SCAR: PARALLEL SPAN CROSS-ATTENTION REPAIR FOR CONTEXT DELETION IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deleting a span from a KV-cached LLM context leaves the surviving suffix encoded with content that no longer exists. Re-prefilling the suffix repairs it exactly, but the cost grows with the suffix rather than with the deletion, and selective recomputation still pushes every recomputed token through all backbone layers in sequence. We observe that each layer's cache already holds the information for estimating that layer's repair: the stale suffix states and the deleted-span states. SCAR (Span Cross-Attention Repair) exploits this with one small corrector per layer that cross-attends from the suffix to the deleted span and predicts a residual update to the suffix keys and values. The correctors read only the cache, run no backbone layer, and depend on no other corrector, so all of them run in parallel as a depth-1 repair and leave a standard decoding cache. Across seven task categories on Qwen3-8B and Llama-3.1-8B, SCAR exceeds 15%-budget CacheBlend-style selective recomputation in answer agreement with exact recomputation (consistency) on 13 of 14 backbone–category cells. On four timed Qwen settings, it runs 2.9–4.8$\times$ faster than exact recomputation and 18–33% below that budget's modeled latency. On conflicting-fact retrieval, SCAR achieves near-exact consistency where selective recomputation falls short even at 75% on Qwen and 50% on Llama; on Qwen GSM graphs, variable tracking, and conversation, it exceeds training-pool-matched KVEraser by 24–48 percentage points. At fixed 32k context with 512 deleted tokens, SCAR beats exact recomputation at every tested suffix length of at least 4k, reaching a 4.2$\times$ speedup at 30k. Higher recomputation budgets still recover more fidelity on several settings, at two to three times SCAR's latency. SCAR thus demonstrates that deletion repair can be learned directly from KV cache space and executed in a single parallel stage, decoupling repair latency from backbone depth and making long surviving suffixes cheap to edit.

1 INTRODUCTION

KV caching is critical for efficient LLM serving: a context is prefilled once, and every layer's keys and values are stored so that decoding of future tokens never recomputes it (Vaswani et al., 2017; Kwon et al., 2023; Zheng et al., 2024). A growing line of systems treats this cache as a reusable asset, enabling reuse of separately prefilled segments (Yao et al., 2025; Hu et al., 2024; Liu et al., 2026). Emerging usage patterns, however, demand the inverse operation: removing context after it has been cached. Examples include a retrieval pipeline that retracts a document discovered to be stale, an agent that discards a failed tool call before retrying, and a user who withdraws a wrong instruction mid-conversation. Context deletion reclaims scarce context-window budget and improves decode performance. Removing invalidated or distracting content has the additional benefit of improving answer quality (Laban et al., 2025). These settings need efficient deletion of a middle span from a KV-cached context (Ma et al., 2026; Li et al., 2026).

Note that simply removing the deleted span's KV cache does not remove its influence on the surviving context. Under causal attention, surviving suffix tokens were encoded with access to the span, so their states above the first layer can retain its content. Recomputing the suffix from scratch, re-prefilling the surviving tokens over the still-valid prefix, provides an exact replacement for the KV cache, but its cost grows with the suffix length rather than the size of the deletion, so a small deletion before a long suffix can be costly.

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.

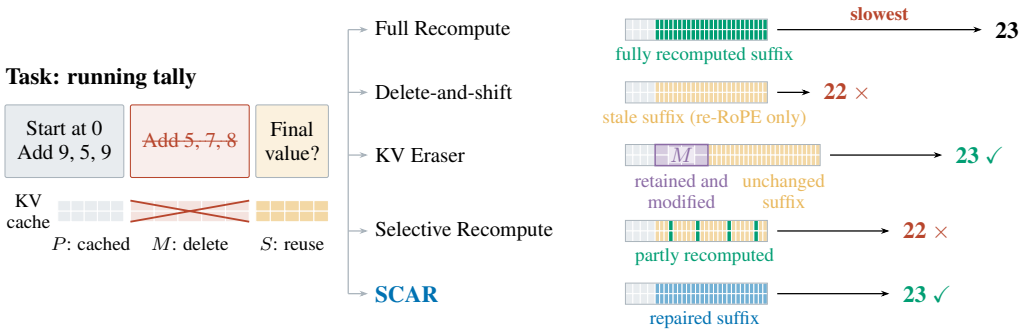


Figure 1: A running-tally example on Qwen3-8B. Prompts are abbreviated.

For a cached context $[P; M; S]$ with prefix P, deleted middle span M, and suffix S, existing methods to avoid re-prefilling all of S largely fall into two families. Figure 1 illustrates how these approaches modify the cache and affect the answer on a running-tally example. The first family leaves the contaminated suffix untouched (or only fixes the positional embedding). For example, delete-and-shift (He et al., 2025; Ma et al., 2026) drops the span's cache entries and corrects the suffix positions; KVEraser (Li et al., 2026) instead fills the deleted interval with learned steering states, trained so that decoding behaves as if the span had never been present. Neither directly repairs the suffix states, although steering can compensate for their effect on future decoding. Without such compensation, delete-and-shift agrees with exact recomputation on only 7.2% of Qwen3-8B needle-retrieval queries and on none for Llama-3.1-8B, where the deleted fact conflicts with a surviving one (Tables 2 and 3). The second family repairs the suffix by selectively re-prefilling a subset of its tokens. We construct deletion baselines in this family by adapting cache-reuse methods (Yao et al., 2025; Hu et al., 2024). Fidelity can improve with the recompute budget, but so does the cost, and because every selected token must still traverse all L layers in order, even a small budget inherits the backbone's full sequential depth. What is missing is a way to repair the suffix without paying for the backbone at all.

Our key observation is that each layer's original cache provides information for estimating its own repair. At deletion time, both the stale suffix states and the deleted-span states are still available at that layer. A corrector can therefore estimate a layer's repair from that layer's cached quantities alone. This estimate does not reconstruct the exact cross-layer computation, but it removes the dependency between layer repairs: no backbone layer runs, and no corrector waits on another.

We introduce SCAR (Span Cross-Attention Repair), a learned cache-repair module that is separate from the frozen LLM backbone and consumes only the deletion-time KV cache. At each layer, SCAR cross-attends from the stale suffix states to the evicted-span states of that same layer and predicts residual updates to the suffix keys and values, as illustrated in Figure 2. Each corrector executes no backbone layer and reads no other corrector's output, so all $L - 1$ correctors run in parallel as a depth-1 repair path, whereas selective re-prefilling sends every selected token through all L layers sequentially.

We evaluate deletion repair as *consistency*, the answer-level agreement with full suffix re-prefill on the same edited context (§2.3), across 10 deletion settings on each of Qwen3-8B and Llama-3.1-8B. The selective baselines are CacheBlend-Del, a deployable deletion adaptation of CacheBlend (Yao et al., 2025; Hu et al., 2024) that re-prefills the top-r fraction of suffix tokens ranked by cache deviation, and Ideal Selective Recompute (ISR), an oracle-assisted variant that ranks by the true deviation from the clean cache and substitutes the clean states of the tokens it selects (§4.1).

SCAR exceeds each selective policy at $r = 0.15$ on 13 of 14 backbone–category cells, including all ten suffix-carried-state cells (Tables 2 and 3). It achieves near-exact needle consistency (1.000 on Qwen, 0.996 on Llama) and exceeds training-pool-matched KVEraser by 24 to 48 points on GSM graphs, variable tracking, and conversation (§4.2).

