in that pass including the register slots, and the retained cache is analytic, where x+y denotes x register positions and y raw tokens. The compute cost is comparable across the five windowed designs. The headers 4k and 16k are context lengths in tokens.

				retained positions	
method	wall time (ms)	peak memory (GB)	positions T	4k	16k
Full attention	863.5	6.74	4096	0+4096	0+16384
Sliding window	901.6	6.74	4096	0+1024	0+1024
StreamingLLM, 4 kept + window	895.4	6.74	4096	4+1024	4+1024
Accumulating ICM R=4	903.9	6.77	4112	16+1024	64+1024
Accumulating ICM R=4 with the sink slot	937.7	6.77	4116	20+1024	80+1024
ARM (G=4)	927.3	6.78	4116	5+1024	5+1024

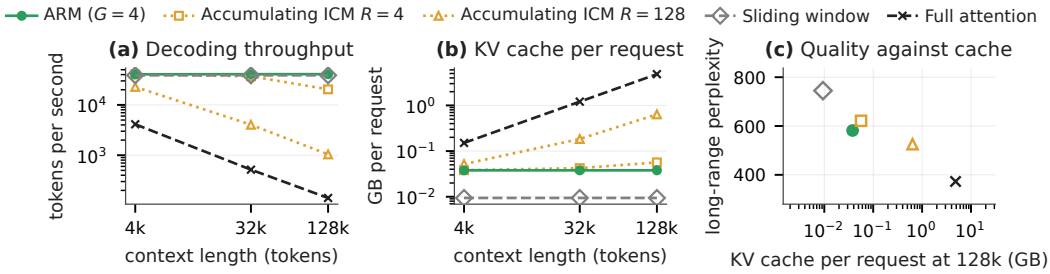


Figure 11: Efficiency and quality of the 124M designs. (a) Decoding throughput and (b) key-value cache of one request against context length, where each point is the median of three measurements. (c) Long-range perplexity at the training length against the key-value cache of one request at 128k tokens. ARM keeps the throughput and the cache of the sliding window and reaches a lower long-range perplexity than per-chunk accumulation at R=4.

Decoding throughput. Figure 11 shows the decoding throughput and the key-value cache of one request against the context length for the 124M designs. Figure 12 follows the protocol of §6.2. At 128k tokens and 340M, ARM achieves 14.1 thousand tokens per second with 395 concurrent requests, compared to 53 tokens per second with 3 concurrent requests for full attention, 264× the throughput. A decoding step at batch size 1 takes 2.7 ms compared to 23.7 ms at 340M, and 1.3 ms compared to 8.9 ms at 124M, with PyTorch scaled-dot-product attention and the layer shapes of each model.

Step time at equal batch. With 8 concurrent requests at 124M, a decoding step of full attention takes 17.2 ms at 32k tokens and 60.4 ms at 128k tokens, whereas a step of ARM

