Life of Research · Published 26 November 2026 · DOI 10.5281/zenodo.24565011

A Residue Is Not a Recurrence: What 'Reinforcement Learning Alone' Shows About Information Carried in a Streamed KV Cache After Context Compaction, Why Presence, Use, Channel and Benefit Are Separate Claims, and the Re-Prefill and Cache-Swap Controls That Would Tell Them Apart

Pranay Mahendrakar

SONYTECH · ORCID 0009-0003-7224-029X

pranaymahendrakar@sonytech.in · mahendrakarpranay@gmail.com · +91 6361723454

Abstract. Long-horizon language-model agents keep their memory bounded by compaction: past turns are deleted, sometimes replaced by a summary, and the agent carries on. There are two ways to carry on. A re-prefill pipeline rebuilds the cache from the text that survived, so the agent's state is exactly what it can read. A streaming pipeline keeps the cached keys and values of the surviving tokens, which were computed while the deleted turns were still visible, so the state also holds a residue of what was deleted. A September 2026 study makes streaming cheap to train with reinforcement learning, and reports, against an earlier summary-masking method that used supervised fine-tuning first, that reinforcement learning alone is enough for the streamed cache to act as a recurrent state. In this analysis the author reviews and synthesises published results to show that this sentence joins four claims of unequal standing. Presence, that the retained keys and values encode deleted content, is architectural: the earlier method's own toy-model probes find it unchanged across training, and it holds for any eviction that does not recompute. Use, that the policy's outputs depend on that residue, is what training adds, and the streaming study's controlled recall task shows reinforcement learning can add it when a reward scores recall directly and chance success is 20 percent; its supervised arm recalls perfectly from 16 decoded tokens, while its reinforcement-learning arm varies by tens of points across seeds at 16, 32 and 256 tokens. Channel, that the information travels in the residue rather than in visible tokens, is established only for the supervised arm, whose decoded sequence is fixed; the reinforcement-learning arm samples its own sequence, which could carry a code. Benefit, that the residue raises task return, is not shown for agents: matched streaming and re-prefill training runs differ in both directions across strategies, under confounds that prevent crediting the residue, while a 15.3-point loss from removing the residue was measured on a model trained with it. The paper consolidates the measured values in a table and a figure, sets out a ledger of where deleted information can survive, and specifies an attribution procedure with re-prefill, sequence-swap and cache-swap controls that would separate the four claims.

Keywords: context compaction; KV cache; agentic reinforcement learning; long-horizon agents; latent state; chain-of-thought monitorability; steganography; evaluation methodology

SONYTECH ORCID: 0009-0003-7224-029X

1. Introduction

An agent that runs for hundreds of turns cannot keep every observation in its key-value cache. Compaction is the common answer. When a budget is reached, part of the context is removed and, in many designs, replaced by a summary the model writes itself (Packer et al., 2023; Wu et al., 2025a; Lu et al., 2025). Training such agents with reinforcement learning has become an active line: summaries optimised jointly with tool use (Lu et al., 2025), compaction with cross-segment advantage estimation, used in the training pipeline of a large open model (Li et al., 2026b), folding of finished sub-trajectories (Sun et al., 2025), constant-memory internal states (Zhou et al., 2025), overwritten memories for documents of millions of tokens (Yu et al., 2025), and reasoning in fixed-size chunks with a short textual carryover (Aghajohari et al., 2025). Most of these rebuild the context after each compaction from text alone.

Rebuilding is expensive. Every retained token is processed again, and the cost grows without bound as the retained share approaches the budget. Penaloza et al. (2026) remove that cost with KV-streams: the inference engine drops the evicted entries in place, keeps the retained entries as they were computed, and the trainer reproduces the eviction with an attention mask over one continuous trace. They report large wall-clock savings at matched final scores on two text games and on software-engineering tasks. They then make a second claim, which is the subject of this paper. Because a retained entry was computed while the evicted turns were visible, it can carry information about them after they are gone, so the streamed cache can act as a recurrent state. An earlier method that also keeps such entries, MEMENTO (Kontonis et al., 2026), trains with supervised fine-tuning before reinforcement learning, and Penaloza et al. (2026) report that, "contrary to prior work, RL alone is all that is needed for this behavior to emerge."

© 2026 Pranay Mahendrakar. Licensed under CC BY 4.0.

The sentence matters beyond one paper. If streamed caches give agents a recurrent state for free, then the summary an agent writes is no longer the whole of what it remembers. That changes how compaction methods should be compared, how an agent's memory can be audited, and whether a policy trained under one serving regime behaves the same under another. It also matters for the argument within reinforcement learning about when a behaviour must be seeded by imitation and when reward alone can find it.

In this analysis the author reviews and synthesises published results to show that the sentence bundles four claims, and that the evidence supports them to very different degrees. The first is presence: retained entries encode evicted content. The second is use: the policy's outputs depend on that encoding. The third is channel: the dependence runs through the retained entries rather than through visible tokens. The fourth is benefit: the dependence raises the return of an agent on its task. Presence needs no training at all, and the prior work that the streaming study sets itself against says so in as many words. Use is what training adds, and the streaming study's controlled task shows that reinforcement learning can add it under conditions that make it easy. Channel is established in that task only for the supervised arm. Benefit has not been measured for agents in a comparison that holds training fixed.

A natural objection is that this is pedantry about a secondary claim in an efficiency paper. The author takes the objection seriously and answers it in Section 9: the four claims lead to different recommendations for monitoring, for prompt-injection defences that work by deletion, and for moving a trained policy between serving stacks, so conflating them has practical cost. A second objection is that the controls this paper asks for are obvious. They are, and none of them has been reported for the reinforcement-learning arm of the task on which the claim rests.

The paper proceeds as follows. Section 2 describes the two ways of continuing after compaction and the methods on each side. Section 3 reports what the streaming study measured, read from its full text and figures. Section 4 sets out the four claims and a ledger of where evicted information can survive (Table 2). Sections 5 to 8 take presence, use, channel and benefit in turn. Section 9 draws out consequences. Section 10 states an attribution procedure (Algorithm 1) and the predictions of four competing accounts (Table 3). Section 11 states the strongest case against this paper, Section 12 records what is not known, and Section 13 concludes. The analysis draws on 66 works. Numbers attributed to the streaming study, to both papers named MEMENTO and to StreamingLLM are taken from their full texts, figures and appendices.

2. Two Ways to Continue After Compaction

A compaction operator removes a span of the context and may emit a bridge in its place, such as a summary; it fires on a fixed budget or when the model decides to compact (Penaloza et al., 2026; Li et al., 2026a). What happens next divides the field.

Re-prefill. The surviving text, together with any bridge, is placed in a fresh context and encoded again. Each surviving token receives new keys and values computed only from what survived, and new positions. The agent's state after compaction is then a function of visible text alone. This is the regime of nearly every reinforcement-learning method for compaction. The Markovian Thinker resets the context at each chunk boundary and reinitialises it with a short carryover, so that the policy learns to write a textual state near the end of each chunk (Aghajohari et al., 2025). InftyThink trains iterative summarisation by supervised learning (Yan et al., 2025), and InftyThink+ adds trajectory-level reinforcement learning after a supervised cold start (Yan et al., 2026). Reasoning Cache replaces autoregressive decoding with an iterative summarise-and-continue loop in both training and inference (Wu et al., 2026). ReSum condenses search histories with an external tool and adapts GRPO by broadcasting final rewards across segments (Wu et al., 2025a). SUPO derives a policy gradient across summarisation boundaries (Lu et al., 2025), CompactionRL adds token-level normalisation and cross-segment advantage estimation (Li et al., 2026b), Context-Folding collapses finished sub-trajectories into summaries with process rewards (Sun et al., 2025), and Zhang et al. (2026) train a long-document agent to write notes into fresh windows. Training-free variants include a compaction tool paired with a rubric for when to fire (Li et al., 2026a) and compression guidelines refined from failure analysis (Kang et al., 2025).

Streaming. The evicted entries are dropped from the cache, and the surviving entries are kept as they were. Their keys and values were computed while the evicted span was visible, and they keep their original positions. The state after compaction therefore holds more than the visible text: each retained entry is a function of everything it attended to when it was computed. This is the regime of KV-cache eviction for inference, from attention sinks with a rolling window (Xiao et al., 2023) through heavy-hitter retention (Zhang et al., 2023), clustered selection by an observation window (Li et al., 2024) and attention-based omission that treats the transformer as a bounded multi-state recurrent network (Oren et al., 2024). It is also the regime of MEMENTO, which masks each reasoning block once its summary is written, inside one generation call, so that the summary's entries keep what they computed from the block (Kontonis et al., 2026), and of KV-streams, which brings the same property to turn-wise compaction in agentic reinforcement learning (Penaloza et al., 2026).