On four timed Qwen settings, SCAR is 2.9 to 4.8× faster than exact recomputation and 18 to 33% below either selective policy's modeled latency at $r = 0.15$, adding nondominated points on variable

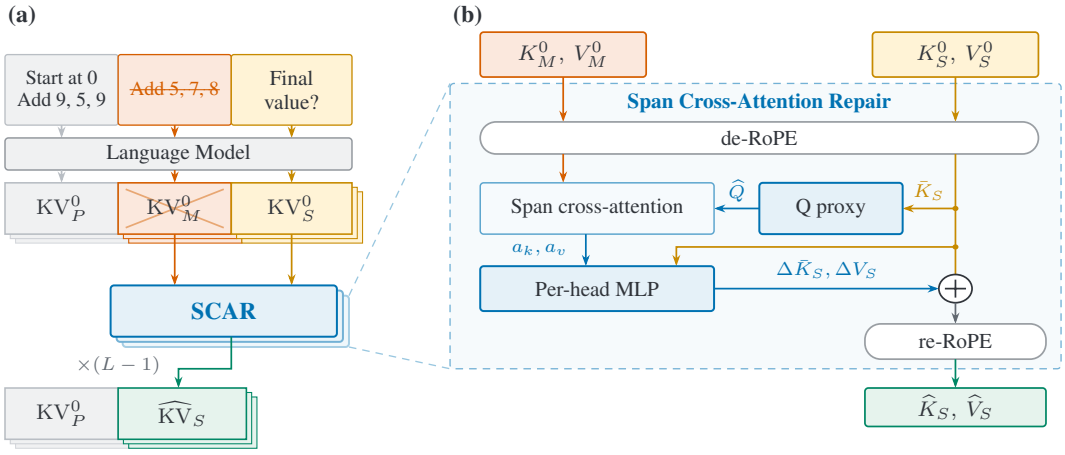


Figure 2: **(a)** The language model prefills the prompt $[P, M, S]$ from Figure 1. The middle span M is then deleted. SCAR repairs the KV cache of the suffix S with $L-1$ independent correctors for all but the first layer. **(b)** A single layer's SCAR corrector. An MLP uses the cross-attention readout from the deleted span to predict key and value residuals for each suffix position. RoPE operations affect keys only.

tracking at both lengths and 32k needle retrieval (Figure 3 and §4.3). At fixed 32k context with a 512-token deletion, it overtakes exact recomputation between 2k and 4k suffix tokens and reaches a 4.2× speedup at 30.7k (Figure 4). A 50% budget recovers more fidelity on several settings at two to three times SCAR's measured latency on the timed settings under the same model, supporting a workload-specific tradeoff.

We release SCAR code, checkpoints, and datasets at https://anonymous.4open.science/r/SCAR-code-ICLR/.

2 Background and Problem Formulation

2.1 Setting and Notation

Consider a cached context $x = [P; M; S]$, with preserved prefix P, deleted span M, and surviving suffix S of lengths n_P, n_M, and n_S. Deleting M produces $\widetilde{x} = [P; S]$ and shifts suffix positions left by n_M tokens. The prefix cache remains valid under causal attention, but suffix states above the first layer may retain information from M even after their positions are corrected. We study one contiguous deletion and the cache used to resume decoding.

2.2 Post-Deletion Cache Repair

Let $C^0 = \mathrm{KV}([P; M; S])$ be the original cache and $C^\star = \mathrm{KV}([P; S])$ the clean cache. For a span $T \in \{P, M, S\}$, $\mathrm{KV}^0_T = (K^0_T, V^0_T)$ denotes its slice of the original cache; hats on cache variables denote repaired states. A repair method constructs $\widehat{C}$ from the context, cache, and deletion boundaries; the cached context contains the complete prompt. For a continuation y generated after deletion, the behavioral target is

$$p_\theta(y \mid \widehat{C}) \approx p_\theta(y \mid C^\star), \tag{1}$$

where θ is the frozen language model. This target permits alternative cache representations with similar outputs.

Our exact reference retains the prefix cache and re-prefills S to produce $C^\star$. Delete-and-shift removes M and adjusts suffix positions without repairing their content. SCAR and the selective-recomputation baselines repair this same position-corrected suffix.

2.3 MEASURING REPAIR QUALITY

Consistency is the fraction of extracted answers that agree with exact suffix recomputation, using greedy decoding and the same task-specific extractor for both caches. It measures agreement on evaluated queries, not distributional equality in Equation (1) or task correctness. Paired extraction failures count as agreement (Appendix A). The reference has consistency 1 by self-comparison, although different numerical prefill paths can produce different greedy answers.

3 SCAR- SPAN CROSS-ATTENTION REPAIR

3.1 REPAIR INPUTS

Each corrector uses span cross-attention from the stale suffix to the deleted span, whose entries remain available until the read completes. Because the original queries and attention weights are not cached, we learn query proxies from the suffix keys. The resulting features require no corrected states from preceding layers (Figure 2).

3.2 PER-LAYER SPAN CROSS-ATTENTION CORRECTOR

SCAR places one corrector at each layer $\ell = 1, \ldots, L-1$. We describe one layer and KV head, omitting their indices for readability.

Separating position changes from learned repair. Let R_p be the RoPE rotation at position p and K_i^0 the original key. We use row vectors and first remove the explicit rotation (de-RoPE): $\bar{K}_i = K_i^0 R_{p_i}^{-\top}$. The corrector predicts an update in these pre-RoPE coordinates; contextual and positional information from earlier layers may remain in the representation.

Reading the deleted span. A bias-free learned projection $W_Q^{(\ell)} \in \mathbb{R}^{D_h \times D_h}$, shared across KV heads within each layer, outputs query proxies $\widehat{Q}$ from the unrotated suffix keys. Span cross-attention computes

$$\widehat{Q} = \bar{K}_S W_Q^{\top}, \qquad A = \mathrm{softmax}\left(\frac{\widehat{Q}\bar{K}_M^{\top}}{\sqrt{D_h}}\right), \tag{2}$$

where D_h is the head dimension and softmax runs over M. The cross-attention readout consists of $a_k = A\bar{K}_M$ and $a_v = AV_M^0$, which summarize the deleted span for each suffix position. Here A denotes implicit attention weights; the implementation computes both readouts without storing the full matrix.

Updating the suffix. For each layer and KV head, two independently parameterized two-layer GELU MLPs predict the key and value residuals. Each receives the feature concatenation $[a_k; a_v; \bar{K}_S; V_S^0]$ of width $4D_h$, uses hidden width $4D_h$, and outputs D_h features. Both linear layers include biases; neither branch uses normalization. We denote their concatenated outputs by $g_{\ell,h}$:

$$[\Delta\bar{K}_S; \Delta V_S] = g_{\ell,h}([a_k; a_v; \bar{K}_S; V_S^0]). \tag{3}$$

We apply the residuals and rotate each key to its edited position (re-RoPE):

$$\widehat{K}_i = (\bar{K}_i + \Delta\bar{K}_i)R_{p_i - n_M}^{\top}, \qquad \widehat{V}_i = V_i^0 + \Delta V_i. \tag{4}$$

Span cross-attention mixes only over the deleted span; suffix positions are repaired independently and joined to the unchanged prefix for decoding. Zero-initialized output projections make the initial corrector reproduce delete-and-shift. Both Qwen3-8B and Llama-3.1-8B use $H_{\mathrm{KV}} = 8$ KV heads per layer and head dimension $D_h = 128$, giving each residual branch dimensions $512 \to 512 \to 128$. Each layer has approximately 5.3M parameters: 35 correctors total 184.4M on Qwen (2.3% of the backbone), and 31 total 163.4M on Llama, independent of context length.

Layer 0 needs no learned correction because its keys and values precede the first attention operation that mixes tokens. We adjust its suffix-key rotations and retain its values.

3.3 TRAINING THE CORRECTOR

We prefill $[P; M; S]$ and $[P; S]$ with the frozen backbone to obtain paired original and clean training caches. The clean target and prefix-cached evaluation reference implement the same edited-context computation, but need not agree numerically in bf16. At each corrected layer, we minimize suffix-cache mean squared error:

$$\mathcal{L}_\ell = \mathrm{MSE}(\widehat{K}_S^{(\ell)}, K_S^{\star(\ell)}) + \mathrm{MSE}(\widehat{V}_S^{(\ell)}, V_S^{\star(\ell)}). \tag{5}$$

Keys are compared at edited positions; equivalent de-rotated-key MSE follows from orthogonality of the shared RoPE rotation. Only corrector parameters are updated, in fp32 over bf16 caches, without task-answer labels. Cache MSE does not guarantee answer agreement: cache distance sometimes increases despite better consistency (Appendix F).