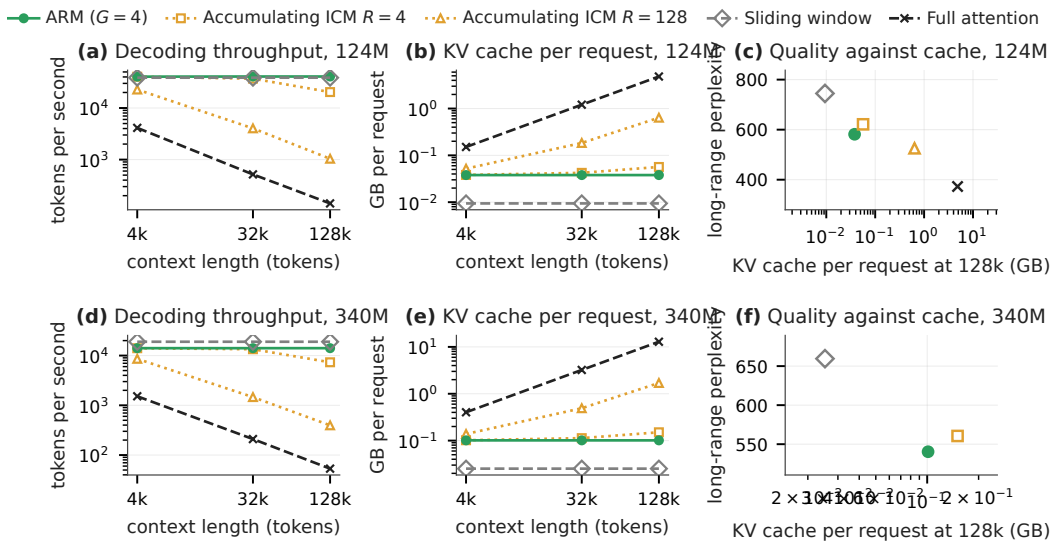


Figure 12: Efficiency and quality for the 124M shape (a) to (c) and the 340M shape (d) to (f). Decoding throughput under a cache budget of 40 GB and the key-value cache of one request against context length, where each point is the median of three repeated measurements, and the long-range perplexity at the training length against the key-value cache of one request at 128k tokens for the designs trained at each scale. ARM keeps the throughput of the sliding window at every length with a lower long-range perplexity at both scales.

takes 1.4 ms at both lengths. With the 340M shape and 32k tokens the step takes 41.7 ms compared to 2.8 ms. The step time of ARM is independent of the context length, since a decoding step reads the same 261 positions at every length. At 128k tokens with the 124M shape, the median of three measurements is 38.7 thousand tokens per second for the sliding window and 40.8 thousand for ARM. The measurement covers the decoding step. In the prefill the rewrite adds $G{+}1$ positions to every chunk of C tokens, 0.5% of the positions at $C{=}1024$ and $G{=}4$.

Table 21: Prefill cost of every cache layout at 4k to 128k tokens with the layer shapes of each model on one A6000 GPU. The last column denotes the seconds for a whole context of 128k tokens, and each value is the median of five measurements. The headers 4k to 128k are context lengths in tokens.

	ms per 1024 raw tokens				128k total
design	4k	16k	32k	128k	s
124M parameters					
Full attention	3.77	7.76	13.06	49.69	6.361
Sliding window	3.03	3.05	3.04	3.04	0.389
StreamingLLM	3.16	3.16	3.16	3.15	0.404
Accumulating ICM ($R{=}4$)	3.15	3.19	3.17	3.31	0.424
Accumulating ICM ($R{=}128$)	3.40	4.00	4.72	9.14	1.171
ARM ($G{=}4$)	3.14	3.18	3.17	3.16	0.404
ARM ($G{=}16$)	3.22	3.18	3.19	3.19	0.409
340M parameters					
Full attention	10.50	19.87	32.56	108.66	13.909
Sliding window	8.79	8.80	8.71	8.66	1.108
StreamingLLM	8.86	8.91	8.89	8.87	1.135
Accumulating ICM ($R{=}4$)	8.96	9.06	9.06	9.44	1.208
Accumulating ICM ($R{=}128$)	9.40	10.64	12.21	21.74	2.783
ARM ($G{=}4$)	9.18	9.17	9.15	9.13	1.169
ARM ($G{=}16$)	8.85	8.91	8.84	8.79	1.125
0.8B parameters					
Full attention	19.09	33.06	51.95	176.10	22.541
Sliding window	16.77	16.78	16.78	16.77	2.147
StreamingLLM	17.09	17.13	17.14	17.14	2.194
Accumulating ICM ($R{=}4$)	19.17	19.20	19.22	19.88	2.545
Accumulating ICM ($R{=}128$)	20.54	23.15	26.39	46.01	5.889
ARM ($G{=}4$)	19.30	19.18	19.18	19.19	2.457
ARM ($G{=}16$)	19.53	19.60	19.54	19.53	2.499

Prefill cost. We time the prefill of a context chunk by chunk with the layer shapes of each model. Every chunk of 1024 raw tokens attends to the window, to the memory positions

of earlier chunks, and to its own register tokens, and the register tokens of the chunk are appended to its queries. Table 21 reports the milliseconds per 1024 raw tokens and the seconds for a whole context of 128k tokens. The cost of ARM per raw token is 3.16 ms at 128k tokens compared to 3.14 ms at 4k with the 124M shape, since every chunk reads the same number of memory positions. Accumulating ICM at R=128 rises from 3.40 ms to 9.14 ms over the same range, and full attention from 3.77 ms to 49.69 ms. With the 340M shape, ARM prefills a context of 128k tokens in 1.17 s compared to 2.78 s for Accumulating ICM at R=128 and 13.91 s for full attention. With the 0.8B shape, ARM prefills the same context in 2.46 s compared to 5.89 s and 22.54 s.

