KV-FOLD: BEYOND CHUNKED PREFILL WITH MANIPULABLE KV STATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference in pretrained transformers relies on a growing key–value (KV) cache that is usually treated as an internal artifact of attention. We instead view the KV cache as an explicit *computational state* that can be carried, modified, and transferred between computations. Standard chunked prefill is the identity case of this view: carrying the KV state unchanged across chunks is exactly equivalent, in exact arithmetic, to a monolithic causal forward pass. We prove this equivalence and verify it across several pretrained transformer families, with next-token loss matching full attention to within 0.002 nats.

This exact identity provides a controlled reference point for studying what happens when the carried state is changed. Without modifying model parameters, we quantize, selectively retain, re-address, and transfer KV state. The unmodified state preserves exact factual retrieval across contexts up to 128K tokens and 511 chunk transitions while reducing peak working memory enough to process a context for which the corresponding monolithic forward does not fit on the same hardware. Under lossy transformations, per-step quantization reveals a graded memory–fidelity trade-off, whereas position removal is substantially less forgiving.

Most strikingly, KV state can serve as a sparse message between otherwise separate inference processes. At 32K tokens, a consumer that never reads the original document recovers a designated fact in all 20 trials from a shared subset of only 36 of $32{,}768$ cache positions; matched-bandwidth random and recency controls recover none. A per-head question-aware selector succeeds even at 20 retained positions, and RoPE-correct re-addressing allows selected state to be moved into a new positional coordinate frame without loss of retrieval. These results suggest that the KV cache is not merely stored context, but a manipulable state interface through which pretrained transformers can preserve, compress, address, and communicate information without architectural changes or fine-tuning.

1 INTRODUCTION

Large pretrained transformers have become the standard architecture for language modeling, code generation, and long-context reasoning. Yet even models with context windows of hundreds of thousands of tokens remain constrained by the memory required to process and retain those contexts (Grattafiori et al., 2024). Real workloads can easily exceed these limits: large scientific codebases such as E3SM, FLASH, and NWChem span millions of lines of code (E3SM Project, 2024; Dubey et al., 2020; Valiev et al., 2010), corresponding to tens of millions of tokens. Processing such sequences therefore still requires truncation, repeated re-encoding, or approximations that alter the attention computation. The challenge is particularly acute when exact recall matters, such as recovering an identifier from a long log or preserving a factual detail introduced much earlier in a document.

A broad range of approaches address this problem by limiting, compressing, or reorganizing the information available during inference. Streaming methods maintain bounded memory by retaining a small set of attention sinks together with a window of recent tokens (Xiao et al., 2024; Han et al., 2024). KV-cache compression methods evict, prune, or quantize parts of the cache to reduce its memory footprint (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024; Cai et al., 2024; Ge et al., 2024; Liu et al., 2024; Hooper et al., 2024). Other approaches introduce recurrent or learned memory (Dai et al., 2019; Rae et al., 2020; Wu et al., 2022; Bulatov et al., 2022; Hutchins et al., 2022; Chevalier

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.

et al., 2023; Munkhdalai et al., 2024), modify positional encodings or attention patterns, or replace attention with state-space dynamics (Gu et al., 2022; Gu & Dao, 2024). Chunked prefill takes a complementary systems approach: it divides a long prompt into smaller forward passes while carrying the accumulated key–value (KV) cache from one chunk to the next (Agrawal et al., 2023; 2024). Because that cache is carried unchanged, chunked prefill preserves the computation of a monolithic causal forward pass while reducing peak activation memory.

KV cache as computational state. This observation suggests a broader viewpoint. Rather than treating the KV cache as an internal artifact of attention, we treat it as an explicit *manipulable computational state*. A decoder-only transformer maps a chunk together with the KV state produced by preceding chunks to a new KV state. Repeated application induces a recurrence over the cache. Under this view, standard chunked prefill is the *identity-transformation case*: the state produced at one step is passed unchanged into the next. In exact arithmetic, this recurrence is equivalent to processing the sequence in a single causal forward pass. Once the carried state is exposed, however, passing it unchanged need not be the only choice. The state can instead be quantized, selectively retained, re-addressed, or transferred before it is consumed by a subsequent computation.

We formalize this perspective as *KV-Fold*. Given a sequence divided into chunks, KV-Fold processes the chunks while carrying the KV cache as the accumulator of a fold, analogous to `foldl` in functional programming. At each step, the current chunk is evaluated against the carried state, producing new keys and values and hence an updated computational state. A transformation may then be applied to this state before it is consumed by the next step. When the transformation is the identity, KV-Fold reduces exactly to standard chunked prefill and therefore to the corresponding monolithic causal computation. Nonidentity transformations expose a larger design space in which memory, fidelity, and communication can be traded against one another without modifying the pretrained model itself.

Beyond chunked prefill. This distinction is important. KV-Fold does not introduce a new attention mechanism, and the identity-transformation case by itself does not change the asymptotic size of the KV cache. If the state is passed forward unchanged, the amount of retained state still grows linearly with context length. Instead, KV-Fold makes explicit *where* the persistent state of a pretrained transformer resides and exposes a natural interface for acting on that state between computations. Even the identity-transformation case has a practical benefit. A single monolithic forward pass can be replaced with a sequence of smaller chunk-wise passes, reducing peak working memory without changing the underlying model computation. More importantly, once the carried state is transformed, KV-Fold lets us ask a broader question: how much of the model's persistent state must be preserved for a downstream computation to succeed?

This perspective also connects naturally to recent work on latent multi-agent communication. Zou et al. (2025) show that one transformer pass can attend directly to the KV cache produced by another pass, using it as a prefix and allowing information to flow between inference processes through latent state rather than generated text. We rely on the same underlying capability, but use it for a different purpose. Rather than treating the KV cache primarily as a communication channel between agents, we view it first as the persistent state of a long-context computation and ask what operations can be performed on that state. From this perspective, persistence, compression, selective retention, re-addressing, and communication are different uses of the same state interface.

From exactness to manipulation. We begin with the identity-transformation case because it provides the reference point for everything that follows. We show that carrying an unchanged KV cache across chunks produces, in exact arithmetic, the same hidden states, logits, and KV entries as a single monolithic causal forward pass. We then confirm this equivalence empirically across several pretrained transformer families, with next-token loss matching full attention to within 0.002 nats. Thus chunked prefill is not an approximation in our framework; it is the exact identity reference against which subsequent state transformations can be measured.