A third family deserves separate mention because it is where the idea of carrying context in cached activations began. Gist tokens are trained, by a change of attention mask during instruction tuning, so that later tokens see only the gists and the gists' entries compress the prompt (Mu et al., 2023). AutoCompressors pass summary vectors between segments with an unsupervised language-modelling objective (Chevalier et al., 2023); the in-context autoencoder pretrains memory slots with autoencoding and language-modelling losses (Ge et al., 2023); Activation Beacon compresses keys and values directly and is trained with compression-based autoregression (Zhang et al., 2024). Segment-level recurrence in Transformer-XL (Dai et al., 2019) and memory tokens in the Recurrent Memory Transformer (Bulatov et al., 2022) are earlier versions of the same move. In every one of these, the carrying of information through cached state is a trained objective. Gist compression is near-lossless on retrieval-augmented generation and long-document question answering yet struggles on synthetic recall, with failures its auditors name "lost by the boundary, lost if surprise, and lost along the way" (Deng et al., 2024).

Two practical facts separate the regimes beyond the residue. First, positions: re-prefill resets them, streaming keeps them, and StreamingLLM, which streams, deliberately assigns positions within the cache rather than in the original text and calls this "crucial" (Xiao et al., 2023). Positions that keep growing take a rotary-embedding model toward or past its trained range, where extrapolation can produce very large attention scores (Chen et al., 2023). Second, failure behaviour: Penaloza et al. (2026) report that token-level eviction produced degenerate text that needed substantial supervised training to repair, which they attribute, as a suspicion, to removing message-boundary tokens, and so they evict whole turns.

3. What the Streaming Study Reports

This section reports the study's measurements as the version 3 full text gives them.

Setting. The text games use Qwen3-4B-Instruct-2507 and the software-engineering tasks use Qwen3.5-4B. Four compaction strategies are studied: a summary that replaces all prior turns; the Markovian Thinker's deletion of half the context with no bridge; a variant in which the model picks the turns to keep; and a sliding window that keeps almost the whole budget. On TextWorld, compaction fires every 10 turns, episodes run up to 200 turns within a 32k-token budget, and training runs 500 steps at batch size 512. On ALFWorld, compaction fires every 10 turns, episodes run up to 100 turns within 16k tokens, and training runs up to 200 steps at batch 128. On software engineering, compaction fires every 30 turns, episodes run up to 100 turns within 64k tokens, and training runs 200 steps at batch 256, evaluated on SWE-bench Verified. Training runs are multi-seed only on ALFWorld. Inference uses forked vLLM and SGLang engines; in the vLLM fork each completion is padded to a multiple of 16 tokens so that the stream is not broken by partially filled cache blocks, and weight updates are applied in flight without flushing the cache.

Efficiency and task scores. On TextWorld, KV-streams reach the final performance of re-prefill compaction with 3.7 to 11.3 times fewer GPU-hours; on ALFWorld the throughput gain is 1.2 to 3.1 times. (The abstract field of the arXiv record gives 2.6 to 5 times over three strategies; the full text gives the figures above over four.) On SWE-bench Verified the sliding window with KV-streams reaches 52.7 +/- 2.1 percent, full context 51.5 +/- 1.2 and the re-prefill Markovian Thinker 53.4 +/- 0.2, and the streamed sliding window reaches its peak in about 20 hours against more than 65 for full context. The study's summary of final scores across benchmarks is that KV-streams match re-prefill compaction within error on most strategies. It also reports that with sufficient training KV-streams can close the gap to full context and even surpass it, and that checkpoints transfer to unseen text games without loss relative to the other arms.

The supervised warm start. To test whether supervised fine-tuning is a necessary precursor, the authors collect 10,000 trajectories from the base model with full context and fine-tune on them with randomly sampled eviction masks for 500 steps, then run reinforcement learning. Once the warm start's compute is counted, reinforcement learning from the base model is more compute-efficient, with no observable drop in performance; with the Markovian Thinker the warm start slows training markedly, and with the sliding window both arms peak at about the same time.

The controlled recall task. The model is told which of five fruits it has been assigned and replies with an acknowledgement that does not repeat the fruit. It then decodes ascending integers for k tokens, with k from 16 to 512. The assignment turn is evicted; the decoded sequence and its cache entries are kept. The model is then asked which fruit it was assigned and chooses from a shuffled list of the five. Sequences containing a candidate answer are rejected, and the authors check that all reported held-out trials pass this check. Two variations test transfer to actor names with no further training and recall of an object, a television, that the recall prompt does not list. Four objectives are compared. Reinforcement learning samples 128 trajectories per batch at temperature 1.0 and top-p 0.95, rewards an exact post-eviction match with no candidate in the decoded sequence, and weights each response token by a leave-one-out advantage. Rejection-sampling fine-tuning trains on the successful samples. Supervised fine-tuning uses demonstrations whose decoded sequence is "a deterministic sequence of ascending integers truncated to k tokens", with the eviction mask in place. Self-distillation adds a KL term to a full-context teacher. Reinforcement learning, rejection-sampling fine-

tuning and supervised fine-tuning start from Qwen3-4B-Instruct and train for 100 updates; self-distillation starts from the corresponding supervised checkpoint and trains for 100 more; results are means over three seeds at the best learning rate per condition.

Figure 1 re-plots the cell values of the study's Figure 8. Supervised fine-tuning recalls the fruit at 100 percent at every k except 128 (86 +/- 23), including k = 16. Reinforcement learning reaches 100 +/- 0 at k = 64, 128 and 512, but 49 +/- 44 at 16, 73 +/- 46 at 32 and 74 +/- 44 at 256. On actor names, reinforcement learning reaches 94 +/- 1 at k = 512 against 100 +/- 0 for supervised training, and both are near 36 to 37 at k = 16. On the six-target task, supervised training is at 99 to 100 throughout, while reinforcement learning rises from 37 +/- 2 at k = 16 to 100 +/- 0 at k = 512, with values between 56 and 72 and standard deviations of 35 to 48 in between. The six-target accuracies average all six objects, not only television trials. The figure's diagram reports the base model's probability of answering "TV" as 0.00 and the trained model's as 1.00; the text says the base model places "almost no probability" on it.

The authors read these results as showing that recall rises with the cache's capacity, becomes unreliable only at the smallest budgets, and is learned through reinforcement learning alone. Section 6 argues that the supervised column does not support a capacity reading, and Section 7 that the reinforcement-learning column does not by itself identify the channel.

What the "prior work" claimed. Penaloza et al. (2026) describe MEMENTO as having found that pseudo-recurrence "requires a preliminary SFT stage in their training setup". MEMENTO's full text supports a narrower description. Its supervised stages teach a block-and-summary format from a 228,000-trace annotated dataset, and its ablation on Qwen2.5-7B finds that training on that format directly from the base model underperforms ordinary reasoning fine-tuning, which its authors attribute to the format being hard to learn before reasoning ability is in place (Kontonis et al., 2026). Its claims about the cached residue are of another kind. A restart ablation, in which each summary's entries are recomputed without its block, lowers AIME 2024 accuracy from 66.1 to 50.8 percent. Probes recover an injected five-digit passcode from the entries of a summary that never attended to the block containing it, well above the 10 percent chance level. A four-layer toy model shows leakage that stays at 17 to 19 percent across training checkpoints while task accuracy climbs from 77 to 95 percent, which the authors describe as "architectural rather than learned" (Kontonis et al., 2026). MEMENTO does not claim that supervised training is needed for the residue to exist, and its own evidence says it is not. What the two papers actually disagree about is whether a supervised stage is needed before reinforcement learning in their respective pipelines, which is a question about format and starting policy, not about the cache.

4. Four Claims Under One Sentence

The author separates the claim that streamed caches act as recurrent states into four.

C1 Presence. After eviction, the retained entries carry information about the evicted span that a suitable reader could decode. This is a property of the cached tensors.

C2 Use. The policy's outputs depend on that information: intervening on it changes behaviour. This is a property of the policy's readout.

C3 Channel. The dependence runs through the retained entries rather than through visible tokens, either the bridge text or the choice of tokens the policy generated or kept. This is a property of the route.

C4 Benefit. The dependence raises the agent's return on its task, relative to an agent trained without the residue. This is a property of the training comparison.

Each is necessary for the next to matter, and none implies the next. A residue can be present and unused, used but redundant with what the text already says, and genuinely latent yet worth nothing to return. The phrase "RL alone is all that is needed for this behavior to emerge" is most naturally read as C2 for a reinforcement-learning-trained policy, with C3 assumed and C4 implied by the efficiency results it accompanies.

Table 2 sets out where information about an evicted span can survive compaction, which continuation regime preserves each place, and which test distinguishes it. The ledger is the author's classification; its entries rest on the sources cited in Sections 5 to 8.

Three of the six places survive re-prefill and are visible text; two are invisible to a text monitor; only one, the residue, is specific to streaming. The streaming study's controlled task rules out the first two rows by construction, since the decoded sequence contains no candidate name and the recall is about an arbitrary assignment, and the last two rows by design, since the assignment is random and cannot be re-observed. The rows it leaves in play are the third and the fourth.