Table 22: Long-range loss and throughput of ARM and of a sliding window of 1024 tokens with the same held cache per request, on PG19 at 124M. Held denotes the cache positions of one request, requests denotes the concurrent requests under a cache budget of 40 GB, throughput is in thousand tokens per second at 128k tokens, and the last columns give the long-range loss on the rare tokens whose previous occurrence lies beyond 1024 tokens. With the same held cache, ARM lowers the long-range loss of the sliding window and raises the throughput. Bold marks the best value of a column within a block and underline the second best.

				loss ↓	long-range loss ↓	
design	held ↓	requests ↑	throughput ↑	4k	4k	16k
Sliding window, W=1024	**1024**	**1059**	15.0	2.9109	6.669	6.676
ARM (G=4)	<u>1029</u>	<u>1054</u>	**40.8**	<u>2.9089</u>	<u>6.512</u>	<u>6.581</u>
ARM (G=16)	1041	1042	<u>40.3</u>	**2.9001**	**6.479**	**6.558**

Sliding window with the same held cache. A request of ARM holds the raw tokens of the chunk in progress in addition to the rewritten set, up to C+G+1 cache positions. We therefore train a sliding window of 1024 tokens under the same recipe. This window holds the same cache per request and corresponds to the segment-level recurrence of Transformer-XL at inference (Dai et al., 2019). As shown in Table 22, with the same held cache, ARM (G=4) lowers the long-range loss of this sliding window by 0.157 at the training length and by 0.095 at 4×. The bootstrap intervals of both differences exclude zero. Moreover, ARM achieves 2.7× the throughput of this window at 128k tokens. This speedup stems from a decoding step that reads 261 cache positions instead of 1024. On all tokens, the loss of ARM (G=4) is comparable to that of the window, 2.9089 compared to 2.9109.

Full attention. The full-attention model is trained at the full length of 4096 tokens under the same budget of 2.5B tokens. Full attention is the reference for the cache that no eviction retains and for the collapse beyond the training length without context extension in Figure 10.

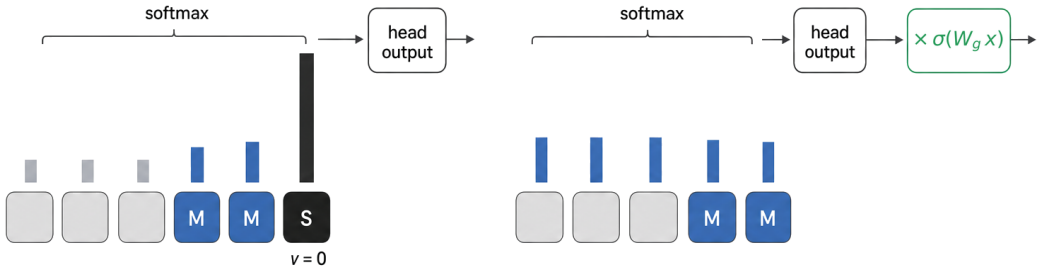
E.4 Comparison with the output gate

The per-head output gate of prior work removes the sink duty without a slot (Qiu et al., 2025). The gate is an option for the sink duty and not a layout, and it combines with the sink slot and with the rewritten set. This appendix compares the two devices in structure, in the audit, and at longer lengths.

Where each device acts. Figure 13 shows one attention head under each device. The sink slot is a key inside the softmax whose value is fixed at zero. Unnecessary attention a_s is allocated to the slot, and the head output is scaled by $1-a_s$. The gate leaves the softmax over the raw tokens and the memory slots unchanged and multiplies the head output by $\sigma(W_g x)$, a factor computed from the hidden state x of the query. A head with nothing to read lowers the factor and needs no position for unnecessary attention.