3.4 PARALLEL EXECUTION AND COST

Correctors read only their own layer's original cache and can be grouped into a fixed number of fused kernel launches. The depth-1 path denotes one stage of independent correctors, each with multiple operations; recomputation traverses L backbone layers sequentially.

We implement span cross-attention in Triton using FlashAttention's principle of tiled attention with normalization statistics retained on chip (Dao et al., 2022). Programs operate independently over layer/head groups and suffix-row tiles. A first pass over the deleted span computes softmax normalizers; a second pass recomputes scores and accumulates both key and value readouts, rounding normalized probabilities to bf16 before the readout products. The kernel writes the readouts directly into the residual MLP feature buffer, without materializing the full score or probability matrix. Separate grouped kernels fuse each residual MLP's two linear layers and GELU; final residual additions use fp32.

Span cross-attention requires $O(n_S n_M)$ arithmetic work per KV head per corrected layer, with independent computations across heads and layers. Projections and MLPs add work linear in n_S. Repair memory scales as $O(n_S + n_M)$, in addition to the fixed corrector weights. Appendix E.1 reports measured allocated memory. With 16K suffix tokens and 4K deleted tokens, peak SCAR memory beyond the prepared input buffers is roughly 9 GiB, including corrector weights, temporary buffers, and corrected outputs (Table A11). After repair, this memory can be released. Decoding needs only the prefix and repaired suffix.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Tasks and training. We evaluate 10 deletion settings from seven categories: running tally, GSM operation graphs (Cobbe et al., 2021; Zhou et al., 2025a), synthetic variable tracking, HotpotQA entity counting (Yang et al., 2018), Lost-in-Conversation math (Laban et al., 2025), needle retrieval, and MuSiQue (Trivedi et al., 2022). The first five test suffix-carried computations or commitments; retrieval tasks remove conflicting facts or distractors. Table 1 gives lengths, counts, and extractors. Deleting an intermediate variable assignment breaks a surviving chain, while removing a tally update or an entity-bearing report changes the required total. MuSiQue instead retains the gold evidence and removes a distractor, so the factual answer should stay unchanged. These edits test both propagation of changed state and recovery from stale evidence; task-example tables appear in Appendix A.1.

We train separate correctors for the frozen Qwen3-8B and Llama-3.1-8B backbones on these task families. Training uses cache MSE, Adam at learning rate = 10^{-3}, one record per step, and 7,500 steps. Data order and initialization were selected on development records separate from the reported evaluation records. These splits share task families; running tally was used during development. Appendix B shows training, selection, and baselines.

Baselines and metrics. We compare exact recomputation, delete-and-shift (He et al., 2025), CacheBlend-Del (a deletion adaptation of CacheBlend (Yao et al., 2025)), and oracle-assisted Ideal Selective Recompute (ISR) at different recompute ratio r. Both selective policies rank suffix tokens

Table 1: **Evaluation tasks and deletion settings.** Lengths are median context lengths in canonical tokens; n is the evaluation count (Qwen/Llama for MuSiQue). Answer extractors: I = first signed integer, L = first answer line, V = tracked-variable set. Every record deletes one contiguous span.

setting	deleted span	length	n	answer
Suffix-carried state				
Running tally	extra updates	1.1k	600	I
GSM graphs, op4–7	one operation	0.6k	569	I
VT, short/16k/32k	intermediate hop	2k/16k/33k	600/300/200	V
HotpotQA counting	entity-bearing document	1.4k	600	I
Lost-in-Conversation	premature answer	0.35k	209	L
Retrieval and distractor removal				
Needle, 16k/32k	outdated conflicting fact	16k/33k	140/140	L
MuSiQue	harmful distractor	2.6k	312/101	L

by layer-1 key deviation; CacheBlend-Del recomputes selected tokens, whereas ISR substitutes their oracle states at every layer, making it not deployable. ISR intends to be a stronger, but not optimal, baseline for answer consistency. Qwen additionally uses released and training-pool-matched KVEraser checkpoints (Li et al., 2026); matching the pool does not match architecture or objective. The matched checkpoint's reported coverage is 94–99% on three settings, so its aggregate is not a strictly paired comparison. We average consistency equally over lengths within each category and treat each backbone–category pair as one cell. The primary comparison counts cells in which SCAR exceeds a baseline, and the macro average over the 14 cells, computed from displayed category values without intermediate rounding, is a secondary summary that weights easy and hard categories equally.

Latency protocol. We measure deletion-to-first-token latency, including repair and cache injection but excluding shared original prefill, for Qwen requests on one NVIDIA B200 through a common vLLM connector. All SCAR latency measurements use its Triton implementation. Selective-policy curves use modeled latency to avoid basing the comparison on the overhead of our current selective-recomputation implementation. Both selective policies receive the same modeled latency at each token fraction. We call this model baseline-favorable: on Qwen VT-32k it gives 5.1–7.0$\times$ lower latency than our measured CacheBlend-Del implementation at $r = 0.15/0.5$ and it omits per-layer gathering and cache reinjection, although it is not a proven lower bound on attainable latency; its definition, assumptions, and measured comparisons appear in Appendix D. Timings use three warmups and 12 repetitions per record, taking medians within and then across records.

4.2 Repair consistency

Across Tables 2 and 3, SCAR is more consistent than CacheBlend-Del at $r = 0.15$ on 13 of 14 backbone–category cells and than ISR at $r = 0.15$ on 13 of 14. We compare at $r = 0.15$ because SCAR's measured latency is below the baseline-favorable modeled latency of both selective policies at that budget on all four timed Qwen settings (§4.3); $r = 0.5$ costs two to three times more under the same model and is discussed below. The exceptions are both on MuSiQue: Llama against CacheBlend-Del (0.505 versus 0.515) and Qwen against ISR (0.526 versus 0.538). As a single summary, the macro average over the 14 displayed cells, $\bar{c} = \frac{1}{14}\sum_{b=1}^{2}\sum_{t=1}^{7} c_{b,t}$, is 71.3% for SCAR, 52.6% for CacheBlend-Del, and 54.0% for ISR at $r = 0.15$; needle retrieval contributes about 71% of the gain over either policy, and excluding it the mean gain is 6.4 points over CacheBlend-Del and 5.8 over ISR. Qwen's training-pool-matched KVEraser reaches 56.6%, trailing SCAR on six of seven categories.

Where repair helps. SCAR improves over delete-and-shift in every category on both backbones, with nearly exact needle agreement. Natural counting stays below 40%, and MuSiQue gains are modest (0.485 to 0.505 on Llama). On the suffix-carried-state categories, SCAR recovers 17 to 51% of the gap between delete-and-shift and exact recomputation on Qwen (tally is saturated) and 10 to 75% on Llama, versus 4 to 15% and 5 to 21% for CacheBlend-Del at $r = 0.15$. At $r = 0.5$, ISR and CacheBlend-Del each exceed SCAR on three Qwen and five Llama categories, and ISR's Qwen

Table 2: **Answer consistency on Qwen3-8B** (higher is better). Exact recomputation has consistency 1 by definition and is omitted. VT (variable tracking) and NIAH (needle retrieval) average length settings equally; Macro avg. averages the seven columns. Natural denotes HotpotQA counting; LiC denotes Lost-in-Conversation. Bold marks point estimates, not significance. Matched KVEraser uses partially covered subsets (§4.1).