5. Presence Is Architectural

Causal attention makes every entry a function of every earlier token it attended to. Information about earlier tokens is not merely available to later positions; it is routinely moved there. Label words in in-context demonstrations aggregate the semantic content of their examples in shallow layers and serve as anchors for the final prediction (Wang et al., 2023). Attribute extraction for factual recall moves subject information to the last position through attention (Geva et al., 2023). A small set of retrieval heads, under five percent of heads in the models examined, does most of the copying from

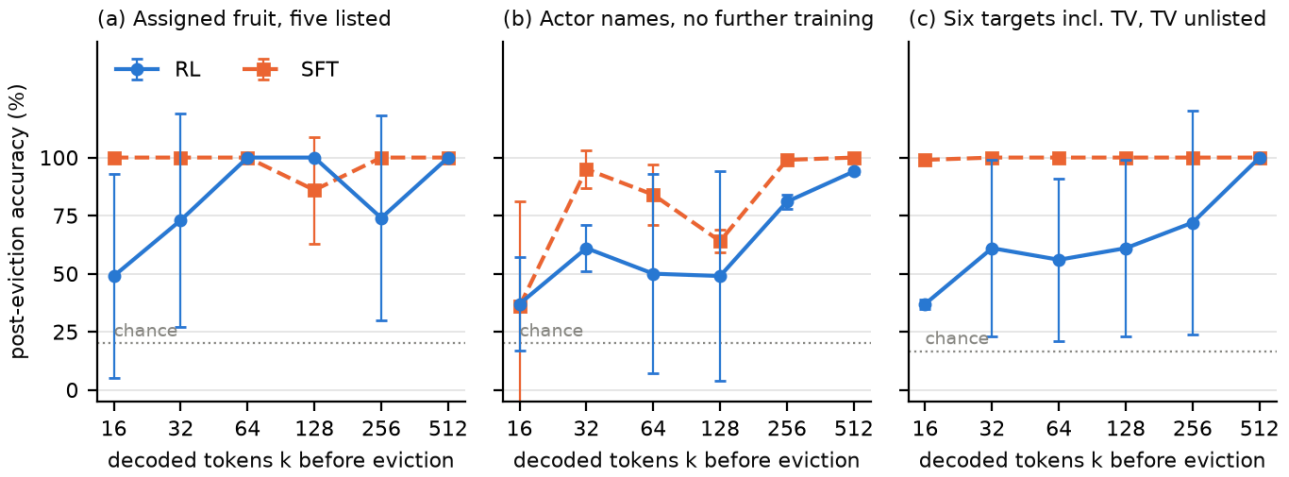

Figure 1. Post-eviction recall accuracy (percent; mean and standard deviation over three seeds, best learning rate per condition) against the number of decoded tokens kept after the assignment turn is evicted, for reinforcement learning (circles, solid) and supervised fine-tuning (squares, dashed). Values are the cell labels of Figure 8 of Penaloza et al. (2026), version 3: (a) five listed fruits, held-out assignments; (b) checkpoints trained on fruits, evaluated on actor names; (c) six trained targets including a television that the recall prompt does not list, averaged over all six. Dotted lines mark chance (20 percent; 1/6 in panel c). Error bars above 100 are an artefact of reporting a standard deviation for bimodal outcomes.

Table 1. Measured values the argument rests on, one row per published result, with the source on each row. "pts" are percentage points.

Source	Setting	Comparison	Reported value
Penaloza et al. 2026	TextWorld / ALFWorld	GPU-hour saving, streaming vs re-prefill	3.7-11.3x / 1.2-3.1x
Penaloza et al. 2026	SWE-bench Verified	streamed window / full / re-prefill Markovian	52.7 / 51.5 / 53.4%
Penaloza et al. 2026	recall, fruit, k = 16	RL vs SFT	49 + /-44 vs 100 + /-0%
Penaloza et al. 2026	recall, fruit, k = 256	RL vs SFT	74 + /-44 vs 100 + /-0%
Penaloza et al. 2026	recall, six targets, k = 512	RL; base p(TV)	100%; 0.00
Kontonis et al. 2026	Qwen3-8B, AIME 2024	residue kept vs recomputed	66.1 vs 50.8%
Kontonis et al. 2026	passcode probe, last layer	masked, 8B / 32B (chance 10)	26.5 / 22.4%
Kontonis et al. 2026	toy model, 4 layers	masked leakage across checkpoints	17-19%
Zhang et al. 2026	Qwen3-8B, book QA, context-folding agent	SFT + RL over RL alone / SFT alone	+ 8.8 / + 23.9 pts
Zhang et al. 2026	same, RL alone	compaction rate after 20 steps	near zero
Aghajohari et al. 2025	R1-Distill 1.5B	8K chunks, thinking length reached	up to 24K
Dobre et al. 2026	Qwen3-VL-8B	target success after source masked	up to ~90%
Xiao et al. 2023	StreamEval, beyond cache	recall at distance > cache size	drops to zero
Shao et al. 2025	Qwen2.5-Math-7B, MATH-500	gain, random vs true reward	21.4 vs 29.1 pts
Roger & Greenblatt 2023	paraphrase defence	bits carried by best encoding	< 3 per KB
Burger et al. 2026	medical diagnosis	unarticulated influence, necessary CoT	≥ 19% / ≥ 24%

Table 2. Ledger of the places information about an evicted span can survive compaction.

Place	Survives re-prefill	Survives streaming	Readable by a monitor	Test that isolates it
Bridge text (summary, note)	yes	yes	yes	ablate or paraphrase the bridge
Retained raw tokens	yes	yes	yes	delete them
Code in visible token choice	yes	yes	not as written	re-prefill; paraphrase
Residue in retained entries	no	yes	no	re-prefill; cache swap
Weights (cross-episode prior)	yes	yes	no	vary the assignment
Environment, re-observable	yes	yes	yes	remove re-observation

context, and pruning them breaks retrieval (Wu et al., 2024). An entry that attended to an assignment while it was visible will, on this evidence, generally encode something about it.

The direct measurements agree. In MEMENTO's probe, a summary's entries that never attended to the block with the injected passcode still yield it at 26.5 percent for Qwen3-8B and 22.4 percent for Qwen3-32B at the last layer, against 10 percent chance and 10.8 and 12.8 percent at the fourth layer; a summary written before the block yields chance, as a causal control should (Kontonis et al., 2026). (The Figure 9 caption reports the both-

layers probe, 26.7 and 23.0 percent; the last-layer values are used here.) In its toy model the signal decays with distance, from 26.2 percent one block after the target to 21.4 percent two blocks after, and is still 1.3 times chance seven blocks away. Dobre et al. (2026) show the adversarial version: soft prompts and images optimised to plant a trigger keep working after they are masked from attention, a cache-swap ablation places the persistent influence in the cache, and the attacks can be trained to survive a compaction that keeps the cached entries of a summary written by the same model.

Presence also follows from a result about exactness. Pustovit (2026) points out that, for a causal transformer, the cache computed from a text is identical to the cache a joint pass on that text plus a later query would produce, and reports zero divergences across 700 questions on two 8-billion-parameter models once chat templates are correct. The cache is the full summary of the past that later tokens see; a streamed entry is simply that summary for a past that has since been partly hidden.

Three consequences follow. First, every streaming method has a residue, whether or not its authors discuss it. Rolling-window caches (Xiao et al., 2023), heavy-hitter retention (Zhang et al., 2023), clustered selection (Li et al., 2024) and attention-based omission (Oren et al., 2024) keep entries computed with attention to tokens they later drop. Oren et al. (2024) make the recurrent reading explicit by describing the decoder-only transformer as a multi-state recurrent network whose state the eviction bounds. Second, presence is not what the streaming study's reinforcement learning produced. The residue was there at initialisation; what training could change is whether anything reads it. Third, a probe's accuracy is not evidence of use. A probe can learn the task itself, which is why control tasks are needed to make probing results selective (Hewitt and Liang, 2019), and probing accuracy can be uncorrelated with a property's importance to the model's behaviour (Elazar et al., 2020; Belinkov, 2021). MEMENTO's probe shows that the passcode can be read from the residue by a trained classifier. It does not show that the model reads it.

There is evidence consistent with an untrained model not reading the residue for exact recall. StreamingLLM keeps four sink tokens and a rolling window, so the window's entries were computed with attention to tokens since evicted. On StreamEval, accuracy holds while the distance between query and answer is within the cache, falls as the distance grows, and drops to zero once it exceeds the cache's capacity (Xiao et al., 2023). This fits a residue that was present but not used for verbatim recall, though it cannot separate non-use from a residue that has decayed too far to carry the answer; MEMENTO's toy probe shows presence itself falling with distance. The authors of StreamingLLM conclude that their method cannot extend the context length.

6. Use: What Reinforcement Learning Added, and When Reward Alone Suffices

If presence is architectural, the content of the claim "RL alone is enough" is that reinforcement learning, without a supervised stage, trained a readout. The streaming study's controlled task supports this in a specific setting, and the conditions of that setting explain both its success and its variance.