Audit. We audit Accumulating ICM R=4 with each device at 124M on 64 validation sequences. We find that both devices leave no memory slot annexed. The sink slot receives 25.3% of the attention, which measures the unnecessary attention of the heads directly, whereas the gate folds this attention into a multiplicative factor. The two devices also combine, and with the sink slot and the gate all four memory slots store content.

Length. The accumulating layout degrades at 8× the training length with the gate as well. The gate scales the output of a head after the softmax and cannot remove a key from the softmax. Bounding the cache requires a change of layout, and the rewritten set of §5.2



(a) Sink slot (b) Output gate

Figure 13: Comparison of the sink slot and the output gate in one attention head. Boxes represent the keys of raw tokens, of the memory slots M, and of the sink slot S whose value is zero, and the bars above them represent illustrative attention weights. (a) The sink slot is a key inside the softmax, and the attention it takes scales the head output by one minus its share. (b) The output gate leaves the softmax unchanged and multiplies the head output by a sigmoid factor of the query. The sink slot gives unnecessary attention a dedicated position, and the gate removes the need for one.

Table 23: Loss on PG19 at 124M with the output gate under the three layouts, with C=1024 and W=256. Cache size counts the positions that a query reads beyond the window of raw tokens, and a row with + adds the gate to the row above. The gate lowers the loss of both memory designs at the training length, and the rewritten set alone holds the loss at 128k tokens with a constant cache. The headers 4k to 128k are context lengths in tokens. Bold marks the best value of a column within a block and underline the second best.

	cache size ↓		loss ↓			
design	4k	128k	4k	16k	32k	128k
Sliding window	**0**	**0**	2.9192	2.9188	2.9331	2.8817
Accumulating ICM R=4	12	508	2.9143	2.9375	2.9871	3.1170
+ output gate	12	508	**2.8821**	2.8940	2.9251	3.0032
ARM (G=4)	5	5	2.9089	2.9038	2.9172	2.8660
+ output gate	5	5	2.8831	**2.8797**	**2.8921**	**2.8410**

provides this change with 5 positions instead of 12. Table 23 lists the loss on PG19 with the gate under the three layouts. The gate lowers the loss of both memory designs at the training length, and with the gate ARM reads 5 positions at 128k tokens and holds the lowest loss of the three layouts, whereas per-chunk accumulation reads 508.

Compatibility. The sink slot adds one embedding and leaves every attention block unchanged, whereas the gate rescales the output of every head and requires a model that is trained with the gate. Prior in-context memory is trained on released models that hold no gate (Mu et al., 2023; Chevalier et al., 2023; Zhang et al., 2025), and the sink slot enters this setting as one more register. The rewritten set keeps the cache constant with either device.

F Long-document analyses

F.1 Definition of the long-range loss

Long-range context improves the predictions of language models on PG19 mainly for tokens that occur in that context (Sun et al., 2021). The long-range loss therefore selects the rare tokens whose earlier occurrence only the memory slots store. Let $x_1, \ldots, x_n$ denote the tokens of a sequence, and let $c(t) = \lceil t/C \rceil$ denote the chunk of position t. Let $\mathcal{V}_r$ denote the rare tokens, which are the word-initial tokens with a unigram frequency below 2×10^{-5} in the training set. For a target $x_{t+1} \in \mathcal{V}_r$, let $P_t = \{p \le t : x_p = x_{t+1}\}$ denote the positions of the earlier occurrences. The long-range targets and the long-range loss are

$$\mathcal{T} = \big\{ t : \; x_{t+1} \in \mathcal{V}_r, \; \max P_t \le t - W, \; \min_{p \in P_t} c(p) < c(t) \big\},$$
$$\mathcal{L}_{\mathrm{LR}} = -\frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \log p_\theta(x_{t+1} \mid x_{\le t}). \tag{3}$$

Here the second condition states that no occurrence lies within the sliding window, and the third condition states that one occurrence lies in an earlier chunk. Under the attention masks of §2, a query at position t attends to the W most recent raw tokens, to the memory slots, and to the sink slot. For every $t \in \mathcal{T}$ the raw tokens of the earlier chunks are evicted, and no occurrence lies within the sliding window. The memory slots are therefore the only cache positions that store an occurrence of the target. The sliding window shares every other path with the memory designs, and we report the long-range loss of a design as the paired difference to the sliding window on the same targets, with a bootstrap over sequences. Two controls verify that the difference comes from the memory slots. The first control covers the rare targets with no earlier occurrence. The second control covers the rare targets whose occurrences lie only in the current chunk and are no longer within the sliding window, and no memory slot stores these occurrences.