We next ask what becomes possible once the KV cache is treated as state that can be acted upon. On needle-in-a-haystack retrieval, the identity KV-Fold preserves 100% exact-match accuracy across context lengths up to 128K tokens, including chain depth 511 on Llama-3.1-8B-Instruct, yielding $152/152$ successful single-needle trials. In the multi-needle setting, it achieves $175/176$ successful per-needle retrievals across the tested configurations. On a 40,GB A100, the 128K-token fold also completes within available memory, whereas a single full-context forward pass does not fit. This illustrates a practical benefit of reorganizing the same computation into a sequence of smaller passes.

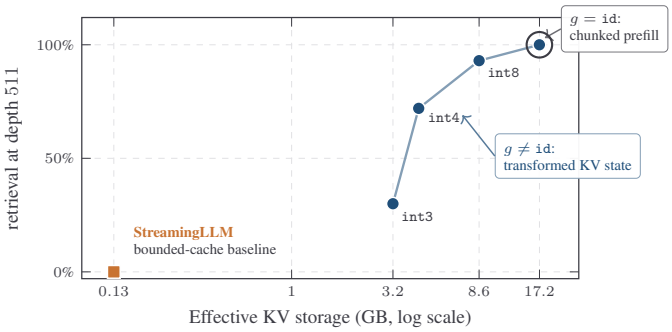

Figure 1: **Retrieval under KV-state transformations at T=128K.** Exact-match retrieval at depth 511 versus effective KV storage for Llama-3.1-8B. The ringed g = `id` point is standard chunked prefill; `int8`, `int4`, and `int3` apply per-step KV quantization. Quantized storage reflects retained precision; round-trip simulation does not reduce allocated GPU memory. StreamingLLM is a bounded-cache reference.

We then move beyond the identity-transformation case and modify the carried state itself. Per-step `int4` quantization preserves roughly 70–73% retrieval at the deepest tested context, while question-aware selection shows that task-relevant information can be concentrated in a very small fraction of cache positions. Finally, we use the cache not only as persistent state but also as a communication medium. At 32K tokens, transmitting only 36 of 32,768 KV positions allows a consumer that never reads the original document to recover the target fact in all 20 trials, while matched-bandwidth random and recency-based controls recover none. The selected state can also be re-addressed into a new positional coordinate frame without loss of retrieval.

Taken together, these experiments move from exact state preservation to progressively stronger interventions on the cache. They expose a memory–fidelity–communication trade-off that is difficult to see when the KV cache is treated only as an implementation detail. Figure 1 summarizes this design space, with chunked prefill providing the exact identity reference point and cache manipulation opening a broader state-transformation frontier.

Contributions. Our contributions are threefold. First, we formalize the KV cache as an explicit computational state and show that standard chunked prefill is the identity-transformation case of KV-Fold: carrying the cache forward unchanged is exactly equivalent, in exact arithmetic, to a monolithic causal forward pass. We validate this equivalence across several pretrained transformer families and identify a subtle evaluation pitfall caused by in-place mutation of a live KV cache during scoring. Second, using this exact identity case as a controlled reference, we study training-free transformations of the carried state, including per-step quantization and selective retention, and characterize their effects on long-range retrieval. Third, we show that the KV cache can act as a communication medium: a small, carefully selected subset of KV positions can transmit task-relevant information to a consumer that never processed the original document, and RoPE-correct re-addressing allows this state to be reused in a new positional coordinate frame.

2 KV-Fold: KV State as a Recurrent Interface

We adopt the chunked-attention setup standard in the long-context literature. A long sequence of length T is divided into chunks of length C, yielding $N = T/C$ chunks indexed $0, 1, \ldots, N-1$. Each chunk is processed as a single forward pass through the model. At chunk t, we process $\mathbf{x}_t = [x_{tC}, x_{tC+1}, \ldots, x_{(t+1)C-1}]$, where boldface $\mathbf{x}_t$ denotes the length-C chunk and plain x_i denotes the token at absolute position i. The computation uses the KV state accumulated from preceding chunks as a prefix.

Concretely, for each transformer layer ℓ, processing chunk t produces queries $Q_t^{(\ell)}$ from the new tokens and appends new keys and values $K_t^{(\ell)}$ and $V_t^{(\ell)}$ to the carried state. With the state left unchanged, the resulting keys are $K_{0:t}^{(\ell)} = [K_0^{(\ell)}; K_1^{(\ell)}; \ldots; K_t^{(\ell)}]$, with analogous values $V_{0:t}^{(\ell)}$. Position identifiers for the new tokens continue from absolute position tC, so the RoPE (Su et al., 2024) rotations agree with those of the original sequence. This is the layer-wise working-memory

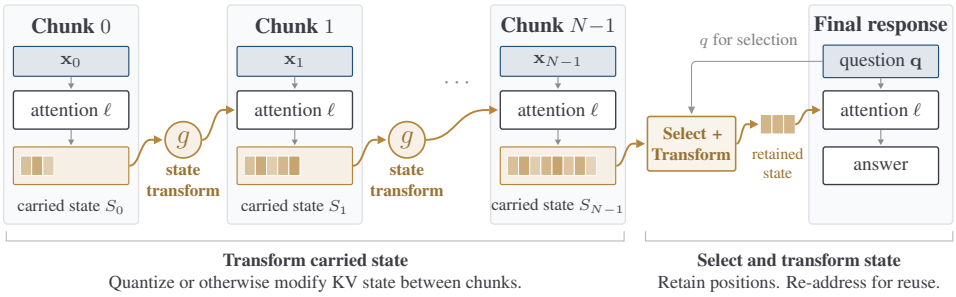

Figure 2: **KV-Fold with transformations of the carried state.** Each chunk attends to accumulated KV state, produces new entries, and passes the resulting state through a transformation g before the next chunk consumes it. The identity choice $g = \texttt{id}$ leaves the state unchanged and recovers standard chunked prefill. Nonidentity transformations can modify the carried state, including the per-step quantization studied here. After the document has been processed, question-aware selection can retain a small subset of KV positions, and RoPE-correct re-addressing can move the selected state into a new positional frame for the final answering pass. Per-step quantization and selective transfer are evaluated separately; quantization is simulated and does not reduce allocated memory.

transfer of LatentMAS (Zou et al., 2025, Section 3.2), applied across consecutive chunks rather than across agents. KV-Fold additionally exposes the carried state between successive forward passes. We denote by g a *state transformation* applied at this boundary, before the next chunk consumes the state. The identity transformation $g = \texttt{id}$ leaves the cache unchanged and recovers standard chunked prefill. Nonidentity choices of g can instead quantize, selectively retain, re-address, or otherwise transform the carried state. Figure 2 illustrates this interface. Within a single sequence, $g = \texttt{id}$ coincides with chunked prefill as deployed in serving systems (Section A); Section 3 verifies its equivalence to full attention empirically, while Section 5 studies nonidentity transformations and cross-context transfer, where this equivalence no longer applies.