The base rate of success. The recall prompt lists the five candidates, so a policy that ignores the residue entirely still answers correctly one time in five. With 128 trajectories per batch and a leave-one-out baseline, a starting policy that answered at chance would supply about 25 rewarded trajectories in every batch from the first update, and one that already read some of the residue would supply more. Penaloza et al. (2026) note the point themselves: random sampling produces successes from the start, and the model can bootstrap from them. This is the most favourable possible regime for outcome-rewarded learning. The broader literature on reinforcement learning for language models finds that what reward can teach is bounded by what the starting policy already samples with useful frequency. Yue et al. (2025) report that, across model families and algorithms, reinforcement-learned models win at pass@1 but lose to their base models at large k, and that the observed abilities originate in the base model; Wu et al. (2025b) find that, at their sampling budgets, more questions lose a verified success under reinforcement learning than gain one. The other side exists: Liu et al. (2025) report that prolonged training with resets and diverse tasks reaches problems the base model fails at any sampling budget. On either side of that dispute, a task whose chance success is 20 percent sits where reward alone is expected to work.

What the TV variant adds. The six-target task is the study's strongest evidence that reinforcement learning created a readout rather than sharpened one. The recall list omits the television, the figure reports the base model's probability of answering it as 0.00, and at $k = 512$ reinforcement learning reaches 100 percent averaged over all six targets. Two qualifications apply. First, the accuracy at smaller k averages fruit trials, where chance is 20 percent among the listed options, with television trials, so the television-specific rate below $k = 512$ is not reported. Second, a policy-gradient update on fruit trials changes shared parameters, so the probability of an unsampled answer can rise without that answer ever being rewarded; a near-zero starting probability is not a proof that the behaviour was outside the policy's reach. The author reads the variant as showing that reinforcement learning can move probability onto an answer the prompt does not offer, given a residue that encodes it and many successful neighbouring trials.

Capacity or optimisation. The authors interpret k as the cache's capacity and the low-k failures as capacity limits. Their supervised column contradicts that reading. With the same number of decoded tokens, 16, supervised fine-tuning recalls the fruit at 100 +/- 0 percent and, on the six-target task, at 99 +/- 0 percent. Sixteen entries are therefore enough to carry the assignment, given a readout trained to use them. The reinforcement-learning arm's 49 +/- 44 at k = 16 is a failure of optimisation, not of capacity. The same column is not monotone in k: 100 +/- 0 at 128, then 74 +/- 44 at 256, then 100 +/- 0 at 512. A standard deviation of 44 to 46 points over three seeds, on a scale bounded by 0 and 100, means that the seeds disagree by tens of points; it is the signature of runs that either find the readout or do not, rather than of a shared partial result. The reinforcement-learning result is better described as a high-variance search that sometimes finds a readout and sometimes does not, with the probability of success not cleanly ordered by k. The study's text says recall becomes unreliable only at 16 and 32 tokens; its figure shows the same unreliability at 256.

Forced and optional compaction. The most direct contrast for the general question comes from the second paper called MEMENTO. Zhang et al. (2026) train a Qwen3-8B agent to answer questions about books of up to 2.5 million tokens while holding at most 20,000 in context, and raise its accuracy from 36.9 to 67.8 percent. In their ablation of the context-folding agent, combining a supervised stage with reinforcement learning improves accuracy by 23.9 points over the supervised stage alone and 8.8 points over reinforcement learning alone. The two policies start reinforcement learning at nearly the same validation accuracy, so the supervised stage does not help by itself. What it changes is behaviour: the supervised policy compacts more than four times as often, and without it reinforcement learning drives the compaction rate to nearly zero within about 20 steps, where it stays for the rest of training. In Zhang et al.'s setting a restart into a fresh window is forced when the agent reaches the context budget, but the agent can also request one earlier, and writing notes and running past a single window are behaviours it can avoid; what collapses is the number of "context-window restarts per RL training rollout". A policy that has not been shown how to compact well finds that avoiding restarts pays better. In the streaming study's setting, compaction is imposed by the environment every 10 or 30 turns, and the policy cannot decline it. The Markovian Thinker's analysis makes the same point from the other direction: its environment forces resets, and off-the-shelf reasoning models from 1.5 to 120 billion parameters often produce traces that survive the reset without training, supplying the positive samples that make reinforcement learning work (Aghajohari et al., 2025).

The general statement that both results appear to fit, with the qualification that Zhang et al.'s restarts are themselves forced at the budget, is that reward alone suffices for a context-management behaviour when the environment forces the behaviour's occasion on a schedule the policy cannot avoid and the starting policy already succeeds on it at a usable rate, and may not suffice when the policy can avoid the occasion and its early attempts are worse than avoiding it. This is consistent with the finding that models differ in self-improvement under identical reinforcement learning according to whether their starting behaviour includes the needed patterns, and that priming with those patterns, even in incorrect solutions, closes the gap (Gandhi et al., 2025). It is also consistent with older statements that supervised training stabilises output format so that later reinforcement learning can work (Chu et al., 2025), with the report that pure reinforcement learning on a large base model can incentivise reflection, verification and other advanced reasoning patterns without human-labelled trajectories (DeepSeek-AI et al., 2025), and with InftyThink+ and MEMENTO using supervised cold starts before reinforcement learning (Yan et al., 2026; Kontonis et al., 2026). The streaming study's warm-start experiment tests a different supervised stage from either MEMENTO's: it fine-tunes on the base model's own full-context trajectories under random eviction masks, which adapts the model to masking without teaching any compaction behaviour or any recall. Its null result says that this particular warm start does not pay for its compute on TextWorld. It does not address whether a supervised stage that teaches a behaviour would.

Which base model. Every reinforcement-learning result in the streaming study uses a Qwen model. Shao et al. (2025) show that GRPO on Qwen2.5-Math-7B gains 21.4 points on MATH-500 from random rewards against 29.1 from true ones, attribute the effect to a clipping bias that amplifies behaviours already favoured by pretraining, and find that the same spurious rewards often fail for Llama and OLMo models. RL post-training tends to converge on a dominant output distribution that amplifies pretraining patterns (Zhao et al., 2025). Nothing in these results says the streaming recall result is spurious; its reward is exact and its chance level is known. They do say that "reinforcement learning alone suffices" is a claim about a model family until it is shown on another.

7. Channel: The Visible-Token Alternative

The controlled task is designed so that, after eviction, the only route from the assignment to the answer is through the retained entries. The design is sound for the supervised arm, and it has a gap for the reinforcement-learning arm.

In supervised fine-tuning the decoded sequence is fixed in the training data: every demonstration contains the same ascending integers, truncated to k tokens, whatever the assignment. The visible text in those demonstrations

is identical across the five fruits and so carries no information about which one was assigned, and a readout trained only on such text can have learned to recall only from the residue. At held-out evaluation the model decodes its own sequence, and the study does not report that this sequence is the same for every assignment; but a policy never trained on an informative sequence has no learned way to read one. A supervised policy that recalls at 100 percent from 16 tokens is therefore reading the residue. The streaming study's supervised column is therefore direct evidence for C2 and C3 together, and it is the cleanest such evidence the author has located for agent-style eviction.

In reinforcement learning the policy samples its own sequence at temperature 1.0 with top-p 0.95, while the assignment is still visible. The reward requires an exact recall and excludes sequences that contain a candidate name. Nothing in the reward, as described, requires the sequence to be the same across assignments, or to start at 1, or to be consecutive, or to use one space between numbers. Five assignments need just over two bits. A starting integer, a skipped number, a doubled space or the point at which the count is cut off would carry them. A policy rewarded for recall after eviction, and able to choose its decoded tokens while the assignment is visible, has two routes to the reward: read the residue, or write the assignment into the tokens and read the tokens. Both survive streaming. Only the second survives re-prefill.

The literature on encoded reasoning establishes that the second route is learnable and that reward pressure finds it. Language models can be trained to encode intermediate reasoning in generated text that a human reader does not understand (Roger and Greenblatt, 2023). Steganographic collusion between agents can arise from misspecified rewards during training, and passive oversight and paraphrasing do not fully prevent it (Mathew et al., 2024). Penalising specific strings in load-bearing reasoning makes models substitute other strings while keeping the same method, and when the penalised strings belong to a class the model develops an encoding that generalises to held-out members (Skaf et al., 2025). Frontier models evaluated without training rarely hide a message from a monitor under standard affordances but succeed with a scratchpad and a coordinated scheme (Zolkowski et al., 2025), and capability evaluations for secret collusion show a jump at GPT-4 (Motwani et al., 2024). The recall task asks a policy to communicate with itself across an eviction, about a five-way choice, with no monitor and with a reward that checks only for the literal candidate names. It is close to the easiest steganographic task one could write.