F.2 Loss at 128k tokens

We evaluate the loss at 128k tokens on 16 sequences of 131,072 tokens from 12 books of the PG19 test set, and every design is evaluated on the same sequences. ARM (G=4) obtains a loss of 2.866 and a word-level perplexity of 78.1, compared to 2.882 and 80.0 for the sliding window and 3.117 and 114.4 for Accumulating ICM at R=4 with cache-relative positions. The 16 sequences come from the longest books of the test set. With a second seed, ARM (G=4) obtains 2.866 and the sliding window 2.887.

Table 24: Long-range loss on PG19 relative to the sliding window at 124M, at 4k and 16k tokens. First occurrence denotes tokens with no earlier occurrence in the sequence, a control that memory cannot help. Earlier occurrence denotes tokens whose previous occurrence lies beyond the window of raw tokens, over all such tokens and over the capitalized names among them. The bootstrap interval excludes zero for every earlier-occurrence cell and for the first-occurrence cell of Accumulating ICM at 16k tokens, and includes zero for the first-occurrence cells of ARM. Bold marks the best value of a column within a block and underline the second best.

	first occurrence	earlier occurrence ↓	
design	control	all	names
4k tokens			
Accumulating ICM R=4	+0.007	<u>−0.185</u>	<u>−0.434</u>
ARM (G=4)	−0.002	**−0.259**	**−0.646**
16k tokens			
Accumulating ICM R=4	+0.039	<u>−0.141</u>	<u>−0.423</u>
ARM (G=4)	−0.002	**−0.154**	**−0.426**

Long-range loss. Table 24 shows the paired differences to the sliding window for every group of targets, on 192 sequences at the training length and on 96 sequences at 4×. At the training length, ARM (G=4) lowers the long-range loss by 0.259 with 5 cache positions, compared to 0.185 for Accumulating ICM with 12, and lowers the loss on capitalized names by 0.646. At 4× the training length, ARM lowers the long-range loss by 0.154 and the loss on names by 0.426 with 5 cache positions, and Accumulating ICM lowers them by 0.141 and 0.423 with 60.

F.3 Training on sixteen chunks

Table 25: Long-range perplexity on PG19 by the number of chunks between a rare token and its previous occurrence. The models are trained on 16 chunks of 1024 tokens and evaluated at 16 chunks, and every cell is the mean over two seeds. ARM achieves a lower long-range perplexity than the sliding window at every distance. The last row gives the number of tokens in each distance bin.

	chunks between the token and its previous occurrence		
design	1 to 3	4 to 8	9 to 15
Sliding window	806	870	952
ARM (G=4)	641	842	905
tokens	21819	8407	2764

We train the sliding window and ARM (G=4) on 16 chunks of 1024 tokens with W=256 and two seeds, and evaluate the long-range perplexity at 16 chunks on 96 test sequences. Table 25 shows the perplexity by the number of chunks between a rare token and its previous

occurrence, means over two seeds, with the rare tokens selected as in Sun et al. (2021). ARM achieves a lower perplexity than the sliding window at every distance. Over all long-range tokens, ARM obtains a perplexity of 707 compared to 833 for the sliding window, and Accumulating ICM at R=4 trained on the same 16 chunks obtains 724 with the first seed while retaining 60 memory slots compared to the 5 cache positions of ARM.

F.4 Results at 340M and 0.8B

Table 26: Main results on PG19 at 340M and 0.8B with C=1024 and W=256, perplexity and long-range perplexity as means over three seeds at 340M except StreamingLLM, and one seed at 0.8B. ARM achieves a lower perplexity than the sliding window at every length at both scales. The headers 4k to 128k are context lengths in tokens. Bold marks the best value of a column within a block and underline the second best.