method	Tally	GSM	VT	Natural	LiC	NIAH	MuSiQue	Macro avg.
SCAR–Qwen	**1.000**	**0.645**	**0.740**	0.387	0.809	**1.000**	0.526	**0.730**
delete-and-shift	0.993	0.506	0.464	0.258	0.670	0.072	0.452	0.488
ISR $r = .15$	0.995	0.532	0.541	0.300	0.742	0.257	0.538	0.558
ISR $r = .5$	0.998	0.629	0.685	0.447	**0.847**	0.650	**0.711**	0.710
CacheBlend-Del $r = .15$	0.998	0.524	0.523	0.308	0.718	0.139	0.506	0.531
CacheBlend-Del $r = .5$	**1.000**	0.580	0.651	0.408	0.833	0.482	0.570	0.646
KVEraser native	0.052	0.483	0.098	0.245	0.651	0.990	0.308	0.404
KVEraser matched	0.972	0.403	0.323	**0.450**	0.325	0.972	0.516	0.566

Table 3: **Answer consistency on Llama-3.1-8B** (higher is better). Aggregation and bolding follow Table 2. KVEraser is omitted because the available checkpoints use Qwen.

method	Tally	GSM	VT	Natural	LiC	NIAH	MuSiQue	Macro avg.
SCAR–Llama	**0.943**	0.529	0.679	0.383	0.842	**0.996**	0.505	**0.697**
delete-and-shift	0.772	0.462	0.614	0.315	0.809	0.000	0.485	0.494
ISR $r = .15$	0.820	0.490	0.657	0.367	0.818	0.004	0.495	0.522
ISR $r = .5$	0.942	**0.622**	**0.747**	**0.552**	0.871	0.085	**0.564**	0.626
CacheBlend-Del $r = .15$	0.820	0.487	0.641	0.358	0.818	0.004	0.515	0.520
CacheBlend-Del $r = .5$	0.933	0.608	0.705	0.518	**0.876**	0.040	0.545	0.604

macro average (0.710) approaches SCAR's (0.730), at a modeled latency two to three times SCAR's measured latency (§4.3). The quality gains therefore motivate a workload-specific latency tradeoff rather than uniform superiority.

4.3 Consistency–latency tradeoff

Figure 3 compares Qwen variable tracking (VT) and needle retrieval (NIAH) at 16k and 32k. Selective policies sweep six fractions with modeled latency; all other latencies are measured (§4.1). The frontier compares point estimates, not statistically resolved quality differences.

Variable tracking. At 16k, SCAR reaches 0.763 consistency in 61.0 ms, versus 179.1 ms for exact recomputation and 39.5 ms for delete-and-shift at 0.347. At $r = 0.05$ both policies are modeled at 58.0 ms, below SCAR, but reach only 0.377 and 0.370; no swept budget through $r = 0.5$ reaches SCAR's consistency (ISR 0.647 and CacheBlend-Del 0.627 at 134.1 ms), and at $r = 0.75$ (modeled 176.4 ms, 2.9× SCAR's latency) CacheBlend-Del ties it exactly (0.763, the same 229 of 300 agreements) while ISR reaches 0.797; matched KVEraser reaches 0.138 in 68.8 ms. At 32k, SCAR reaches 0.780 in 99.3 ms, versus 376.2 ms for exact recomputation and 65.6 ms for delete-and-shift at 0.575. Every modeled selective point is slower than SCAR: at $r = 0.05$ both policies take 103.9 ms and reach 0.595, and neither exceeds SCAR's consistency below $r = 0.75$, where ISR reaches 0.815 and CacheBlend-Del 0.785 at a modeled 352.6 ms, 3.6× SCAR's measured latency. Matched KVEraser reaches 0.210 in 119.4 ms. Our CacheBlend-Del implementation measures 705.2/1,836.8 ms at $r = 0.15/0.5$ on the same 32k records, against modeled 139.4/263.7 ms; the measured implementation alone is not the basis of the comparison.

Under an independent-record model, the marginal 95% Wilson intervals for SCAR and ISR at $r = 0.75$ overlap at both VT lengths (Appendix A.2). They quantify query-sampling uncertainty, not paired differences or training-run variation.

Needle retrieval. SCAR matches all Qwen reference answers, taking 65.6/113.9 ms at 16k/32k versus 211.2/543.0 ms for exact recomputation. Each length has 140 records, giving a marginal

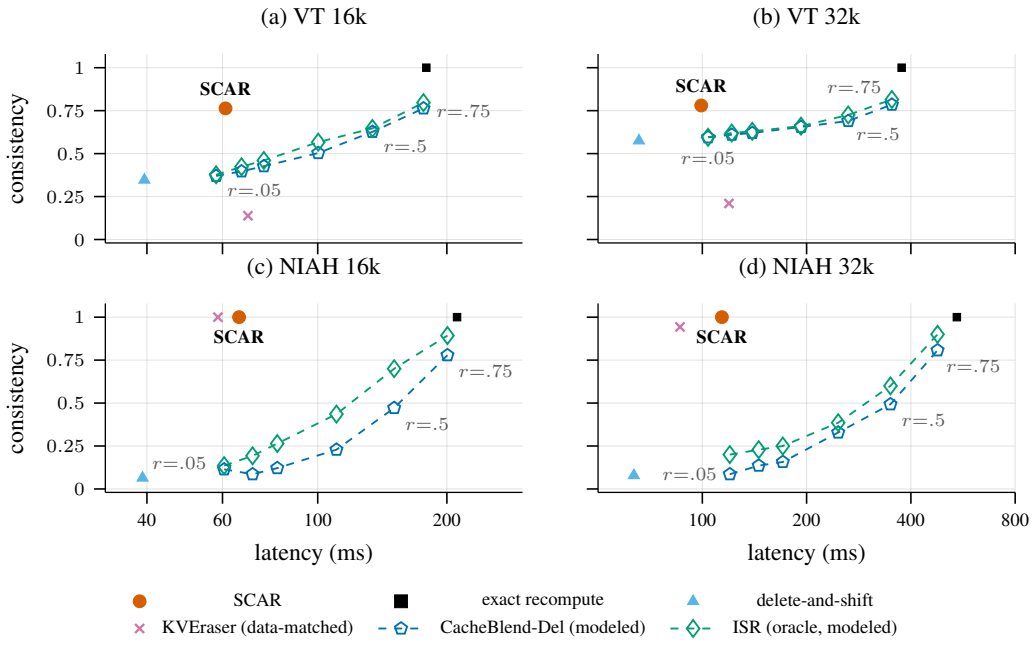


Figure 3: **Consistency versus deletion-to-first-token latency on Qwen3-8B.** CacheBlend-Del and ISR use baseline-favorable modeled latency at six budgets from $r = 0.05$ to 0.75 (§4.1), all other points are measured; latencies are medians over timed samples (Appendix D).

Wilson interval of $[0.973, 1.000]$ under the same sampling model; perfect observed agreement is not a guarantee on other queries. Even at $r = 0.75$, ISR reaches only 0.893/0.900 consistency at modeled latencies of 200.6/477.5 ms. Matched KVEraser reaches 1.000/0.943 in 58.5/86.2 ms: it dominates SCAR at 16k, while 32k offers a quality–latency tradeoff.

Across the four evaluated Qwen settings, SCAR is faster than modeled ISR at $r = 0.15$. Under this model, it adds nondominated points on both VT settings and 32k needle retrieval among the evaluated methods.

4.4 Latency scaling with deletion geometry

At fixed total length 32,768 and deletion length 512, varying the surviving suffix gives a crossover with exact recomputation between 2,048 and 4,096 tokens (Figure 4). Across a 30× suffix-length range, SCAR grows from 70.7 to 147.2 ms (2.1×); exact repair reaches 612.3 ms at $n_S = 30{,}720$, giving a 4.2× speedup. Short suffixes can favor exact recomputation because of corrector overhead.

For small deleted spans of 10, 64, and 256 tokens with $n_S = 16{,}384$, SCAR takes 99.1–104.9 ms; Table A10 in Appendix D reports the full deleted-span latency sweep.

4.5 Component ablations

We retrain each variant with the same 17,557-record pool, fixed order and initialization, 7,500 steps, and cache-MSE objective (Table 4). These single-seed comparisons isolate design changes under one training recipe, not robustness across runs.