The task's transfer results constrain the alternative without excluding it. A code learned for five fruits would not obviously transfer to five actor names, so the reinforcement-learning arm's 94 percent on actors at $k = 512$ counts against a fruit-specific code. A code keyed to something that does transfer, such as the position of the assigned item in a candidate list shown at assignment time, would survive the change of names. Whether the assignment list keeps a fixed order is not stated; the recall list is shuffled. The study also evaluates actor recall without the names listed, a condition that would put pressure on any position-keyed code. Its values, given as cell labels in the study's Figure 15B, show both arms collapsing: reinforcement learning reaches 27.2 percent at $k = 512$ and 0.0 percent at $k = 16$, and supervised fine-tuning, whose route is the residue by construction, reaches 31.7 percent at $k = 512$ and its highest value, 63.2 percent, at $k = 32$. Because the arm known to read the residue fails as well, the condition does not separate a position-keyed code from a residue read. The author therefore treats the channel of the reinforcement-learning arm as undetermined.

The undetermined channel matters for the claim because it changes what "RL alone suffices" would mean. If the reinforcement-learning policy reads the residue, reinforcement learning trained a latent readout without supervision, which is the claim. If it reads a code in its own tokens, reinforcement learning trained a covert textual channel, which would work equally well under re-prefill and says nothing about the cache. The two can also mix, with some seeds finding one route and some the other, which is one way to produce the bimodal outcomes in Figure 1.

Three controls would decide the question, and each is cheap given the study's own infrastructure. The first is a re-prefill evaluation: take a trained reinforcement-learning checkpoint, evict the assignment, discard the cache, encode the retained sequence afresh, and ask for the recall. Accuracy at chance indicates the residue; accuracy near the streamed value indicates a textual code. Because a policy trained on residues may also fail under re-prefill simply because recomputed entries are unfamiliar, the control is most informative when paired with the second. The second is a sequence swap: force the reinforcement-learning policy to decode the canonical sequence used in the supervised demonstrations while the assignment is visible, keep the resulting entries, evict and ask. High accuracy indicates that the policy can read the residue without relying on its own token choices. The third is a cache swap of the kind Dobre et al. (2026) used: transplant the retained entries from a trial with one assignment into a trial with another while keeping the text, and observe which assignment the answer follows. KV-cache patching between models has already been used to show that some fine-tuned behaviours are carried in the cache rather than in the answer tokens, concentrated at a few positions (Loos and Eberle, 2026). A fourth, purely observational, check is to train a classifier from the decoded text alone to the assignment; any accuracy above chance shows that the text carries information.

8. Benefit: Matched and Mismatched Comparisons

The evidence that a cached residue helps a model come from two kinds of comparison, and they point in different directions.

Mismatched. MEMENTO's restart ablation evaluates one checkpoint, trained with summary entries that kept their residue, in two modes: as trained, and with the entries recomputed without the block. AIME 2024 accuracy falls from 66.1 to 50.8 percent (Kontonis et al., 2026). The ablation's table notes that the first figure comes from a 64-repetition evaluation and the second from 8 repetitions. More importantly, the ablation changes the inputs to a readout that was trained on residues. A drop could reflect information lost, or a readout meeting entries unlike any it saw in training, or both. The author does not doubt that MEMENTO's model uses its residue; the probe and the drop together make that likely. What the ablation measures is the cost of removing the residue from a policy that learned to depend on it, which is an upper bound on its value and not an estimate of what training with the residue gained.

Matched. On TextWorld and ALFWorld the streaming study trains each compaction strategy twice, once under re-prefill and once under streaming, and evaluates each in the regime it was trained in. On software engineering only the Markovian Thinker is trained in both regimes; Summary and Sliding-Window are trained with streaming alone. That is the comparison that answers C4. The authors read it favourably: on TextWorld streaming reaches re-prefill's final performance "while performing better or similarly", and overall it "matches or exceeds the performance of re-prefill compaction" (Penaloza et al., 2026). The caption of their Figure 1 is more guarded, saying it matches re-prefill "within error on most strategies". The exceptions visible in that figure run in both directions. On TextWorld the summary strategy scores about 0.89 streamed against about 0.71 re-prefilled, a gap of roughly 18 points with error bars that do not overlap, from single-seed runs. On ALFWorld the streamed sliding window scores about 0.93 against about 0.81 re-prefilled, a gap of about 12 points. On SWE-bench Verified the matched Markovian Thinker goes the other way, about 49 percent streamed against 53.4 +/- 0.2 re-prefilled. If the residue carried useful information that the text did not, the streamed arm of each strategy should do better on tasks that, as the study itself argues, often require recalling details received early in an episode; the figure shows gains for two strategies, a loss for one and no separable difference for the rest. The summary strategy is the closest analogue of MEMENTO's restart ablation, because under streaming its summary's entries keep what they computed from the deleted turns and under re-prefill they do not, so its TextWorld gap is the result most favourable to C4. The software-engineering comparison the study reports in its text sets a streamed sliding window against a re-prefill Markovian Thinker, which crosses strategies as well as regimes; the matched pair in its figure favours re-prefill.

The matched comparison also varies more than the residue. Streaming keeps original positions, while re-prefill resets them; streaming spends less compute per step, so at a fixed wall-clock budget it takes more steps; and with in-flight weight updates that do not flush the cache, a streamed entry may have been computed under an earlier version of the policy than the one that reads it. The step difference is not hypothetical. The study stops a run early "only when its wall-clock time exceeds that of the full-context run", and reports that "Summary and Sliding-Window with re-prefill compaction exceed this budget and have longer step times" (Penaloza et al., 2026), so the re-prefill arms behind the two largest gaps in favour of streaming may have trained for fewer steps. Differences in both directions, from single seeds on TextWorld, with re-prefill arms cut short and positions differing, therefore cannot be credited to the residue: a streamed gain is consistent with a re-prefill arm that stopped earlier, and the software-engineering loss with a residue whose value is offset by another difference. They are not evidence that the residue helps. The single clean statement the agentic results support is that training with a streamed residue costs little or no final score at the scales tested, the largest shortfall in the figure being about four points for the Markovian Thinker on SWE-bench Verified, which is what an efficiency method needs to show.

A two-by-two design would isolate the residue from positions. Re-prefill the retained text with its original position indices rather than reset ones: this removes the residue and keeps the positions. Stream with positions reassigned within the cache, as StreamingLLM does (Xiao et al., 2023): this keeps the residue and resets the positions. Training all four arms at a matched number of steps, and evaluating each both in its own regime and in the others, would separate the residue's contribution to return from the position effect and from the familiarity effect of a mismatched evaluation.

The benefit question has an ancestor in the reasoning literature. Pause tokens help when a model is both pretrained and fine-tuned with them (Goyal et al., 2023), and filler tokens can replace a chain of thought on certain algorithmic tasks, but learning to use them is difficult and needs specific, dense supervision (Pfau et al., 2024). Hidden computation through extra positions is possible and not automatic. Lanham et al. (2023) found that the performance boost from chain of thought did not appear to come from added test-time compute alone or from information encoded in particular phrasings. The prior expectation from this literature is that a latent channel will be used when training makes it the easiest route to reward, and that agents trained on ordinary tasks, where the text already carries what is needed, will lean on it little.

9. Consequences

Monitoring. A re-prefilled agent's state after compaction is its visible text, so a monitor that reads the summary reads everything the agent carries forward. A streamed agent's state is not. Chain-of-thought monitoring is valued precisely because reasoning in language exposes intent (Korbak et al., 2025), it can be far more effective than watching actions alone (Baker et al., 2025), and models evade it far less easily when the reasoning is necessary for the behaviour (Emmons et al., 2025). The same literature documents its fragility: explanations omit the features that drove an answer (Turpin et al., 2023), reasoning models reveal hint use in their chains of thought often less than 20 percent of the time (Chen et al., 2025), influence can go unarticulated even when the chain of thought is necessary, in at least 19 and 24 percent of correct answers for two attributes on a medical task (Burger et al., 2026), and strong optimisation against a monitor produces obfuscated reward hacking (Baker et al., 2025). A residue adds a route for information to persist through compaction without appearing in any text, and Coconut-style continuous reasoning (Hao et al., 2024) shows that models can be trained to reason through hidden states directly. Under streaming, the summary is a partial record, and how partial depends on C2 and C3. A monitor that audits compaction summaries should be told which regime the agent ran in.

Deletion as a defence. Several prompt-injection defences work by detecting injected text and removing it before the agent acts; PromptArmor reports false-positive and false-negative rates below one percent on AgentDojo and attack success below one percent after removal (Shi et al., 2025). In a re-prefill pipeline, removal removes. In a streaming pipeline, an injection removed after it has been read, as in a sanitiser applied to history or a compaction that drops a poisoned tool output, leaves its residue in every retained entry computed while it was visible. Dobre et al. (2026) show that adversarial inputs can be optimised for exactly this persistence, including through a compaction that keeps a summary's entries. The same consideration applies to any request to forget something within a session. Deleting a span from a streamed context is not the same as never having read it.