A recurrent view. Let $S_t = (\mathcal{K}^{(t)}, \mathcal{V}^{(t)})$ denote the accumulated KV state after processing chunk t, with $S_{-1} = \emptyset$. One KV-Fold step is

$$S_t = \mathcal{F}_\theta\big(g(S_{t-1}), \mathbf{x}_t\big),$$

where $\mathcal{F}_\theta$ is the standard transformer forward pass. Defining the step operator $\mathcal{H}_g(S, \mathbf{x}) := \mathcal{F}_\theta\big(g(S), \mathbf{x}\big)$, the complete computation is the left fold

$$S_{N-1} = \texttt{foldl}\big(\mathcal{H}_g,\ \emptyset,\ [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1}]\big).$$

The KV state is therefore the accumulator, while g specifies what, if anything, happens to that state between successive fold steps. For $g = \texttt{id}$, $\mathcal{H}_g = \mathcal{F}_\theta$ and the fold is standard chunked prefill. For $g \neq \texttt{id}$, the state consumed by the next chunk differs from the state produced by the preceding forward pass. Unlike classical recurrent models, the KV state need not be compressed into a fixed-dimensional vector: under the identity transformation it grows with the sequence and retains structured representations of past tokens, allowing continued content-based addressing through attention. Nonidentity transformations can alter this state-size and information-retention behavior, which we study in Section 5.

Three reference conditions. For every experiment we measure the next-token negative log-likelihood (NLL) per chunk under FULL, a single full-attention forward over the entire T-token sequence (the ceiling); ISOLATED, each chunk processed with no prefix (the floor); and KV-FOLD, the identity-transformation case $g = \texttt{id}$, in which each chunk is processed with the accumulated KV-cache prefix. We define the per-depth drift as $\text{drift}(d) = \text{NLL}_{\text{KV-FOLD}}(d) - \text{NLL}_{\text{FULL}}(d)$ and the per-depth recurrence advantage as $\text{advantage}(d) = \text{NLL}_{\text{ISOLATED}}(d) - \text{NLL}_{\text{KV-FOLD}}(d)$, where d is the chain depth (chunk index-1). We use the validation split of PG-19 (Rae et al., 2020), a long-form prose benchmark, with a fixed seed. Each window starts 200 tokens into a document and spans the next T tokens. All NLL values are computed in `bfloat16` unless stated otherwise.

3 EXACTNESS OF THE IDENTITY FOLD

We first study the identity-transformation case $g = \texttt{id}$. For a single sequence with continuous positions and an unmodified cache, KV-Fold computes the same function as the monolithic forward it replaces. At every layer, the queries of chunk t attend to exactly the keys and values that the full forward computes for positions $[0, (t{+}1)C)$, under the same causal structure and RoPE rotations. Thus the two computations can differ only through floating-point operation order. This is the same equivalence underlying chunked-prefill serving systems (Section A); here we state it explicitly because it provides the reference point against which nonidentity transformations are measured.

Proposition 1 (exactness under the identity transformation)**.** *Let $g = \texttt{id}$. In exact arithmetic, for any chunk size C dividing T, KV-Fold and the monolithic forward over the same T tokens produce identical logits at every position and an identical final cache $(\mathcal{K}^{(N-1)}, \mathcal{V}^{(N-1)})$.*

Proof. By induction on the chunk index t, with a nested induction over layers within each step. For $t = 0$, the two computations are the same forward pass. Assume that after chunk $t{-}1$ the carried cache equals, at every layer, the keys and values assigned by the monolithic forward to positions $[0, tC)$. Since $g = \texttt{id}$, this cache is passed unchanged to chunk t. Processing chunk t, the layer-ℓ inputs at positions $[tC, (t{+}1)C)$ depend only on earlier layers at those same positions, by the nested induction, and through attention on the cached keys and values for $[0, tC)$ together with those produced for the current chunk. These agree with the monolithic computation by the induction hypothesis, while the causal mask and RoPE rotations agree by construction. Hence the layer outputs, appended cache entries, and logits coincide. □

Proposition 1 is the single-sequence analogue of Theorem 3.3 of Zou et al. (2025), which establishes the corresponding equivalence for KV working memory passed between agents. The proposition also identifies precisely where the equivalence ends: for $g \neq \texttt{id}$, the next chunk no longer receives the state produced by the corresponding monolithic forward, so fidelity becomes an empirical property rather than a consequence of exactness.

In floating-point arithmetic, different evaluation schedules can change operation order, so the empirical question for $g = \texttt{id}$ is the size of the resulting numerical residual.

Matched-pipeline measurement. We measure per-token NLL under FULL and identity KV-FOLD through an identical pipeline: the same SDPA attention backend, decoder-stack invocation, fp32 chunked cross-entropy, and PG-19 windows. Table 3 (see Appendix) reports drift across eight (model, T) cells spanning three architectures and chain depths 7 to 255. Every cell lies within ± 0.002 nats, with fluctuating sign. A per-depth verification at the operating point of the previously reported measurement gives the same conclusion at individual depths: the maximum per-depth |drift| is 0.0051 nats and the deep-band mean is -0.0008 on OLMoE at depths 1 to 15 (Appendix D). Thus the identity fold agrees with full attention to within kernel numerics.

Logit-level verification. The same conclusion holds at the level of raw logits. Under a deterministic math-mode kernel with TF32 disabled (Qwen2.5-1.5B, T=4,096, three windows), the identity fold and monolithic forward agree in fp32 to a maximum absolute logit difference of 5.9×10^{-4}, with greedy decisions identical at all 12,288 measured positions. In bf16, the residual lies at that precision's accumulation floor (median per-position maximum 0.20, greedy agreement 98.0%) and is flat in chain depth (late-to-early band ratio 1.04). Increasing precision by $10{,}000\times$ therefore reduces the residual by three orders of magnitude, while the same change moved the previously observed spurious plateau by only 2.8%. The remaining discrepancy is numerical rather than a property of the fold.

Cross-chunk context is preserved. The ISOLATED condition trails FULL by roughly 0.3–0.5 nats per token on PG-19 across the measured depths, quantifying the contribution of cross-chunk context. Since identity KV-FOLD matches FULL to within 0.002 nats, it closes essentially the entire isolated-to-full NLL gap over these experiments. For example, on Llama-3.1-8B at 32K, continuation NLL is 2.32 under the fold compared with 2.80 under isolated processing.