Removing the deleted-span readout lowers consistency by 10.4 points on GSM, 10.6 on VT-16k, and 3.9 on LiC. Attention is still computed but its readout is discarded, making this a feature ablation with no latency claim (Appendix C). The learned query proxy helps VT-16k and LiC but slightly hurts GSM. Among tested MLP depths, two layers perform best on all three tasks; one layer falls below delete-and-shift on GSM. Five layers increase parameters from 184.4M to 625.7M without improving quality under the fixed step budget; different training could still help. Appendix F reports secondary cache-space diagnostics.

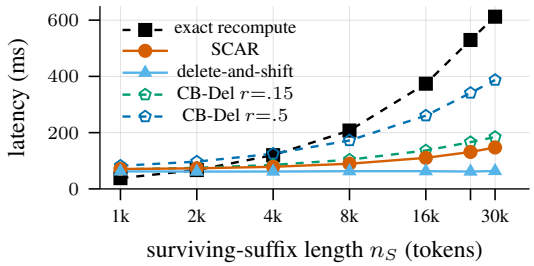


Figure 4: **Latency versus surviving-suffix length** at a fixed 32k context with a 512-token deleted span. SCAR overtakes exact recomputation between 2k and 4k surviving tokens. CB-Del curves use modeled latency; other points are measured medians over 10 records (Appendix D).

Table 4: **Retrained component ablations on Qwen3-8B.** Each variant changes one component under the same training recipe (Appendix C). Bold marks the column maximum.

variant	Q proxy	span readout	MLP layers	GSM	VT-16k	LiC
SCAR	✓	✓	2	0.645	**0.763**	**0.809**
no Q proxy		✓	2	**0.654**	0.713	0.770
no readout			2	0.541	0.657	0.770
1-layer MLP	✓	✓	1	0.487	0.683	0.679
5-layer MLP	✓	✓	5	0.608	0.683	0.584

5 Related work

KV eviction (Xiao et al., 2024; Li et al., 2024; Zhang et al., 2023; Qin et al., 2025) retains selected cache entries for approximate attention during inference, reducing storage and computation costs.

Position-independent caching (Yao et al., 2025; Agarwal et al., 2025; Zhou et al., 2025b; Hu et al., 2024; Wang et al., 2026; Liu et al., 2026; Yang et al., 2025) reuses precomputed cache chunks with uncertain relative positions, typically recomputing selected tokens to approach full-recomputation performance while retaining the efficiency of cache reuse.

Context deletion. Several recent works study removing the middle portion of a long context while reusing the KV cache of the remaining tokens. PIE(delete-and-shift) (He et al., 2025) rotates suffix keys to correct the positional-encoding mismatch after middle-context deletion, a technique also used by Ma et al. (2026). KVEraser (Li et al., 2026) replaces the deleted span with steering states while leaving the suffix cache unchanged. SCAR directly repairs the surviving suffix KV cache through per-layer correctors without backbone replay. After repair, the deleted span's KV entries can be released, shortening the decoding cache.

6 Limitations and discussion

SCAR is approximate and workload-dependent: higher recomputation budgets win several tasks, natural counting and MuSiQue remain difficult. Training covers the evaluated task families; initialization and data order were selected using separate development records.

We test one deletion per record and one-request latency on one B200; repeated edits, concurrency, batching, and other hardware remain untested. Consistency measures agreement with a reference that may itself be wrong and includes paired extraction failures. It does not certify that deleted information is absent from repaired states or inaccessible to other queries, and provides no privacy guarantee.

7 Conclusion

SCAR repairs suffix K/V directly through independent per-layer correctors, avoiding backbone replay after context deletion. Wins over deployable selective recomputation on 13 of 14 backbone–category cells, nondominated Qwen operating points on variable tracking and 32k needle retrieval, and retrained ablations support this architecture as a useful option for long surviving suffixes. The evidence favors workload-specific quality–latency tradeoffs; exact or higher-budget recomputation remains preferable when fidelity is paramount.

AI USE DISCLOSURE

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, implement methods, and support qualitative and thematic data analysis. We have not used generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, clean or reformat datasets, or interpret results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, create artifacts, draft parts of the research paper, summarize or analyze existing literature, source or search for information, and edit the research paper to improve readability. We reviewed all AI-assisted text for accuracy, clarity, and alignment with the work. We checked technical claims, methods, and interpretations against our experiments and analysis; tested and verified AI-assisted code and artifacts for correctness. We did not use AI-generated figures or images. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The anonymous repository linked in §1 contains the SCAR code, trained checkpoints, evaluation datasets, and the raw data behind every table and figure (`https://anonymous.4open.science/r/SCAR-code-ICLR/`). Task construction and scoring are described in Appendix A, training and baseline protocols in Appendix B, the latency protocol in §4.1 and Appendix D, and the kernel implementation in Appendix E.

REFERENCES

Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, and Shiv Saini. Cache-craft: Managing chunk-caches for efficient retrieval-augmented generation. *Proc. ACM Manag. Data*, 3(3), June 2025. doi: 10.1145/3725273. URL https://doi.org/10.1145/3725273.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2205.14135.

Zhenyu He, Jun Zhang, Shengjie Luo, Jingjing Xu, Zhi Zhang, and Di He. Let the code LLM edit itself when you edit the code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zqzsZ5cXbB.

Junhao Hu et al. EPIC: Efficient position-independent caching for serving large language models. *arXiv preprint arXiv:2410.15332*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

Mufei Li, Shikun Liu, Dongqi Fu, et al. KVEraser: Learning to steer KV cache for efficient localized context erasing. *arXiv preprint arXiv:2606.17034*, 2026. URL https://github.com/Graph-COM/KVEraser.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yang Liu et al. CacheSlide: Unlocking cross position-aware KV cache reuse for accelerating LLM serving. In *USENIX Conference on File and Storage Technologies (FAST)*, 2026.

Ma, Eitzinger, and Koestler. Leyline: KV cache directives for agentic inference. *arXiv preprint arXiv:2606.01065*, 2026.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EQgEMAD4kv.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics (TACL)*, 10:539–554, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Shihao Wang et al. ProphetKV: User-query-driven selective recomputation for efficient KV cache reuse in retrieval-augmented generation. *arXiv preprint arXiv:2602.02579*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

Jingbo Yang et al. KVLink: Accelerating large language models via efficient KV cache reuse. *arXiv preprint arXiv:2502.16002*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for retrieval-augmented generation with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025. arXiv:2405.16444.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VqkAKQibpq.

Yang Zhou et al. GSM-Infinite: How do your LLMs behave over infinitely increasing context length and reasoning complexity? In *International Conference on Machine Learning (ICML)*, 2025a. arXiv:2502.05252.

Yuechi Zhou et al. A^3: Attention-aware accurate KV cache fusion for fast large language model serving. *arXiv preprint arXiv:2511.17560*, 2025b.

A EVALUATION DETAILS

A.1 TASK EXAMPLES

We show one selected example from each of the seven task categories in Table 1, using saved Qwen3-8B outputs for running tally and Llama-3.1-8B-Instruct outputs for the other tasks. Each example was chosen so that SCAR agrees with full recomputation; these cases illustrate the tasks and are not a representative performance sample. Prompts are abbreviated, with omissions marked in brackets; all outputs were generated from the complete saved prompts. The tally example is shared with Figures 1 and 2 and includes the eight approximate methods in Table 2, together with full recomputation. The other tables include the six approximate methods in Table 3, together with full recomputation; the evaluated KVEraser checkpoints use Qwen.

Bold outputs match the full-recompute answer displayed in each table. We display the first signed integer for tally, GSM, and entity counting; the variable-name set from the first nonempty answer line for variable tracking; and the normalized answer line for retrieval and question answering. For conversation, we display the final numeric answer. The quantitative evaluation uses the consistency metric defined in §2.3. The reference is the saved full-recompute method's output on the edited prompt.

Table A1: Running tally on Qwen3-8B.