	perplexity ↓				long-range perplexity ↓	
design	4k	16k	32k	128k	4k	16k
340M parameters						
Sliding window	69.3	74.1	66.9	69.2	660	641
StreamingLLM	67.9	75.3	68.7	71.4	641	624
Accumulating ICM R=4	68.2	72.6	76.8	89.2	561	613
+ sink slot	68.5	73.2	77.7	90.5	548	606
+ rewritten set, ARM (G=4)	67.9	**72.2**	**65.2**	**67.3**	**540**	**568**
ARM (G=1)	**67.7**	**72.2**	**65.2**	67.4	582	590
0.8B parameters						
Sliding window	64.0	64.4	66.0	61.2	525	514
ARM (G=4)	**62.0**	**62.2**	**63.6**	**59.0**	**417**	**447**

Table 27: Slot audit of the PG19 models at 124M and 340M, seed 0. Joint denotes the content effect of all four memory slots together. The sink slot raises the joint content effect, and no memory slot of ARM is annexed. Bold marks the best value of a column within a block and underline the second best.

			content effect ↑					attention
design	content ↑	annexed ↓	slot 0	slot 1	slot 2	slot 3	joint	(%)
124M								
Accumulating ICM R=4	1	**0**	**+0.0262**	+0.0050	+0.0042	−0.0001	+0.0513	17.3
+ sink slot	3	**0**	+0.0111	+0.0113	+0.0039	**+0.0095**	**+0.0623**	32.2
+ rewritten set, ARM (G=4)	**4**	**0**	+0.0106	**+0.0150**	**+0.0089**	+0.0069	+0.0603	30.5
Rewritten set, no sink slot	2	1	+0.0169	+0.0125	+0.0050	+0.0009	+0.0512	13.4
340M								
Accumulating ICM R=4	2	**0**	+0.0028	+0.0124	**+0.0217**	+0.0002	+0.0567	15.1
+ sink slot	2	**0**	+0.0027	**+0.0190**	+0.0033	+0.0066	**+0.0606**	36.9
+ rewritten set, ARM (G=4)	**4**	**0**	**+0.0076**	+0.0077	+0.0127	**+0.0078**	+0.0604	33.6

Table 26 lists the word-level perplexity and the long-range perplexity of the 340M and 0.8B models, means over three seeds at 340M. ARM (G=4) achieves a lower perplexity than the sliding window at every length up to 128k tokens. It obtains the lowest long-range perplexity of every design at the training length and at 4×, with 4 memory slots compared to 12 for per-chunk accumulation. At 128k tokens ARM obtains a perplexity of 67.3 compared to 69.2 for the sliding window, and all four memory slots of ARM consistently store content (Table 27). At 0.8B, ARM (G=4) obtains a perplexity of 62.0 compared to 64.0 for the sliding window at the training length, 63.6 compared to 66.0 at 32k tokens, and 59.0 compared to 61.2 at 128k tokens. It reduces the long-range perplexity from 525 to 417 at the training length and from 514 to 447 at 4×. Three of its four memory slots store content, none is annexed, and the sink slot takes 32.7% of the attention.

F.5 Results with three seeds

In Table 3, perplexity and long-range perplexity are means over three seeds for the sliding window, the recurrent memory transformer, Accumulating ICM at R=4 and at R=128, the sink-slot row, and ARM (G=4), and over one seed otherwise. Recall is the mean over three seeds for ARM (G=4), over two seeds for Accumulating ICM at R=4, for the recurrent memory transformer, and for ARM (G=16), and over one seed otherwise. In the recall runs, ARM (G=4) reaches 51.6%, 51.6%, and 51.2% with three seeds. Table 28 lists the perplexity and the long-range perplexity of the sliding window, Accumulating ICM, the design with the sink slot, and ARM (G=4) as means over three seeds. ARM reduces the

Table 28: PG19 test perplexity and long-range perplexity of the four designs of the main results at 124M, means over three seeds. ARM achieves a lower perplexity than the sliding window at every length. The headers 4k to 128k are context lengths in tokens. Bold marks the best value of a column within a block and underline the second best.

	perplexity ↓				long-range perplexity ↓	
design	4k	16k	32k	128k	4k	16k
Sliding window	83.3	<u>83.8</u>	<u>85.8</u>	<u>80.3</u>	745	<u>733</u>
Accumulating ICM R=4	82.4	85.8	93.0	117.0	621	792
+ sink slot	**81.3**	84.6	91.7	123.2	**568**	775
+ rewritten set, ARM (G=4)	<u>81.5</u>	**81.5**	**83.2**	**78.0**	<u>582</u>	**633**

long-range perplexity from 745 to 582 compared to the sliding window at the training length and from 733 to 633 at 4×.