A measurement pitfall that manufactures drift. A previously reported measurement of the identity recurrence exhibited a drift plateau of 0.040 to 0.122 nats, depending on architecture. The apparent drift rose over the first several depths, then remained approximately flat; it also survived a $10{,}000\times$ precision change and remained stable across an $8\times$ chunk-size sweep. Forensic analysis (Appendix D) attributes this plateau to an evaluation bug rather than to the recurrence. The Hug-

Table 1: Single-needle retrieval, memory, and compute on Llama-3.1-8B-Instruct (A100 40GB) for $g = \texttt{id}$. Retrieval is 100% at all tested distances ($152/152$ trials overall). Memory and timing use the first trial at each T. KV cache grows linearly at 131 KB/token, matching $32\times8\times128\times2\times2 = 131{,}072$ bytes/token in `bf16`.

T	depth	trials/d	retrieval	KV cache	peak GPU	per-chunk	total chain
32K	127	10	100%	4.29 GB	21.00 GB	0.103 s	13.2 s
64K	255	5	100%	8.59 GB	25.86 GB	0.176 s	44.9 s
96K	383	3	100%	12.88 GB	30.72 GB	0.252 s	96.9 s
128K	511	3	100%	17.18 GB	35.57 GB	0.335 s	171.3 s

gingFace `transformers` implementation appends to a passed cache regardless of `use_cache`, so a scoring forward given the live chain cache mutates that cache before the chain appends the same chunk again. The duplicated fraction $(c-1)/(2c-1)$ approaches $1/2$, reproducing the observed rise-then-plateau shape, while its invariance to precision and chunk size explains why those robustness checks did not expose the error. Scoring against a cloned cache removes the mutation and changes the measured drift at the original operating point from $+0.118$ to -0.001. We document this failure mode because any cache-passing evaluation harness that performs a separate scoring forward on a live cache is potentially exposed to it.

4 Long-Context Retrieval and Scaling

Long-range retrieval. NLL is a marginal-distribution measure that averages over many kinds of predictions. We therefore also evaluate the identity-transformation case $g = \texttt{id}$ at the task level using a needle-in-a-haystack retrieval benchmark (Kamradt, 2023), a standard probe of long-context retrieval used by long-context evaluation suites (Hsieh et al., 2024; Bai et al., 2024). Each trial inserts "*The magic number for [key] is [value].*" at a controlled chunk position of a PG-19 window, asks for the value after the final chunk, and scores exact match (Appendix F).

On Qwen2.5-7B-Instruct, with twenty trials at each of four needle-to-question distances up to 62 chunk transitions (Appendix Table 12), identity KV-Fold achieves 100% exact-match retrieval at every tested distance, matching full attention exactly, while isolated chunks remain at 0%. Thus passing the KV state unchanged preserves specific factual content across deep chains: retrieving a previously unseen five-digit value from chunk 1 after 62 transitions succeeds in all 20 trials.

4.1 Scaling to long contexts

We next ask whether the identity-transformation case $g = \texttt{id}$ continues to preserve retrieval at larger contexts, including operating points where a single monolithic forward becomes infeasible on the same hardware. We use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) because it has a 128K native context window, so all tested values of T remain within the model's trained position range and avoid confounding chain depth with out-of-distribution RoPE behavior. We run the needle-in-a-haystack protocol of Section 4 at four operating points: $T = 32$K (chain depth 127, 10 trials per distance), $T = 64$K (depth 255, 5 trials), $T = 96$K (depth 383, 3 trials), and $T = 128$K (depth 511, 3 trials). At each operating point we test four needle-to-question distances: 1, 31, approximately the chain midpoint, and the maximum chain depth.

Retrieval, memory, and compute. Table 1 reports retrieval accuracy, KV-cache size, peak GPU memory, and wall-clock at each operating point. Across all tested distances, spanning chain depths 1 through 511, every trial recovers the gold needle exactly, for $152/152$ single-needle trials in total (Wilson 95% lower bound 97.5%). The 128K cells contain only three trials each, so any individual cell alone provides a much weaker lower confidence bound (44%). Peak memory for identity KV-Fold at 128K is 35.57 GB, leaving approximately 4.4 GB of headroom on a 40 GB A100.

The same behavior extends to multiple simultaneously stored facts. With $K \in \{4, 8\}$ independent needles at $T = 128$K, and $K \in \{2, 4, 8\}$ on Qwen2.5-7B at 16K, identity KV-Fold recovers $175/176$ needles overall, with no clear degradation from increasing the number of needles (Appendix F).

Identity KV-Fold does *not* reduce the size of the persistent KV state relative to full attention: after processing T tokens, it stores the same cache that the corresponding monolithic forward would

produce. Its operational advantage is instead lower *working* memory during prefill, because each forward processes only one chunk at a time.

A single full-attention forward at $T = 128$K does not fit on a 40 GB A100. The binding constraint is not an explicitly materialized attention-score matrix: modern kernels, including FlashAttention-2 and the FlashAttention-backed PyTorch SDPA path (Dao, 2024), avoid storing that matrix. Instead, FFN intermediate activations become the limiting allocation. At $T = 128$K, the gate and up projections each require a $[T, 14{,}336]$ buffer of approximately 3.5 GiB, in addition to the model weights and 17.18 GB KV cache.

We measure this feasibility boundary directly. Single-shot SDPA prefill completes at $T \in \{32\text{K}, 64\text{K}, 96\text{K}\}$ with peak memory 24.3, 32.5, and 40.7 GB and wall-clock 4.2, 9.9, and 19.3 s, respectively, but runs out of memory at $T = 128$K while attempting an additional 3.5 GiB allocation. Identity KV-Fold completes the same 128K context at 35.6 GB peak (Appendix B). Thus 128K is the first tested operating point at which the chunked execution is required on this hardware; below that point, single-shot prefill is 3–5$\times$ faster, while KV-Fold uses less peak working memory. Per-token decode latency is independent of the prefill route once the same final cache has been produced (Appendix B.1).

At scale, the identity transformation therefore changes the execution schedule rather than the underlying computation: it replaces one infeasible monolithic forward with a sequence of smaller forwards while preserving the same KV state and retrieval behavior. Going beyond this operating point without increasing available memory requires reducing or otherwise modifying the carried state. Section 5 studies precisely this regime, where $g \neq \texttt{id}$ and fidelity becomes an empirical trade-off.