Portability. A policy trained with streaming and served by a stack that rebuilds context, or by a provider whose prefix cache does not survive a model switch or a long idle period, will meet recomputed entries. If the policy came to depend on its residue, its behaviour will change in ways that no offline evaluation in the training regime reveals. MEMENTO's 15.3-point drop is an estimate of how large such a change can be for a policy trained to rely on the residue (Kontonis et al., 2026). The reverse transfer, a re-prefill-trained policy served with streaming, changes nothing for the policy's text but exposes it to a residue it never learned to use or ignore. Practitioners shipping compaction-trained agents should evaluate them under the continuation regime of the serving stack, not only of the trainer.

Comparing compaction methods. Strategies with the same name differ between regimes. A Markovian Thinker under streaming no longer conditions only on its visible carryover: the retained half of its context holds entries computed with the deleted half in view. A summary strategy under streaming is closer to MEMENTO than to ReSum or SUPO. Comparisons across papers that do not state the continuation regime compare different objects. The rate at which summaries preserve particular kinds of information is already known to depend on small prompt choices, with one gist prompt preserving 3.05 percent of temporal expressions and a one-sentence change raising that to 62.39 percent (Kyrkewood, 2026), and the volume and content of summaries vary from run to run (Cim et al., 2026). Under streaming, part of what a summary fails to keep may be carried anyway, which makes summary quality, as scored on the text, a weaker predictor of task success than it is under re-prefill.

Efficiency stands. None of this weakens the streaming study's efficiency results. Removing repeated prefill is a real saving, and the saving does not depend on the residue being useful. Sliding-window caches with sinks are competitive with retrofitted linear attention at no training cost (Jolicoeur-Martineau et al., 2026), prefix-plus-window eviction makes reasoning models about three times faster without training (Muennighoff et al., 2026), and training-free soft tokens can carry absorbed context forward in hybrid recurrent models (He et al., 2026). The point of separating the four claims is that efficiency, recurrence and auditability are different properties, and a method can have the first without having been shown to have the second, while losing some of the third.

10. An Attribution Procedure and What Four Accounts Predict

The author proposes the following procedure for reading a claim that a streamed cache carries a recurrent state. It classifies the evidence by which of C1 to C4 it can support, and states the control that would upgrade it.

Applied to the published record, the procedure gives the following. MEMENTO's probe supports C1 with a causal control but without a control task (step 1). Its restart ablation is step 4 on a residue-trained policy, without step 5 or 6, and so supports C2 and C3 for its model, with the familiarity confound noted in Section 8. The streaming study's supervised recall arm satisfies step 3 by construction in its training data, since the demonstrated text is constant, and supports C2 and C3. Its reinforcement-learning arm passes step 2 only in the sense that chance success is 20 percent, which places it in the elicitation regime; steps 3 to 6 are not reported. Its agentic results are step 7 without the position factor and support "little or no cost to final score" rather than C4.

Algorithm 1. Attributing a post-compaction recall or return gain to a streamed residue

```
Input: policy pi, compaction operator kappa, task T, claimed effect Y (recall or return)
1. Presence (C1): probe retained entries for evicted content with a control task
   (Hewitt-Liang selectivity) and a pre-target causal control. Record accuracy.
2. Untrained readout: evaluate the initial policy with streaming and with re-prefill.
   If chance success on T is not ~0, report it; RL from a positive base rate is elicitation.
3. Text leakage: train a classifier from retained visible tokens to the evicted content.
   If above chance, the text carries a code; C3 cannot be credited from streaming alone.
4. Re-prefill evaluation: discard the cache after eviction, re-encode retained text,
   re-measure Y. Drop to chance -> residue route; no drop -> textual route.
5. Sequence swap: teacher-force a canonical retained sequence while the evicted span
   is visible; keep entries; evict; re-measure Y. High Y -> pi reads the residue.
6. Cache swap: transplant retained entries between trials with different evicted
   content, text held fixed. Y follows the transplanted entries -> C2 and C3 hold.
7. Benefit (C4): train matched arms {residue kept, recomputed} x {positions original,
   reset} at equal steps; evaluate each in all regimes. Credit C4 only to the
   matched residue contrast, not to a mismatched restart.
8. Seeds and family: report per-seed outcomes and at least one non-Qwen model.
Output: the subset of {C1, C2, C3, C4} the evidence supports, and the missing step.
```

Dobre et al. (2026) perform step 6 for an adversarial input and establish C2 and C3 for that case.

Table 3 states what four accounts of the reinforcement-learning recall result predict under the controls of Algorithm 1. It contains predictions, not measured values.

A1 is the streaming study's reading. A2 is the steganographic alternative of Section 7. A3 holds that different seeds found different routes, which would fit the bimodal standard deviations. A4 holds that the readout already existed at a low level in the base model and reinforcement learning amplified it, which on the evidence of Section 6 is likely to be part of any account; it differs from A1 mainly in its prediction for other model families. The accounts are not exclusive, and the controls are designed so that each changes at least one prediction.

11. The Strongest Case Against This Paper

Four objections deserve a direct answer.

"**The supervised arm already proves the point.**" The streaming study's supervised policy, trained on constant visible text, recalls from the residue, so the residue can be read, and the reinforcement-learning arm reaches the same accuracy at the largest budget. Why insist that the reinforcement-learning arm prove its route separately? Because the claim is about what reinforcement learning, without supervision, produces. That a readout of the residue exists in the space of policies, which the supervised arm shows, does not show that reward finds that readout rather than another route to the same reward. Reinforcement learning optimises reward, not route, and the literature in Section 7 shows that it finds textual routes when they are open. If the reinforcement-learning arm turns out to read the residue, the claim stands as written; the request is for the control that would show it.

"**A textual code is implausible in 100 updates.**" The reinforcement-learning runs are short, the instruction says to count, and a 4-billion-parameter model may not invent a code at all in that time. That may be so. Three facts keep the alternative open. The reward needs only about two bits; the policy samples at temperature 1.0, so its sequences already vary from trial to trial and some of that variation may correlate with the assignment from the first update; and a correlation of that kind, once rewarded, is exactly what a policy gradient amplifies. Skaf et al. (2025) report that substitution codes emerge when specific strings are penalised in load-bearing reasoning. Whether a code emerged here is an empirical question that a classifier on the decoded text would answer in minutes.

"**Presence, use and channel are distinctions without a difference for an efficiency paper.**" For the efficiency result they are, and Section 9 says so. They make a difference once the recurrent reading is carried into design. A practitioner told that streaming gives agents a recurrent state for free may drop summary quality checks, ship a streamed policy to a re-prefill server, or rely on deletion to remove an injected instruction. Each of those decisions depends on which of the four claims holds.

"**The MEMENTO restart drop shows benefit.**" It shows that removing the residue from a policy trained on it costs 15.3 points on one benchmark (Kontonis et al., 2026). That is evidence that MEMENTO's model uses its residue, and the author accepts it as such. It does not show that training with the residue produced a better policy than training without it, because the comparison it makes is between two evaluation modes of one policy. The streaming study's matched comparison is the one that addresses benefit, and its differences run in both directions across strategies, with confounds that prevent crediting the residue.

The author also notes the limits of this analysis. It rests on one controlled task, whose values come from a published figure that gives means and standard deviations over three seeds rather than per-seed results, so the bimodality inferred in Section 6 is an inference from standard deviations near 44 and 48 on a scale from

Table 3. Predictions of four accounts of the reinforcement-learning recall result under Algorithm 1's controls.

Control	A1 residue read	A2 textual code	A3 mixed by seed	A4 prior sharpened
Text classifier (step 3)	chance	above chance	above in some seeds	chance
Re-prefill eval (step 4)	near chance	near streamed	splits by seed	near chance
Sequence swap (step 5)	high	low	splits by seed	high
Cache swap (step 6)	follows cache	follows text	splits by seed	follows cache
Same recipe, non-Qwen model	similar	similar	similar	lower

0 to 100, not an observation. The reading of MEMENTO as making no claim that supervision creates the residue is a reading of its text; its authors may have intended a broader claim about the pipeline than their analysis section states.

12. What Is Not Known

It is not known which route the streaming study's reinforcement-learning policy uses to recall an evicted assignment. The supervised policy reads the residue; the reinforcement-learning policy may read the residue, a code in its own tokens, or both.

It is not known whether any agent trained with streaming on an ordinary task, such as TextWorld, ALFWorld or software engineering, uses its residue at all. No re-prefill evaluation, sequence swap or cache swap of an agentic checkpoint has been published. The matched comparisons differ in both directions across strategies, and their gains for streaming come with confounds, single seeds on TextWorld and re-prefill arms that exceeded the wall-clock budget; they are equally consistent with an unused residue and with a used residue whose value is offset, or overstated, by position or step-count effects.

It is not known how much of MEMENTO's 15.3-point restart drop is lost information and how much is a residue-trained readout meeting unfamiliar entries. A model trained under restart and evaluated under restart would supply the missing baseline.