F.6 Intervals of the PG19 comparisons

Table 29: Long-range loss on the memory bucket at the training length for one checkpoint of every design of the main results, with the paired difference to the sliding window and its 95% interval from a bootstrap over the test sequences. The perplexity column is the exponential of the loss.

design	long-range loss	perplexity	difference	95% interval
Full attention	5.921	373	-0.680	[-0.770, -0.602]
Sliding window	6.601	736		
StreamingLLM	6.615	747	+0.015	[-0.006, +0.035]
RMT	6.505	669	-0.096	[-0.125, -0.068]
Accumulating ICM R=128	6.264	525	-0.337	[-0.403, -0.281]
Accumulating ICM R=4	6.416	612	-0.185	[-0.223, -0.147]
+ sink slot	6.322	557	-0.279	[-0.328, -0.234]
+ rewritten set, ARM	6.341	568	-0.259	[-0.309, -0.215]
ARM (G=16)	6.269	528	-0.332	[-0.385, -0.285]

Table 30: Recall of the main results with 95% Wilson intervals over the pooled queries of the seeds that the table uses. Seeds denotes the number of seeds, and every seed contributes 2048 queries over all distances and 224 queries two updates back. Bold marks the best value of a column within a block and underline the second best.

		all queries		two updates back	
design	seeds	recall (%) ↑	95% interval	recall (%) ↑	95% interval
Full attention	1	**100.0**	[99.8, 100.0]	**100.0**	[98.3, 100.0]
Sliding window	1	0.2	[0.1, 0.6]	0.0	[-0.0, 1.7]
StreamingLLM	1	4.5	[3.7, 5.5]	12.0	[8.4, 17.0]
RMT	2	0.3	[0.2, 0.5]	0.2	[0.0, 1.3]
Accumulating ICM R=128	1	53.5	[51.3, 55.6]	52.2	[45.7, 58.7]
Accumulating ICM R=4	2	52.9	[51.4, 54.4]	50.9	[46.3, 55.5]
+ rewritten set, ARM	3	51.5	[50.2, 52.7]	49.1	[45.3, 52.9]
ARM (G=16)	2	<u>78.6</u>	[77.3, 79.8]	<u>75.4</u>	[71.3, 79.2]

Table 29 gives the long-range loss of one checkpoint of every design of Table 3 with the paired difference to the sliding window and its 95% interval over the 192 test sequences. The intervals of ARM (G=16) and of Accumulating ICM at R=128 overlap, at −0.33 and −0.34 compared to the sliding window. The difference of ARM (G=4) is −0.26 with 5 cache positions, compared to −0.18 for Accumulating ICM at R=4 with 12. Table 30 gives the recall of Table 3 with 95% Wilson intervals over the pooled queries. The interval of ARM (G=16), 77.3 to 79.8, lies above that of Accumulating ICM at R=128, 51.3 to 55.6. The interval of ARM (G=4), 50.2 to 52.7, overlaps that of Accumulating ICM at R=4, 51.4 to 54.4.

F.7 Slot audit of the PG19 models with three seeds

Table 27 shows the slot audit of §3 on the PG19 models of Table 3, together with the rewritten set without the sink slot. The rewritten set without the sink slot follows the recurrence of the recurrent memory transformer (Bulatov et al., 2022) under our recipe, with the memory written at the end of a chunk, read by the next chunk, and carried in the key-value cache of every layer. Under Accumulating ICM, one or two memory slots store content across three seeds, and the memory slot that receives the most attention stores no content with any seed. With the sink slot, three memory slots store content with every seed, and the joint content effect increases from at most +0.052 to at least +0.062. Under ARM

(G=4), no memory slot is annexed with any seed, swapping a single memory slot raises the loss by 0.005 to 0.016, and the sink slot receives 24.7% to 25.8% of the attention. Without the sink slot, the sink takes one memory slot of the rewritten set, and the memory slot with the largest presence cost stores no content. At 340M, two of the four memory slots of Accumulating ICM store content, the memory slot that receives the most attention stores none, and all four memory slots of ARM (G=4) store content with the sink slot at 29.7% of the attention. In Table 27, the audit uses 64 validation sequences of PG19, content and annexed count the four memory slots, and attention denotes the share of the last-chunk queries on the register set.