Comparison to StreamingLLM. StreamingLLM (Xiao et al., 2024) provides a bounded-memory reference point, retaining a constant 1024-token cache consisting of 4 attention sinks and a 1020-token recent window; LM-Infinite uses a related strategy (Han et al., 2024). At T=128K on Llama-3.1-8B-Instruct, StreamingLLM runs 7.5$\times$ faster than identity KV-Fold ($g = \texttt{id}$) and uses 16.6 versus 35.6 GB peak GPU memory. It matches KV-Fold at d=1, but retrieves $0/3$ at each of $d \in \{31, 255, 511\}$, whereas KV-Fold retrieves $3/3$, $3/3$, and $2/3$, respectively. KV-Fold also achieves lower next-token NLL in this experiment, 2.46 versus 2.66 nats. These measurements come from a separate stress test at extreme context depth and are distinct from the main retrieval sweep of Sections 4–4.1. The bounded 0.13 GB cache of StreamingLLM discards tokens once they leave its retained window, so needles at sufficiently large distances are no longer available for direct attention (Appendix G). The comparison illustrates different memory–retrieval operating points. StreamingLLM keeps KV memory bounded and substantially reduces runtime and peak memory, at the cost of losing direct access to content outside its retained window. Identity KV-Fold retains the complete KV state, requiring memory that grows linearly with context length but preserving access to content at much larger tested distances.

5 Transforming and Transferring KV State

We now turn to nonidentity state transformations, $g \neq \texttt{id}$. The equivalence of Section 3 relies on passing the carried KV state between chunks unchanged. Once g modifies that state, the next chunk no longer receives the cache that the corresponding monolithic forward would have produced, and fidelity becomes an empirical question. Likewise, when producer and consumer are different inference streams, there is no single monolithic forward that provides an equivalent reference computation.

We study several transformations. Following an analogy with LSTM gates, we separate changes to carried-state precision, selective preservation of positions, and decisions about what information can be retained before the eventual query is known. The analogy is organizational rather than architectural: nothing is learned, and the underlying transformer is unchanged. We first characterize these transformations at the deepest operating point of Section 4.1 (T=128K, d=511, Llama-3.1-8B), and then consider cross-context transfer. Full protocols, tables, and statistics appear in Appendix C.

5.1 Transforming the carried state

Precision gate: per-step quantization. We first let g quantize each chunk's keys and values to `intn` immediately after its forward pass, so that subsequent chunks attend to rounded state. Per-step `int8`

retains $13/14 = 93\%$ retrieval at d=511, while `int4` retains 70–73% ($7/10$ at the deepest band, Wilson $[40\%, 89\%]$, consistent with $24/33$ across the distance grid). `int3` falls to $3/10 = 30\%$, and `int2` fails at every tested scale. The `int3` failures are approximately constant across distance rather than increasing with chain depth, suggesting that the dominant effect is corruption of the needle's local representation rather than accumulation of error across hundreds of transitions. Information-removing transformations behave substantially worse: uniform decay and attention-based pruning fail to preserve retrieval at meaningful depth. These results suggest that the carried state is more tolerant of distributed quantization noise than of systematic removal of positions.

Output gate: selective preservation. Question-aware salience, computed as the mean attention from question tokens to each cache position, localizes needle content within the top 0.01% of $131{,}072$ positions in all 10 trials. Preserving the most salient 5–10% of positions at full precision while quantizing the remainder to `int3` raises retrieval from $3/10$ to $5/10$ and $6/10$, respectively. At this sample size the effect is directional rather than conclusive (Appendix H). Figure 1 places these measured operating points between the bounded-cache StreamingLLM reference and the identity point $g =$ `id` corresponding to chunked prefill.

Input gate: the limitation of query-blind selection. The eventual question is unavailable while the document is being processed. In eviction experiments at T=4K, salience computed from the current chunk's queries performs poorly at retaining content needed by a later question: retrieval at d=7 remains $1/5$ across a $5\times$ sweep of recency-debiasing strength (five trials per setting; Wilson interval 4–62%). This provides an empirical contrast with the strong localization obtained once the eventual question is available. In our experiments, chunk-time attention is therefore insufficient to identify which apparently unremarkable content will later become task relevant; question-aware selection avoids this limitation but requires the query to arrive after the context.

5.2 THE CACHE AS A MESSAGE: CROSS-CONTEXT TRANSFER

The experiments above retain a single producer-consumer stream: the same inference process writes and later reads the carried state. We next consider a different regime in which KV state produced under one context is consumed by another computation. This is the more general capability demonstrated by LatentMAS (Zou et al., 2025); here we ask quantitatively how much of the producer's state must be transmitted for the consumer to recover a designated fact.

A producer processes a T=32,768-token document containing a needle at chain depth $\sim$126. A consumer receives a subset of the producer's KV positions, together with the question, and never sees the document tokens (Llama-3.1-8B-Instruct, N=20 trials). The transmitted entries retain their keys, values, and RoPE rotations, with attention-sink positions always included. We compare question-aware salience with SnapKV-style per-head selection (Li et al., 2024), and use matched-bandwidth random and recency controls. A re-indexed condition tests whether the selected state can be moved out of the producer's original positional coordinate frame. Table 2 reports the results.

Question-aware transfer remains perfect down to 36 positions, 0.11% of the full cache, corresponding to a $\sim 910\times$ reduction in transmitted positions; at 20 positions it succeeds in $19/20$ trials. At this smallest shared-subset setting, the selected state contains a median of only 5 of the needle's $\sim$13 tokens, indicating that performance begins to degrade as the transmitted set approaches the value-bearing span itself. Matched-bandwidth random and recency selection both obtain $0/20$. Unlike low-bit quantization, which perturbs every retained representation, position selection removes entries while leaving the surviving keys and values unchanged.

The effect is not specific to our shared-position selector. SnapKV-style selection, using per-layer, per-head max-pooled top-k question attention (Li et al., 2024), achieves $20/20$ transfer down to a 20-position budget. Its retained positions vary across heads: the median of the 256 (layer, head) retained sets contains no needle tokens even though transfer remains perfect. This is consistent with relevant information being concentrated in a subset of retrieval heads rather than across heads.

The selected state is also not tied to the producer's original positional coordinates. Re-indexing the 167 retained positions to $0, \ldots, 166$, re-rotating each stored key by the corresponding RoPE offset, $k' = k\cos\Delta + \mathrm{rotate_half}(k)\sin\Delta$, and placing the question at position 167 preserves $20/20$ retrieval. Thus selected KV state can be re-addressed into a new position budget before being consumed. Selection is not purely lexical: transfer remains $10/10$ when the needle and question share

Table 2: Cache-as-message retrieval on Llama-3.1-8B-Instruct (T=32,768, depth $\sim$126, N=20). The producer builds the full cache; the consumer receives listed KV positions plus the question and never sees the document. "Coverage" is the median number of the needle's $\sim$13 tokens transmitted. For N=20, Wilson 95% bounds are $[84\%, 100\%]$ for $20/20$ and $[0\%, 16\%]$ for $0/20$. An earlier N=10 run agrees on all overlapping conditions. †SnapKV retains different positions per (layer, head); the median across 256 retained sets contains zero needle tokens despite successful transfer.