It is not known whether the elicitation reading of "reinforcement learning alone suffices" holds when chance success is near zero, outside the Qwen family, or for content richer than a five-way choice. The six-target variant is the only evidence on the first point, and it reports accuracy averaged over targets.

It is not known how the effect of keeping original positions in a streamed cache compares with the effect of the residue. Streaming and re-prefill differ in both, and no published comparison varies them separately.

What is known is that the residue is present without training, that a supervised readout can use it from 16 retained tokens, that reinforcement learning with a 20 percent chance rate can learn a readout of some kind with high seed-to-seed variance, that a supervised stage was needed for reinforcement learning to sustain compaction in a long-document agent able to avoid it, and that adversarial content can be optimised to persist in a residue after the content itself is masked.

13. Conclusion

KV-streams make compaction cheap to train, and that result stands on its own. The accompanying claim, that reinforcement learning alone suffices for a streamed cache to act as a recurrent state, is better stated as four claims. Presence of a residue is a property of causal attention, documented by the very prior work the claim is set against, and it needs no training. Use is what training adds; the controlled evidence shows supervised training adding it reliably from very few retained tokens and reinforcement learning adding something, with high variance, in a task where one answer in five is right by chance. Channel is established for the supervised arm and open for the reinforcement-learning arm, whose own token choices could carry the answer. Benefit to an agent has not been shown in a comparison that holds training fixed.

The disagreement with the earlier method dissolves on inspection. MEMENTO's supervised stages teach a summary format; its own analysis calls the residue architectural. The question of when reward alone can find a context-management behaviour has a better answer in the contrast between forced and optional compaction: where the environment imposes the occasion and the starting policy already succeeds on it, reward suffices; where the behaviour is optional and its first attempts cost more than they return, reinforcement learning drops it.

A residue is not a recurrence until something is shown to read it, by the route claimed, to the agent's benefit. The controls that would show this are a re-prefill evaluation, a sequence swap and a cache swap on the trained checkpoints that already exist, together with a matched two-by-two training comparison of residue and positions. Until they are reported, a summary written by a streamed agent should be treated as an incomplete record of what that agent carries, and a deletion from its context should not be treated as a deletion from its state.

References

Aghajohari, M., Chitsaz, K., Kazemnejad, A., Chandar, S., Sordoni, A., Courville, A., & Reddy, S. (2025). The Markovian Thinker: Architecture-Agnostic Linear Scaling of Reasoning. arXiv:2510.06557. doi:10.48550/arXiv.2510.06557.

Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., Madry, A., Zaremba, W., Pachocki, J., et al. (2025). Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation.

arXiv:2503.11926. doi:10.48550/arXiv.2503.11926.

Belinkov, Y. (2021). Probing Classifiers: Promises, Shortcomings, and Advances. arXiv:2102.12452. doi:10.48550/arXiv.2102.12452.

Bulatov, A., Kuratov, Y., & Burtsev, M. S. (2022). Recurrent Memory Transformer. arXiv:2207.06881. doi:10.48550/arXiv.2207.06881.

Burger, L., Komorowski, P., Weckbecker, M., Kortukov, E., Jain, A., Lapuschkin, S., & Samek, W. (2026). Chain of Thought Monitoring Can Fail Even When CoT Is Necessary. arXiv:2610.18105. doi:10.48550/arXiv.2610.18105.

Chen, S., Wong, S., Chen, L., & Tian, Y. (2023). Extending Context Window of Large Language Models via Positional Interpolation. arXiv:2306.15595. doi:10.48550/arXiv.2306.15595.

Chen, Y., Benton, J., Radhakrishnan, A., Uesato, J., Denison, C., Schulman, J., Somani, A., Hase, P., et al. (2025). Reasoning Models Don't Always Say What They Think. arXiv:2505.05410. doi:10.48550/arXiv.2505.05410.

Chevalier, A., Wettig, A., Ajith, A., & Chen, D. (2023). Adapting Language Models to Compress Contexts. arXiv:2305.14788. doi:10.48550/arXiv.2305.14788.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., et al. (2025). SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. arXiv:2501.17161. doi:10.48550/arXiv.2501.17161.

Cim, M., Topcu, B., Das, C., & Kandemir, M. T. (2026). Parallel Context Compaction for Long-Horizon LLM Agent Serving. arXiv:2605.23296. doi:10.48550/arXiv.2605.23296.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. arXiv:1901.02860. doi:10.48550/arXiv.1901.02860.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., et al. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948. doi:10.48550/arXiv.2501.12948.

Deng, C., Zhang, Z., Mao, K., Li, S., Huang, X., Yu, D., & Dou, Z. (2024). A Silver Bullet or a Compromise for Full Attention? A Comprehensive Study of Gist Token-based Context Compression. arXiv:2412.17483. doi:10.48550/arXiv.2412.17483.

Dobre, D., Schwinn, L., Gidel, G., Gella, S., Taslakian, P., & Noel, P. (2026). Visual Memory Attacks Can Persist Through The KV Cache. arXiv:2610.09027. doi:10.48550/arXiv.2610.09027.

Elazar, Y., Ravfogel, S., Jacovi, A., & Goldberg, Y. (2020). Amnesic Probing: Behavioral Explanation with Amnesic Counterfactuals. arXiv:2006.00995. doi:10.48550/arXiv.2006.00995.

Emmons, S., Jenner, E., Elson, D. K., Saurous, R. A., Rajamanoharan, S., Chen, H., Shafkat, I., & Shah, R. (2025). When Chain of Thought is Necessary, Language Models Struggle to Evade Monitors. arXiv:2507.05246. doi:10.48550/arXiv.2507.05246.

Gandhi, K., Chakravarthy, A., Singh, A., Lile, N., & Goodman, N. D. (2025). Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. arXiv:2503.01307. doi:10.48550/arXiv.2503.01307.

Ge, T., Hu, J., Wang, L., Wang, X., Chen, S., & Wei, F. (2023). In-context Autoencoder for Context Compression in a Large Language Model. arXiv:2307.06945. doi:10.48550/arXiv.2307.06945.

Geva, M., Bastings, J., Filippova, K., & Globerson, A. (2023). Dissecting Recall of Factual Associations in Auto-Regressive Language Models. arXiv:2304.14767. doi:10.48550/arXiv.2304.14767.

Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., & Nagarajan, V. (2023). Think before you speak: Training Language Models With Pause Tokens. arXiv:2310.02226. doi:10.48550/arXiv.2310.02226.

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., & Tian, Y. (2024). Training Large Language Models to Reason in a Continuous Latent Space. arXiv:2412.06769. doi:10.48550/arXiv.2412.06769.

He, X., Fan, K., Zhang, R., Wang, B., Zhang, Y., Jiang, X., & Dong, Y. (2026). In-Model Context Compression with Agentic Soft Tokens for Long-Horizon LLM Agents. arXiv:2610.41291. doi:10.48550/arXiv.2610.41291.

Hewitt, J., & Liang, P. (2019). Designing and Interpreting Probes with Control Tasks. arXiv:1909.03368. doi:10.48550/arXiv.1909.03368.

Jolicoeur-Martineau, A., Sukthanker, R. S., Cameron, P., & Gervais, E. (2026). Sliding-window beats linear attention. arXiv:2608.28444. doi:10.48550/arXiv.2608.28444.

Kang, M., Chen, W., Han, D., Inan, H. A., Wutschitz, L., Chen, Y., Sim, R., & Rajmohan, S. (2025). ACON: Optimizing Context Compression for Long-horizon LLM Agents. arXiv:2510.00615. doi:10.48550/arXiv.2510.00615.

Kontonis, V., Zeng, Y., Garg, S., Chen, L., Tang, H., Wang, Z., Awadallah, A., Horvitz, E., et al. (2026). MEMENTO: Teaching LLMs to Manage Their Own Context. arXiv:2604.09852. doi:10.48550/arXiv.2604.09852.

Korbak, T., Balesni, M., Barnes, E., Bengio, Y., Benton, J., Bloom, J., Chen, M., Cooney, A., et al. (2025). Chain of Thought Monitorability: A New and Fragile Opportunity for AI Safety. arXiv:2507.11473. doi:10.48550/arXiv.2507.11473.

Kyrkewood, N. E. (2026). The Sleeping Agent: What Gist-Based Context Compression Loses and Why. arXiv:2608.11775. doi:10.48550/arXiv.2608.11775.

Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., et al. (2023). Measuring Faithfulness in Chain-of-Thought Reasoning. arXiv:2307.13702. doi:10.48550/arXiv.2307.13702.

Li, T., Zhang, J., Jurayj, W., Wang, X., Jin, C., Farajtabar, M., Nalisnick, E., & Khashabi, D. (2026a). Self-Compacting Language Model Agents. arXiv:2606.23525. doi:10.48550/arXiv.2606.23525.