F.8 Set size of the rewritten set on PG19

Table 31: Set size of ARM on PG19 at 124M, one seed. Cache size counts the memory slots and the sink slot. Word-level perplexity is reported from 4k to 128k tokens, and the long-range perplexity improves with the set size at the training length and at 4× the training length. The headers 4k to 128k are context lengths in tokens. Bold marks the best value of a column within a block and underline the second best.

	cache	perplexity ↓				long-range perplexity ↓	
design	size	4k	16k	32k	128k	4k	16k
Sliding window	0	83.0	83.6	85.6	80.0	736	734
ARM (G=1)	2	<u>81.3</u>	81.5	83.4	78.0	613	660
ARM (G=2)	3	81.5	81.7	83.5	78.1	613	651
ARM (G=4)	5	81.7	81.7	83.5	78.1	568	629
ARM (G=8)	9	<u>81.3</u>	<u>81.3</u>	<u>83.0</u>	<u>77.6</u>	<u>564</u>	<u>619</u>
ARM (G=16)	17	**80.6**	**80.5**	**82.2**	**76.9**	**528**	**597**

Table 31 shows every set size of ARM on PG19. The long-range perplexity improves from 613 with one memory slot to 528 with 16 at the training length, and from 660 to 597 at 4× the training length. With 48 positions, Accumulating ICM at R=16 has a long-range perplexity of 542 at the training length, above the 528 of ARM (G=16) with 17.

Idle slots and the set size. The joint content effect of the set rises from 0.045 with one memory slot to 0.060 with four and 0.071 with 16, and the long-range loss falls in the same order. Accumulating ICM at R=128 shows the same pattern, with 2 of its 128 slots content by swap and a joint content effect of 0.104. The set size G therefore sets how much of the earlier text the set retains.

F.9 Recall by the number of facts and by distance

The recall models of §6.2 follow the associative recall protocol of prior work (Arora et al., 2024; Hsieh et al., 2024) and are trained with four queried facts per sequence. We continue the training of ARM and of Accumulating ICM for 250M tokens with 8, 16, and 32 queried facts per sequence, and we evaluate the 512 test sequences of Appendix B.1. ARM (G=4) recalls as many facts with 4 memory slots as Accumulating ICM with 12, at 2.8 facts of 8 and 3.6 facts of 16.

Table 32: Recall (%) on PG19 by the number of chunks between the fact and the query, with 16 chunks of 256 tokens and one update per chunk. ARM maintains the recall over four updates with a constant cache, and the fed-back output extends the recall to fifteen updates. Bold marks the best value of a column within a block and underline the second best.

	recall (%) ↑			
design	1	2	3	4
Accumulating ICM R=4, all register sets	<u>54.3</u>	**52.2**	49.1	<u>50.0</u>
ARM (G=4), longer distances from the start	**54.7**	<u>51.5</u>	**54.9**	<u>50.0</u>
ARM (G=4), last-layer output fed into the next update	**54.7**	<u>51.5</u>	<u>54.2</u>	**56.0**

Recall by distance. The recall models of §6.2 see a query at most three chunks after the fact. We continue the training of ARM (G=4) and of Accumulating ICM for 1B tokens with 16 chunks of 256 tokens and a sliding window of 64 tokens, where the query follows the fact at longer distances. Table 32 shows the recall by the number of chunks between the fact and the query. ARM recalls 50% to 56% of the facts after four updates of the rewritten set, the same level as after one update, with 5 cache positions. Accumulating ICM recalls the same share at every distance while its cache grows to 60 memory slots at 16 chunks. We

also continue the training of a variant of ARM (G=4) that feeds the last-layer output of the memory slots into the next update, as the recurrent memory transformer does (Bulatov et al., 2022). This variant recalls 53% of the facts placed 1 to 15 chunks before the query with 5 cache positions, compared to 52% for Accumulating ICM. The validation loss of the variant lies within 0.003 of ARM.