Transmitted positions	count	transfer	coverage
full cache	32,768	$20/20$	12
question-aware, f=0.005	167	$20/20$	12
question-aware, f=0.002	69	$20/20$	11
question-aware, f=0.001	36	$20/20$	9
question-aware, f=0.0005	20	$19/20$	5
SnapKV per-head, f=0.005	167	$20/20$	–†
SnapKV per-head, f=0.0005	20	$20/20$	–†
compacted question-aware, f=0.005	167	$20/20$	12
random, f=0.005	167	$0/20$	0
recency, f=0.005	167	$0/20$	0

no content word. At the same time, the selected message preserves only about half of the document's general language-modeling signal (Appendix H), indicating that the sparse state is task specific rather than a general replacement for the full cache.

Three qualifications delimit this result. First, selection is conditioned on the consumer's question, corresponding to a receiver-driven setting in which the request is known before transmission; the query-blind experiments above show that our chain-time selectors do not reach this operating point. Second, the question is evaluated against the full producer cache to compute salience, so these experiments measure the bandwidth required to *transfer* task-relevant state once it has been identified, not the cost of discovering that state without access to the full cache. We have not yet compared this latent message with transmitting the producer's decoded answer, nor tested a consumer query different from the query used for selection. Third, cross-architecture replication remains unresolved: in our current Qwen2.5-7B consumer harness the full-cache condition itself fails in half the trials, and the sparse channel tracks that ceiling. We therefore report the cross-context result for Llama-3.1-8B and describe the Qwen replication status in Appendix E.

6 Conclusion and Outlook

KV-Fold treats a pretrained transformer's KV cache as explicit computational state. In the identity case $g = \texttt{id}$, carrying that state across chunks reproduces monolithic full attention to measurement precision while allowing long contexts to be processed as a sequence of smaller forward passes. Once exposed, the state admits a broader design space: nonidentity transformations tolerate substantial quantization, selected positions can preserve task-relevant information at very low bandwidth, and RoPE-correct re-addressing allows retained state to be consumed in a new positional frame.

Three directions follow. First, the two-tier design of Appendix C.9, using cumulative attention exposure as a promotion signal, offers a route toward constant-memory execution. Second, combining KV-Fold with position-extension methods such as YaRN (Peng et al., 2024), Positional Interpolation (Chen et al., 2023), or LongRoPE (Ding et al., 2024) could extend it beyond the 128K operating point studied here. Third, question-aware transfer with only 36 of 32,768 shared cache positions makes latent communication between forward passes quantitatively measurable and motivates different consumer prompts, mixture-of-experts routing regimes, and multi-agent settings (Zou et al., 2025). More broadly, KV-state recurrence may compose with recurrence along other computational axes: recirculation (Mozer et al., 2026b), for example, adds depth-wise state to the same frozen models for which KV-Fold exposes sequence-wise state. Taken together, these results suggest that the KV cache is useful not only as memory for attention, but as a programmable interface for carrying, transforming, and communicating latent state across computations.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and editing text, writing LaTeX and figure code, developing and debugging experiment notebooks, and checking citations against their sources. All experiments were conducted by the authors, and all AI-assisted text, code, figures, and citations were reviewed and verified by the authors. The authors take full responsibility for the content of this work, including all claims, results, code, and artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

Proposition 1 is proved in Section 3. The evaluation protocols, including the corrected chained-evaluation harness and its cache-length assertion, are described in Appendices D, E, and F. We will release the evaluation harness and experiment notebooks.

REFERENCES

A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve. In *OSDI*, 2024.

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills, 2023. URL https://arxiv.org/abs/2308.16369.

Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, Y. Dong, J. Tang, and J. Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In *ACL*, 2024.

L. Brito and C. Baquero. When Latent Agents Lie: KV-Cache Integrity in Multi-Agent LLM Collaboration. *arXiv preprint arXiv:2606.28958*, 2026.

A. Bulatov, Y. Kuratov, and M. S. Burtsev. Recurrent Memory Transformer. In *NeurIPS*, 2022.

Z. Cai, Y. Zhang, B. Gao, Y. Liu, T. Liu, K. Lu, W. Xiong, Y. Dong, B. Chang, J. Hu, and W. Xiao. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *arXiv preprint arXiv:2406.02069*, 2024.

S. Chen, S. Wong, L. Chen, and Y. Tian. Extending Context Window of Large Language Models via Positional Interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

A. Chevalier, A. Wettig, A. Ajith, and D. Chen. Adapting Language Models to Compress Contexts. *arXiv preprint arXiv:2305.14788*, 2023.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *ACL*, 2019.

T. Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *ICLR*, 2024.

Y. Ding, L. L. Zhang, C. Zhang, Y. Xu, N. Shang, J. Xu, F. Yang, and M. Yang. LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens. In *ICML*, 2024.

Anshu Dubey, Jared O'Neal, Klaus Weide, and Saurabh Chawdhary. Distillation of best practices from refactoring flash for exascale. *SN Computer Science*, 1(4):223, 2020. doi: 10.1007/s42979-020-0077-x.

E3SM Project. Energy exascale earth system model (e3sm). Computer Software, March 2024. URL https://doi.org/10.11578/E3SM/dc.20240301.3.

T. Fu, Z. Min, H. Zhang, J. Yan, G. Dai, W. Ouyang, and Y. Wang. Cache-to-Cache: Direct Semantic Communication Between Large Language Models. In *ICLR*, 2026.

S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs. In *ICLR*, 2024.

A. Grattafiori, A. Dubey, A. Jauhri, and others. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

A. Gu and T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *COLM*, 2024.

A. Gu, K. Goel, and C. Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In *ICLR*, 2022.

C. Han, Q. Wang, H. Peng, W. Xiong, Y. Chen, H. Ji, and S. Wang. LM-Infinite: Zero-Shot Extreme Length Generalization for Large Language Models. In *NAACL*, 2024.

C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, Y. S. Shao, K. Keutzer, and A. Gholami. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. In *NeurIPS*, 2024.

C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, Y. Zhang, and B. Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? In *COLM*, 2024.

D. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur. Block-Recurrent Transformers. In *NeurIPS*, 2022.

G. Kamradt. Needle In A Haystack: Pressure Testing LLMs. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`, 2023.

W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *SOSP*, 2023.

Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. SnapKV: LLM Knows What You are Looking for Before Generation. *arXiv preprint arXiv:2404.14469*, 2024.

Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava. Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time. In *NeurIPS*, 2023.

Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache. In *ICML*, 2024.