Li, Y., Hou, Z., Jing, Y., Tang, J., & Dong, Y. (2026b). CompactionRL: Reinforcement Learning with Context Compaction for Long-Horizon Agents. arXiv:2607.05378. doi:10.48550/arXiv.2607.05378.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., et al. (2024). SnapKV: LLM Knows What You are Looking for Before Generation. arXiv:2404.14469. doi:10.48550/arXiv.2404.14469.

Liu, M., Diao, S., Lu, X., Hu, J., Dong, X., Choi, Y., Kautz, J., & Dong, Y. (2025). ProRL: Prolonged Reinforcement Learning Expands Reasoning Boundaries in Large Language Models. arXiv:2505.24864. doi:10.48550/arXiv.2505.24864.

Loos, J., & Eberle, O. (2026). KV-Cache Patching: Tracing How and Where Fine-Tuned Behaviors Arise. arXiv:2610.41480. doi:10.48550/arXiv.2610.41480.

Lu, M., Sun, W., Du, W., Ling, Z., Yao, X., Liu, K., & Chen, J. (2025). Scaling LLM Multi-turn RL with End-to-end Summarization-based Context Management. arXiv:2510.06727. doi:10.48550/arXiv.2510.06727.

Mathew, Y., Matthews, O., McCarthy, R., Velja, J., Schroeder de Witt, C., Cope, D., & Schoots, N. (2024). Hidden in Plain Text: Emergence & Mitigation of Steganographic Collusion in LLMs. arXiv:2410.03768. doi:10.48550/arXiv.2410.03768.

Motwani, S. R., Baranchuk, M., Strohmeier, M., Bolina, V., Torr, P. H. S., Hammond, L., & Schroeder de Witt, C. (2024). Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. arXiv:2402.07510. doi:10.48550/arXiv.2402.07510.

Mu, J., Li, X. L., & Goodman, N. (2023). Learning to Compress Prompts with Gist Tokens. arXiv:2304.08467. doi:10.48550/arXiv.2304.08467.

Muennighoff, N., Wang, Z., Chen, Z., Shi, W., Hui, B., Yang, J., Jiang, D., Senghaas, M., et al. (2026). Prefix Sliding for efficient test-time scaling. arXiv:2608.26070. doi:10.48550/

arXiv.2608.26070.

Oren, M., Hassid, M., Yarden, N., Adi, Y., & Schwartz, R. (2024). Transformers are Multi-State RNNs. arXiv:2401.06104. doi:10.48550/arXiv.2401.06104.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., & Gonzalez, J. E. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560. doi:10.48550/arXiv.2310.08560.

Penaloza, E., Malenfant, D., Vattikonda, D., Castanyer, R. C., Venkatraman, S., Puri, A., Light, J., Sargent, M. J., et al. (2026). KV-streams for Efficient Compaction in Agentic Reinforcement Learning. arXiv:2609.35750. doi:10.48550/arXiv.2609.35750.

Pfau, J., Merrill, W., & Bowman, S. R. (2024). Let's Think Dot by Dot: Hidden Computation in Transformer Language Models. arXiv:2404.15758. doi:10.48550/arXiv.2404.15758.

Pustovit, A. (2026). Knowledge Packs: Zero-Token Knowledge Delivery via KV Cache Injection. arXiv:2604.03270. doi:10.48550/arXiv.2604.03270.

Roger, F., & Greenblatt, R. (2023). Preventing Language Models From Hiding Their Reasoning. arXiv:2310.18512. doi:10.48550/arXiv.2310.18512.

Shao, R., Li, S. S., Xin, R., Geng, S., Wang, Y., Oh, S., Du, S. S., Lambert, N., et al. (2025). Spurious Rewards: Rethinking Training Signals in RLVR. arXiv:2506.10947. doi:10.48550/arXiv.2506.10947.

Shi, T., Zhu, K., Wang, Z., Jia, Y., Cai, W., Liang, W., Wang, H., Alzahrani, H., et al. (2025). PromptArmor: Simple yet Effective Prompt Injection Defenses. arXiv:2507.15219. doi:10.48550/arXiv.2507.15219.

Skaf, J., Ibanez-Lissen, L., McCarthy, R., Watts, C., Georgiv, V., Whittingham, H., Gonzalez-Manzano, L., Lindner, D., et al. (2025). Large language models can learn and generalize steganographic chain-of-thought under process supervision. arXiv:2506.01926. doi:10.48550/arXiv.2506.01926.

Sun, W., Lu, M., Ling, Z., Liu, K., Yao, X., Yang, Y., & Chen, J. (2025). Scaling Long-Horizon LLM Agent via Context-Folding. arXiv:2510.11967. doi:10.48550/arXiv.2510.11967.

Turpin, M., Michael, J., Perez, E., & Bowman, S. R. (2023). Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. arXiv:2305.04388. doi:10.48550/arXiv.2305.04388.

Wang, L., Li, L., Dai, D., Chen, D., Zhou, H., Meng, F., Zhou, J., & Sun, X. (2023). Label Words are Anchors: An Information Flow Perspective for Understanding In-Context Learning. arXiv:2305.14160. doi:10.48550/arXiv.2305.14160.

Wu, F., Xuan, W., Xing, D., Lu, X., Liu, M., Dong, Y., Harchaoui, Z., & Choi, Y. (2025b). The Invisible Leash: Why RLVR May or May Not Escape Its Origin. arXiv:2507.14843. doi:10.48550/arXiv.2507.14843.

Wu, I., Qu, Y., Setlur, A., & Kumar, A. (2026). Reasoning Cache: Continual Improvement Over Long Horizons via Short-Horizon RL. arXiv:2602.03773. doi:10.48550/arXiv.2602.03773.

Wu, W., Wang, Y., Xiao, G., Peng, H., & Fu, Y. (2024). Retrieval Head Mechanistically Explains Long-Context Factuality. arXiv:2404.15574. doi:10.48550/arXiv.2404.15574.

Wu, X., Li, K., Zhao, Y., Zhang, L., Ou, L., Yin, H., Zhang, Z., Yu, X., et al. (2025a). ReSum: Unlocking Long-Horizon Search Intelligence via Context Summarization. arXiv:2509.13313. doi:10.48550/arXiv.2509.13313.

Xiao, G., Tian, Y., Chen, B., Han, S., & Lewis, M. (2023). Efficient Streaming Language Models with Attention Sinks. arXiv:2309.17453. doi:10.48550/arXiv.2309.17453.

Yan, Y., Jiang, L., Jiang, J., Li, S., Wen, Z., Zhang, Z., Zhou, J., Shao, J., et al. (2026). InftyThink+: Effective and Efficient Infinite-Horizon Reasoning via Reinforcement Learning. arXiv:2602.06960. doi:10.48550/arXiv.2602.06960.

Yan, Y., Shen, Y., Liu, Y., Jiang, J., Zhang, M., Shao, J., & Zhuang, Y. (2025). InftyThink: Breaking the Length Limits of Long-Context Reasoning in Large Language Models. arXiv:2503.06692. doi:10.48550/arXiv.2503.06692.

Yu, H., Chen, T., Feng, J., Chen, J., Dai, W., Yu, Q., Zhang, Y., Ma, W., et al. (2025). MemAgent: Reshaping Long-Context LLM with Multi-Conv RL-based Memory Agent. arXiv:2507.02259. doi:10.48550/arXiv.2507.02259.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y., Song, S., & Huang, G. (2025). Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? arXiv:2504.13837. doi:10.48550/arXiv.2504.13837.

Zhang, P., Liu, Z., Xiao, S., Shao, N., Ye, Q., & Dou, Z. (2024). Long Context Compression with Activation Beacon. arXiv:2401.03462. doi:10.48550/arXiv.2401.03462.

Zhang, Y., Hwang, T., McKeown, K., & Muresan, S. (2026). MEMENTO: Learning Context Management for Long Context Processing Agents. arXiv:2611.00244. doi:10.48550/arXiv.2611.00244.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., et al. (2023). H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. arXiv:2306.14048. doi:10.48550/arXiv.2306.14048.

Zhao, R., Meterez, A., Kakade, S., Pehlevan, C., Jelassi, S., & Malach, E. (2025). Echo Chamber: RL Post-training Amplifies Behaviors Learned in Pretraining. arXiv:2504.07912. doi:10.48550/arXiv.2504.07912.

Zhou, Z., Qu, A., Wu, Z., Kim, S., Prakash, A., Rus, D., Zhao, J., Low, B. K. H., et al. (2025). MEM1: Learning to Synergize Memory and Reasoning for Efficient Long-Horizon Agents. arXiv:2506.15841. doi:10.48550/arXiv.2506.15841.

Zolkowski, A., Nishimura-Gasparian, K., McCarthy, R., Zimmermann, R. S., & Lindner, D. (2025). Early Signs of Steganographic Capabilities in Frontier LLMs. arXiv:2507.02737. doi:10.48550/arXiv.2507.02737.

© 2026 Pranay Mahendrakar. Licensed under CC BY 4.0.

DOI: 10.5281/zenodo.24565011