Retained prefix P	[Filler omitted.] A counter starts at 0. Add 9. Add 5. Add 9.
Deleted span M	[Filler omitted.] ~~Add 5. Add 7. Add 8.~~
Retained suffix S	The running subtotal was recorded above.
Question	What is the counter's final value? Reply with the number only.
Reference answer	**23**: $9 + 5 + 9$. The original total was 43.
Method	**Output (bold: consistent)**
Full recomputation	**23**
Delete-and-shift	22
ISR $r = 0.15$	22
ISR $r = 0.5$	22
CacheBlend-Del $r = 0.15$	22
CacheBlend-Del $r = 0.5$	**23**
KVEraser (native)	30
KVEraser (data-matched)	**23**
SCAR	**23**

Table A2: GSM operation graphs.

Retained prefix P	[Demonstrations and unrelated animal counts omitted.]
Deleted span M	~~The number of adult crow in Pine Ridge equals the number of adult blue jay in Pine Ridge.~~
Retained suffix S	The number of adult blue jay in Pine Ridge equals 9. [Unrelated counts omitted.]
Question	What is the total number of adult animals in Pine Ridge?
Reference answer	**9**. This task removes the deleted animal category from the aggregation graph; the surviving blue-jay count is 9, versus an original total of 18.
Method	**Output (bold: consistent)**
Full recomputation	**9**
Delete-and-shift	18
ISR $r = 0.15$	18
ISR $r = 0.5$	**9**
CacheBlend-Del $r = 0.15$	18
CacheBlend-Del $r = 0.5$	**9**
SCAR	**9**

Table A3: Synthetic variable tracking.

Retained prefix P	[Demonstrations and unrelated filler text omitted.] `VAR LXSOO = 11095.` [Unrelated filler text omitted.]
Deleted span M	~~`VAR VCXAX = VAR LXSOO.`~~
Retained suffix S	[Unrelated filler text omitted.] `VAR JJYSP = VAR VCXAX.` [Unrelated filler text omitted.] `VAR RZXXT = VAR JJYSP.` [Unrelated filler text omitted.]
Question	Find all variables that are assigned the value 11095 in the text above.
Reference answer	**LXSOO**. Deletion breaks the assignment chain to the other three variables.
Method	Output (bold: consistent)
Full recomputation	**LXSOO**
Delete-and-shift	LXSOO, RZXXT
ISR $r = 0.15$	LXSOO, RZXXT
ISR $r = 0.5$	LXSOO, RZXXT
CacheBlend-Del $r = 0.15$	LXSOO, RZXXT
CacheBlend-Del $r = 0.5$	LXSOO, RZXXT
SCAR	**LXSOO**

Table A4: HotpotQA entity counting.

Retained prefix P	[Reports on Kenema, Oriental, and Ostrava; none mentions Rabat.]
Deleted span M	[Contiguous reports on Baghdad, Casablanca, and Rabat.] The Rabat report begins: ~~Rabat [. . .] is the capital city of Morocco [. . .].~~
Retained suffix S	[Reports on Osaka and Tan-Tan.] Driss Lachgar [. . .], born 1954, Rabat [. . .]. [Report on Iraan, Texas.]
Question	How many of the reports in the briefing above mention Rabat?
Reference answer	**1**. Only the Driss Lachgar report still mentions Rabat; the original count was 2.
Method	Output (bold: consistent)
Full recomputation	**1**
Delete-and-shift	2
ISR $r = 0.15$	2
ISR $r = 0.5$	**1**
CacheBlend-Del $r = 0.15$	2
CacheBlend-Del $r = 0.5$	**1**
SCAR	**1**

Table A5: Lost-in-Conversation math.

Retained user turn P	Calculate the distance between the bird's northern and southern homes. A bird is migrating from its summer home in the north to its winter home in the south. First, the bird flies south for 10 hours at 30 mph.
Deleted assistant turn M	[Reasoning omitted.] ~~So, the distance between the bird's northern and southern homes is 300 miles.~~
Retained user turn S	Next, the bird changes direction and flies north for 2 hours at 18 mph. Finally, the bird turns south again and flies for 5 hours at 22 mph.
Reference answer	**374 miles**: $10 \times 30 - 2 \times 18 + 5 \times 22$. The question is in the first user turn; the second supplies the remaining facts.
Method	Output (bold: consistent)
Full recomputation	**374 miles**
Delete-and-shift	74 miles
ISR $r = 0.15$	410 miles
ISR $r = 0.5$	**374 miles**
CacheBlend-Del $r = 0.15$	74 miles
CacheBlend-Del $r = 0.5$	74 miles
SCAR	**374 miles**

Table A6: Needle retrieval (16k context).

Retained prefix P	Extract the requested number(s) from the text. [Background omitted.]
Deleted span M	~~One of the special magic numbers for query is: 6490234.~~
Retained suffix S	[Background omitted.] One of the special magic numbers for query is: 3619435. [Background omitted.]
Question	Find number(s) in sentence(s) of the form: "One of the special magic numbers for query is: ⟨NUMBER⟩." Output the ⟨NUMBER⟩(s) in order as digits only, comma-separated, no spaces, no other text.
Reference answer	**3619435**. Extra numbers in the selective-recomputation outputs make them inconsistent with this reference.
Method	Output (bold: consistent)
Full recomputation	**3619435**
Delete-and-shift	649,000,000, 1,000,000, 7,
ISR $r = 0.15$	649,000,000, 1,000,000, 7,
ISR $r = 0.5$	3619435, 1000000, 7
CacheBlend-Del $r = 0.15$	649,000,000, 1,000,000, 7,
CacheBlend-Del $r = 0.5$	3619435, 1000000
SCAR	**3619435**

Table A7: MuSiQue question answering.

Retained prefix P	Zena Tsarfin graduated with dual degrees in journalism and political science from Brooklyn College. [Other context omitted.]
Deleted span M	[An unrelated Avicenna paragraph, beginning:] ~~Ibn Sina created an extensive corpus of works during what is commonly known as the Islamic Golden Age [. . .].~~
Retained suffix S	[Other context omitted.] Brooklyn College is a public college in Brooklyn, New York City. It is part of the City University of New York. [Other context omitted.]
Question	What company is Zena Tsarfin's school a part of?
Reference answer	**The City University of New York.** Both supporting facts survive; all methods agree with full recomputation in this example.
Method	Output (bold: consistent)
Full recomputation	**The City University of New York.**
Delete-and-shift	**The City University of New York.**
ISR $r = 0.15$	**The City University of New York.**
ISR $r = 0.5$	**The City University of New York.**
CacheBlend-Del $r = 0.15$	**The City University of New York.**
CacheBlend-Del $r = 0.5$	**The City University of New York.**
SCAR	**The City University of New York.**

Scoring and uncertainty. Consistency compares the task-specific extracted answer with exact prefix-cached suffix recomputation on the same record. `None == None` counts as consistent, so agreement can include paired extraction failures. The reference has consistency 1 by definition. Separately recomputed bf16 caches can nevertheless produce different greedy continuations; retaining a prefix cache and prefilling the edited prompt from scratch need not give bitwise-identical states. Cache-space diagnostics therefore distinguish numerical variation from exact equality (Appendix F). Tables and frontiers compare point estimates; boldface does not indicate statistical significance.

A.2 Uncertainty from reported agreement counts

We calculate nominal 95% Wilson score intervals from the reported full-pool proportions and denominators, without rerunning inference. For x agreements among n records, let $\widehat{p} = x/n$ and $z = 1.96$:

$$\frac{\widehat{p} + z^2/(2n) \;\pm\; z\sqrt{\widehat{p}(1-\widehat{p})/n + z^2/(4n^2)}}{1 + z^2/n}.$$

The printed three-decimal rates and denominators uniquely determine the integer counts in Table A8; matching every needle reference implies $x = n = 140$ at each length. These are marginal intervals under an independent Bernoulli sampling model, conditional on the fitted corrector, reference, and scoring rule. They concern similarly sampled queries, not uncertainty in the already observed finite-pool proportions. They do not account for dependent records, training seeds, numerical variation across repeated inference, or latency, and are not adjusted for simultaneous coverage across settings. Without paired discordance counts, they do not test differences between methods or establish uncertainty over frontier membership.