M. C. Mozer, S. A. Siddiqui, and R. Liu. The Topological Trouble with Transformers. *arXiv preprint arXiv:2604.17121*, 2026a.

M. C. Mozer, S. A. Siddiqui, D. Sawyer, S. Sanyal, and R. Liu. Recirculation. *arXiv preprint arXiv:2608.17981*, 2026b.

N. Muennighoff, L. Soldaini, D. Groeneveld, K. Lo, J. Morrison, S. Min, W. Shi, P. Walsh, O. Tafjord, N. Lambert, Y. Gu, S. Arora, A. Bhagia, D. Schwenk, D. Wadden, A. Wettig, B. Hui, T. Dettmers, D. Kiela, A. Farhadi, N. A. Smith, P. W. Koh, A. Singh, and H. Hajishirzi. OLMoE: Open Mixture-of-Experts Language Models. In *ICLR*, 2025.

T. Munkhdalai, M. Faruqui, and S. Gopal. Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

B. Peng, J. Quesnelle, H. Fan, and E. Shippole. YaRN: Efficient Context Window Extension of Large Language Models. In *ICLR*, 2024.

J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive Transformers for Long-Range Sequence Modelling. In *ICLR*, 2020.

J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing, 568:127063*, 2024.

Marat Valiev, Eric J. Bylaska, Niri Govind, Karol Kowalski, Tjerk P. Straatsma, Hubertus J. J. van Dam, Donghai Wang, Jarek Nieplocha, Edoardo Aprà, Theresa L. Windus, and Wibe A. de Jong. Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications*, 181(9):1477–1489, 2010. doi: 10.1016/j.cpc.2010.04.018.

Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy. Memorizing Transformers. In *ICLR*, 2022.

G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient Streaming Language Models with Attention Sinks. In *ICLR*, 2024.

A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, and others. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024a.

Y. Yang, Z. Cao, Q. Chen, L. Qin, D. Yang, H. Zhao, and Z. Chen. KVSharer: Efficient Inference via Layer-Wise Dissimilar KV Cache Sharing. *arXiv preprint arXiv:2410.18517*, 2024b.

Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett, Z. Wang, and B. Chen. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *NeurIPS*, 2023.

J. Zou, X. Yang, R. Qiu, G. Li, K. Tieu, P. Lu, K. Shen, H. Tong, Y. Choi, J. He, J. Zou, M. Wang, and L. Yang. Latent Collaboration in Multi-Agent Systems. *arXiv preprint arXiv:2511.20639*, 2025.

Table 3: Matched-pipeline drift ($\text{NLL}_{\text{KV-FOLD}} - \text{NLL}_{\text{FULL}}$) on PG-19 for the identity transformation $g = \texttt{id}$. FULL is one forward over T tokens and KV-FOLD is the chunked fold at C=256; both use the same attention backend, decoder invocation, and fp32 chunked cross-entropy. Maximum chain depth is $T/C - 1$. No single-shot reference exists at Llama T=128K, where the single forward runs out of memory (Section 4.1).

Model	T	max depth	drift (nats)
OLMoE-1B-7B (Muennighoff et al., 2025)	2K	7	+0.0017
OLMoE-1B-7B	4K	15	+0.0018
Qwen2.5-7B-Instruct (Yang et al., 2024a)	8K	31	−0.0005
Qwen2.5-7B-Instruct	16K	63	+0.0001
Qwen2.5-7B-Instruct	32K	127	+0.0007
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	16K	63	+0.0011
Llama-3.1-8B-Instruct	32K	127	+0.0002
Llama-3.1-8B-Instruct	64K	255	−0.0000

APPENDIX OVERVIEW

Appendix B supports the scaling results of Section 4.1, detailing the measured feasibility frontier between single-shot SDPA and identity KV-Fold on a 40 GB A100, along with decode rates, amortization thresholds, and effective throughput in Appendix B.1. Appendix C supports Section 5 with protocols and full results for per-step quantization, question-aware salience, the empirical query-blind selection floor, selective preservation, and the proposed two-tier constant-memory design of Appendix C.9. Appendix D documents the evaluation pitfall that produced a spurious drift plateau in an earlier measurement, its forensic attribution, and the corrected protocol underlying Section 3. Appendix E details the cross-context transfer experiments of Section 5, including selector comparisons, position compaction, paraphrase and continuation-NLL evaluations, and the status of cross-architecture replication.

A RELATED WORK

Serving systems chunk long prefills to bound working memory (Kwon et al., 2023; Agrawal et al., 2024). Within a single sequence, this is exactly the identity case of KV-Fold (Proposition 1). KV-cache compression methods evict, prune, share, or quantize cached state (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024; Cai et al., 2024; Ge et al., 2024; Liu et al., 2024; Hooper et al., 2024; Yang et al., 2024b), while streaming methods retain attention sinks and a recent window (Xiao et al., 2024; Han et al., 2024). Our experiments instead apply transformations to carried KV state across repeated chunk transitions and evaluate their effect on exact long-range retrieval.

Recurrent and learned-memory transformers introduce or train a carried state (Dai et al., 2019; Bulatov et al., 2022; Chevalier et al., 2023; Munkhdalai et al., 2024), whereas KV-Fold uses the pretrained model's own KV cache without architectural changes or additional training. In the taxonomy of Mozer et al. (2026a), the identity fold does not introduce a new sequential dependency beyond the causal computation it reorganizes; nonidentity transformations do. Recirculation (Mozer et al., 2026b) instead introduces training-free recurrence along depth, complementary to the sequence-wise state studied here.

LatentMAS (Zou et al., 2025) passes KV working memory between agents, Cache-to-Cache (Fu et al., 2026) learns a projection for transferring cache state between models, and full-cache latent collaboration transmits substantially larger latent states (Brito & Baquero, 2026). Our cross-context experiments use no learned transfer mechanism and instead measure how little selected KV state is sufficient for a consumer to recover task-relevant information.

B THE OPERATIONAL FRONTIER: SINGLE-SHOT SDPA VS KV-FOLD

Section 4.1 reports the measured feasibility frontier between single-shot SDPA and identity KV-Fold on a 40 GB A100. Here we provide the corresponding experimental details and identify the allocation that limits single-shot prefill at the largest tested context.

Table 4: Single-shot SDPA versus identity KV-Fold at matched T on Llama-3.1-8B-Instruct using a 40 GB A100. Single-shot peak memory, wall-clock, and NLL are measured here; KV-Fold peak memory and wall-clock are from the main scaling experiment. Single-shot wall-clock measures the transformer forward; KV-Fold wall-clock is cumulative across chunks. NLL is averaged over three PG-19 windows.