Table A8: Analytical query-sampling uncertainty from the reported counts. Intervals are marginal 95% Wilson intervals, not paired method comparisons. Needle denotes each of the 16k and 32k pools separately.

Setting	Method	x/n	Consistency	Interval
VT 16k	SCAR	229/300	0.763	$[0.712, 0.808]$
VT 16k	ISR $r = .75$	239/300	0.797	$[0.748, 0.838]$
VT 16k	CacheBlend-Del $r = .75$	229/300	0.763	$[0.712, 0.808]$
VT 32k	SCAR	156/200	0.780	$[0.718, 0.832]$
VT 32k	ISR $r = .75$	163/200	0.815	$[0.755, 0.863]$
VT 32k	CacheBlend-Del $r = .75$	157/200	0.785	$[0.723, 0.836]$
Needle, each length	SCAR	140/140	1.000	$[0.973, 1.000]$

Consistency and task accuracy. Consistency measures agreement with the edited-context reference, including its errors; task accuracy instead compares the output with the benchmark label. Needle accuracy uses substring inclusion whereas consistency uses the normalized answer, so a generation containing both the surviving answer and stale content can pass accuracy while failing consistency. On Llama, deleting the premature Lost-in-Conversation answer does not improve task accuracy; that setting tests recomputation agreement only. Related length settings are averaged within each category before categories and backbones receive equal weight in the main aggregate.

B TRAINING, BASELINES, AND COVERAGE

Training. SCAR uses Adam with learning rate 10^{-3}, one record per step, and 7,500 steps. The objective weights mean squared key and value errors over suffix entries equally, with no auxiliary attention or task-label loss. The implementation compares de-rotated keys. Applying the same orthogonal edited-position rotation to the prediction and target preserves squared error, giving the objective in Equation (5). The backbone is frozen; training runs the corrector in fp32 over bf16 caches with zero-initialized output projections. Inference uses mixed precision as described in Appendix E.

The Qwen training pool contains 17,557 records: 7,500 arithmetic graphs (including 2,069 repeated records), 4,000 natural-counting records, 1,500 needle records, 750 variable-tracking records, 656 MuSiQue records, 570 controls, 1,813 no-survivor probes, 560 tallies, and 208 Lost-in-Conversation records. Training consumes the first 7,500 records in the pinned shuffled order, rather than a full pass through this pool. Llama uses 17,608 records with the same family counts except for 259 model-specific conversation records; needle and MuSiQue inputs are also prepared for that backbone.

Qwen's data order (seed 0) was selected post hoc from five observed draws using deeper-arithmetic results. The pinned order was evaluated at three fresh initializations, and initialization 6 was selected for its Lost-in-Conversation performance. Llama uses initialization 0 and data-order seed 0, selected from three orders using GSM results. The records used to select data order and initialization are separate from the records reported in the main evaluation tables; evaluation records were not used for those choices. Development and evaluation nevertheless share task families, so record separation does not establish unseen-family generalization. Running-tally training and evaluation records are index-disjoint, but tally was used during development. Corrector weights do not transfer between backbones.

Selective-recomputation baselines. Both policies start from delete-and-shift and select $\mathrm{round}(rn_S)$ suffix positions with the largest fp32 L_2 differences between fresh and position-

corrected stale keys at layer 1. For each position, the squared differences are summed over KV heads and head dimensions. CacheBlend-Del computes all suffix positions through layers 0 and 1 to obtain these scores, then runs only selected positions through deeper layers. Attention reads a mixture of fresh selected entries and stale unselected entries under the causal mask. ISR obtains keys and values from exact suffix recomputation and substitutes the selected positions at every layer. Its oracle states are unavailable in a deployable selector; its key-based ranking does not optimize answer consistency and need not dominate CacheBlend-Del on every record.

KVEraser protocol and coverage. KVEraser replaces the deleted interval with learned, length-preserving steering K/V and retains the stale suffix. We evaluate the released Qwen checkpoint through the authors' interface and a Qwen-backbone retrain on the same 17,557-record mix as SCAR. Matching the training pool does not match architecture, objective, or interface. Neither checkpoint supports Llama without a new backbone-specific retrain. The Qwen comparison contains eight approximate methods and the exact reference; Llama contains six approximate methods and the exact reference. The data-matched eraser covers 94–99% of record IDs on three Qwen settings, and its reported rates use those covered subsets rather than the complete evaluation pools.

C Retrained ablations

Each Qwen variant uses the same training pool, pinned data order, initialization seed, 7,500-step budget, and cache-MSE objective. Only the component named in Table 4 changes, and layer 0 receives no learned correction. Removing the cross-attention readout is a feature ablation: attention is still computed but its readout is discarded, so this arm makes no latency claim. All variants use one initialization. The five-layer MLP increases parameters from 184.4M to 625.7M without increasing the training-step budget. The results support the chosen two-layer head under this recipe; they do not establish that greater capacity cannot help with a different training budget.

D Latency methodology

Rig and aggregation. All timings use one request at a time on one NVIDIA B200 through a common vLLM (Kwon et al., 2023) connector, with three warmups and 12 timed repetitions per record. Deletion-to-first-token latency includes repair or recomputation and cache injection, and excludes the shared original prefill. Reported times are medians across records of each record's median over timed repetitions. For the frontier experiments, needle quality uses all $n = 140$ records per length, while latency uses $n = 20$ timed records per length. The VT-16k and VT-32k timings use 25 records each, whereas their full quality pools contain 300 and 200 records. SCAR serving measurements use PyTorch 2.11, Triton 3.6, vLLM 0.23, and CUDA 13. Baseline measurements come from separate campaigns on the same GPU model and retain their paired delete-and-shift and recomputation anchors; the modeled curves use those anchors consistently.

Selective-recomputation latency model. For each record i, both selective policies receive the same modeled latency at token fraction r:

$$t_{\mathrm{model},i}(r) = d_i + f(r)q_i, \qquad f(r) = \frac{\mathrm{sel} + 1 + r(L - \mathrm{sel} - 1)}{L},$$

where d_i and q_i are the measured per-record median delete-and-shift and prefix-cached suffix-recomputation latencies. We compute modeled latency per record and then report the median across records. For Qwen, $\mathrm{sel} = 1$ and $L = 36$, so $f(r) = (2 + 34r)/36$: two full selector layers followed by the selected token fraction in the remaining 34 layers. This gives $f(.15) = 0.1972$, $f(.5) = 0.5278$, and $f(0) = 0.0556$.

The model approximates recomputation work using layer counts and token fractions, assumes linear latency scaling with this work fraction, and omits additional per-layer gathering and cache reinjection costs. The modeled latencies are lower than those of our tested implementation on matched records (Table A9), but are not proven lower bounds on attainable selective-recomputation latency. Frontier comparisons remain conditional on these modeling assumptions.

For ISR, assigning the same modeled latency provides a common recomputation-budget comparison; the cost of obtaining its oracle cache states is excluded.

Measured selective recomputation. On the same 25 Qwen VT-32k records used for latency in Figure 3, measured CacheBlend-Del takes $5.1\times$ and $7.0\times$ the modeled latency at $r = 0.15$ and 0.5, respectively (Table A9). The modeled latency therefore gives the tested baseline a favorable comparison. Separate profiling identifies the selector forward pass and sparse attention as the main repair costs.

Table A9: Modeled and measured CacheBlend-Del latency on Qwen3-8B VT-32k, using the same 25 timed records and model anchors as Figure 3. Measured latencies and model inputs are per-record medians over 12 uninstrumented repetitions; both columns report medians across records.

r	Modeled (ms)	Measured (ms)
0.15	139.4	705.2
0.5	263.7	1,836.8

Deletion geometry. The surviving-suffix sweep in Figure 4 fixes total context length at 32,768 tokens and deleted-span length at 512 tokens, changing the prefix length as the suffix varies. Geometry points use one seed and 10 records each; the largest per-point IQR in the suffix sweep is 0.35% of its median (rounded up).

Deleted-span length increases attention work; streaming attention avoids allocating storage proportional to $n_M n_S$ (Appendix E.1). With $n_S = 16{,}384$ fixed, the measured deleted-span sweep covers $n_M \in \{10, 64, 256, 1{,}024\}$ (Table A10). SCAR takes 99.1–122.2 ms versus 369.7–392.3 ms for exact recomputation. At $n_M = 1{,}024$, its 122.2 ms is about 8.8% below the 134.0 ms modeled $r = 0.15$ baseline; this margin narrows as the deleted span grows. This sweep measures latency, not preservation of quality across geometries.

Table A10: **Deleted-span sweep at fixed 32k context** ($n_S = 16{,}384$; median ms, 10 records per point, single seed). The final two columns are modeled; other timings are measured.

n_M	Recompute	SCAR	Delete/shift	KVEraser	CacheBlend-Del (modeled)	
					$r = 0.15$	$r = 0.5$
10	392.3	99.5	63.5	117.1	140.8	270.5
64	377.4	99.1	62.3	119.1	136.7	261.4
256	384.5	104.9	65.1	120.6	140.5	266.9
1024	369.7	122.2	61.2	188.0	134.0	256.2

E Implementation and memory

Streaming attention. SCAR follows FlashAttention's principle of tiled attention with on-chip normalization statistics (Dao et al., 2022). Each Triton program handles a block of 32 suffix tokens for one KV head in one layer being repaired. It streams through the deleted span in blocks of 64 tokens. The kernel uses four warps per program and two pipeline stages. The first pass accumulates stable row maxima and softmax denominators in fp32; the second recomputes score tiles and accumulates both key and value readouts. Query–key products accumulate in fp32 and round to bf16 before fp32 scaling. Completed normalizers are used before rounding probabilities to bf16, so the two passes preserve this rounding boundary. Readouts accumulate in fp32 and round to bf16 directly into the concatenated feature buffer. No full score or probability matrix is allocated. Query projection and fused residual MLPs use bf16 matrix operands with fp32 accumulation; corrected K/V outputs are fp32. Address arithmetic uses 64-bit offsets before stride multiplication, and partial tiles are masked. Layer 0 receives no learned correction.

Numerical validation. Synthetic tests compare attention readouts, corrected K/V, and correction deltas with an untiled evaluation of the same mixed-precision operations; small cases also use an fp32 reference. All 38 small numerical cases, nine integration cases, and 18 full-geometry comparisons pass. The tests include tile boundaries, empty spans, concentrated and uniform attention, cancellation, and layer-0 handling. The full-geometry comparisons use seeds 0, 1, and 2 across

the six shapes in Table A11, with the reference evaluated in 512-row suffix chunks. Their largest K/V correction-delta relative L_2 error is 4.14×10^{-5} and smallest cosine is 0.9999999991. These synthetic checks establish numerical agreement within tolerance, not bitwise equality or identical generated answers; they do not rerun the task-quality experiments.

E.1 CORRECTOR MEMORY

Let $G = (L-1)H_{\mathrm{KV}}$ be the number of corrected-layer/head groups. Attention requires $O(Gn_S n_M D_h)$ arithmetic but $O(GD_h(n_S + n_M))$ storage, plus fixed corrector weights. Normalization state stays on chip, while projected queries, features, and outputs grow linearly with suffix length.

Table A11: **Measured SCAR corrector memory (GiB).** Input buffer counts the prepared fp32 K/V tensors for the suffix and deleted span. SCAR memory is the complete-corrector peak allocation minus these input-buffer bytes; it includes corrector weights, temporary buffers, corrected outputs, and other resident allocations. The two columns sum to the total peak, up to rounding.

n_S	n_M	Input buffer	SCAR memory
4,096	1,024	1.367	2.565
16,384	64	4.392	9.127
16,384	256	4.443	9.129
16,384	1,024	4.648	9.127
16,384	4,096	5.469	9.127
32,768	1,024	9.023	17.877

Table A11 uses one B200, 35 corrected layers, eight KV heads, $D_h = 128$, and separate K/V MLPs of hidden width 512. Synthetic input caches and corrected outputs are fp32, matrix weights and attention features are bf16; one GiB is 2^{30} bytes. Each shape's complete-corrector measurement runs in a fresh process using PyTorch 2.12, Triton 3.7, and CUDA 13. After compilation and warmup, outputs and unused allocations are cleared and the GPU is synchronized; each of three forwards resets the peak counters, and the table reports the maximum allocated peak. Numerical comparisons run outside measured regions.

These are corrector measurements: they exclude backbone weights, prefix caches, the serving cache pool, and CUDA allocations outside PyTorch. The input buffer counts prepared fp32 corrector inputs; SCAR memory reports peak allocation beyond these inputs, including corrector weights and corrected outputs. This accounting measures the standalone corrector rather than the additional memory of the full serving pipeline. Increasing n_M grows the input buffer and attention-stage workspace. Across the tested deletion lengths, peak SCAR memory is dominated by suffix feature and output buffers. Allocation inspection confirms that no full $n_S \times n_M$ score or probability tensor is allocated.

F CACHE-SPACE DIAGNOSTICS

Cache-space error helps characterize the learned correction, but does not fully explain behavioral quality. Across the Qwen task battery, the reported relative suffix residuals range from 0.84 to 1.26 on settings where SCAR improves consistency, with 1 denoting the delete-and-shift residual. Thus, better answer agreement can occur even when the overall cache distance changes little or increases. Numerical differences between prefix-cached and from-scratch recomputation further complicate small residual comparisons. These observations support evaluating the behavioral target directly, while using cache distance as a diagnostic of the training objective.

Running tally provides an example where the learned correction improves several cache-space measures together. In Table A12, SCAR reduces the key L_2 residual from 0.3008 to 0.1419 and achieves a correction cosine of 0.7557. Its median KL@16 is 0.0017, compared with 0.0691 for delete-and-shift and 0.0145 for ISR at $r = 0.5$. ISR has lower L_1 residuals, while SCAR has lower L_2, L_∞, and continuation KL on these records. The metrics therefore distinguish the repaired states even though task accuracy is saturated for these methods on this example.

Table A12: **Cache-space diagnostics on Qwen3-8B running tally ($n = 100$).** Suffix residuals are measured in fp32 against the clean cache. L_1 and L_2 average per-record residual statistics across layers; L_∞ is the maximum over records and layers. Δcos measures alignment between the applied and target corrections, and KL@16 is the teacher-forced KL over the first 16 continuation tokens in nats. The recomputation row compares prefix-cached and from-scratch prefill to measure numerical variation. Bold marks the best value among the five approximate methods.

method	L_1		L_2		L_∞ (max)		Δcos		KL@16	
	K	V	K	V	K	V	K	V	med	mean
SCAR	0.0883	0.1039	**0.1419**	**0.1481**	**12.92**	**27.47**	**0.7557**	**0.7465**	**0.0017**	**0.0021**
delete-and-shift	0.1743	0.1908	0.3008	0.2730	16.06	30.98	0.0000	0.0000	0.0691	0.0782
ISR $r = .15$	0.1326	0.1513	0.2398	0.2328	16.06	30.98	0.1596	0.1551	0.0476	0.0562
ISR $r = .5$	**0.0767**	**0.0931**	0.1597	0.1706	14.88	30.98	0.4780	0.4646	0.0145	0.0174
CacheBlend-Del $r = .15$	0.1411	0.1613	0.2444	0.2374	16.06	30.98	0.1471	0.1414	0.0534	0.0596
recompute (floor)	0.0126	0.0189	0.0194	0.0255	0.78	4.94	0.9931	0.9651	0.0003	0.0004