T	fold peak	fold wall	single peak	single wall	single NLL	status
32K	21.00 GB	13.2 s	24.27 GB	4.2 s	2.199	OK
64K	25.86 GB	44.9 s	32.48 GB	9.9 s	2.306	OK
96K	30.72 GB	96.9 s	40.68 GB	19.3 s	2.339	OK
128K	35.57 GB	171.3 s	–	–	–	OOM

For each $T \in \{32{,}768,\ 65{,}536,\ 98{,}304,\ 131{,}072\}$, we run a single SDPA forward over a PG-19 window on Llama-3.1-8B-Instruct with `use_cache=True` and the same SDPA backend used by KV-Fold. We report peak GPU memory, wall-clock for the transformer forward, and per-token NLL on the same window. NLL is computed with a chunked LM head, producing logits for $2{,}048$ positions at a time, to avoid materializing the full $T\times$-vocabulary logits tensor.

Results. Table 4 reports the measurements. Single-shot SDPA completes at $T \in \{32\text{K}, 64\text{K}, 96\text{K}\}$, with increasing peak memory and wall-clock. At $T = 128$K, the single-shot forward runs out of memory while attempting to allocate an additional 3.50 GB tensor with approximately 38 GB already allocated. The limiting allocation is an FFN intermediate: at $T = 128$K, the gate and up projections have shape $[T, 14{,}336]$, requiring approximately 3.5 GiB each in bf16, on top of model weights and the in-progress KV cache.

Interpretation. Three observations follow.

First, 128K is the first tested operating point at which single-shot SDPA fails on this hardware; the exact feasibility boundary lies between the tested 96K and 128K points. The failure is caused by FFN intermediate activations rather than a materialized attention-score matrix.

Second, where both paths fit, identity KV-Fold uses less peak working memory. At $T \in \{32\text{K}, 64\text{K}, 96\text{K}\}$, single-shot prefill uses approximately 3–10 GB more peak memory at matched T. The difference arises because FFN intermediates scale with the full sequence length in the single-shot path but only with chunk size ($C = 256$) under KV-Fold.

Third, single-shot prefill is substantially faster where it fits. At $T = 96$K, it completes in 19.3 s compared with 96.9 s for KV-Fold, a roughly $5\times$ difference. The operational benefit of identity KV-Fold is therefore reduced peak working memory and feasibility at the tested 128K point, not higher single-request prefill throughput.

Hardware dependence. The feasibility boundary is specific to the tested hardware and model. Smaller-memory devices will generally reach their allocation limit at shorter contexts, while larger-memory devices can move the boundary later. We therefore treat the 96K–128K interval as an empirical result for the 40 GB A100 rather than a hardware-independent threshold.

B.1 Decode amortization at the operational frontier

The prefill cost in Table 4 is paid once per request; the resulting KV cache is then reused during autoregressive decode. The contribution of prefill to end-to-end latency therefore depends on response length. We characterize this amortization at each tested operating point, with particular attention to $T = 128$K, where identity KV-Fold fits but single-shot SDPA does not.

Setup. For each $T \in \{32\text{K}, 64\text{K}, 96\text{K}, 128\text{K}\}$, we run identity KV-Fold prefill ($C = 256$) and single-shot SDPA prefill where it fits. We then greedy-decode 500 tokens from the resulting cache and record per-step wall-clock. Decode latency depends on the final cache size and decoding kernel, not on the route used to construct that cache.

Decode is independent of the prefill route. At each T where both prefills complete, measured per-token decode latencies agree within measurement resolution (Table 5). This is expected because both methods produce the same final KV state up to the numerical residual characterized in Section 3,

Table 5: Per-token decode latency after prefill on Llama-3.1-8B using a 40 GB A100. Values are medians over 500 decoded tokens. Identity KV-Fold and single-shot prefill agree within measurement resolution wherever both fit; variation across T reflects cache-size scaling.

T	KV-Fold ms/tok	single-shot ms/tok	cache size
32K	36.3	36.1	4.29 GB
64K	37.5	37.5	8.59 GB
96K	46.8	46.8	12.88 GB
128K	56.1	OOM	17.18 GB

Table 6: Decode length at which prefill accounts for 50%, 25%, or 10% of total wall-clock, computed from measured prefill and decode times. "OOM" indicates infeasible single-shot prefill.

Method, T	50% threshold	25% threshold	10% threshold
KV-Fold, 32K	386 tok	1,159 tok	3,477 tok
KV-Fold, 64K	1,197 tok	3,593 tok	10,779 tok
KV-Fold, 96K	2,042 tok	6,128 tok	18,384 tok
KV-Fold, 128K	2,932 tok	8,798 tok	26,396 tok
single-shot, 32K	99 tok	298 tok	895 tok
single-shot, 64K	263 tok	790 tok	2,370 tok
single-shot, 96K	408 tok	1,225 tok	3,675 tok
single-shot, 128K	OOM	OOM	OOM

after which decode uses the same cache representation and kernel. Variation across T reflects the increasing cost of reading a larger cache.

Amortization thresholds. If prefill takes time P and decoding costs D seconds per token, the decode length at which prefill contributes a fraction p of total wall-clock is

$$N = \frac{P}{D}\left(\frac{1}{p} - 1\right).$$

Table 6 reports the corresponding thresholds for $p \in \{0.50, 0.25, 0.10\}$.

At $T = 128$K, KV-Fold prefill falls below 25% of total wall-clock after approximately 8,800 decoded tokens and below 10% after approximately 26,000. Thus responses of hundreds to a few thousand tokens remain substantially prefill-dominated at this operating point.

Effective throughput at a fixed decode budget. For a 1,024-token response, we compute end-to-end effective throughput as

$$\frac{1024}{\text{prefill}_s + 1024\,\text{decode}_{s/\text{tok}}}.$$

Table 7 reports the result.

Interpretation. At $T \leq 96$K, where both methods fit, single-shot SDPA is approximately 2–3× faster end-to-end at a 1,024-token decode budget, with the difference arising primarily from prefill. At $T = 128$K, identity KV-Fold completes at approximately 4.6 decoded tokens/s under this workload, while the single-shot prefill does not fit. These measurements characterize the latency–memory trade-off rather than establishing a throughput advantage for fold-style execution.

Multi-request serving is not measured here. Chunked prefill can also support multi-request serving, where chunks of one request are interleaved with decode steps from another, as in Sarathi-Serve (Agrawal et al., 2024). We do not evaluate that setting here; doing so would require a serving-system benchmark rather than the single-request measurements used in this work.

C SELECTIVE KV-CACHE QUANTIZATION WITH QUESTION-AWARE SALIENCE

In the terminology of Section 5, this appendix studies nonidentity transformations of the carried KV state, with emphasis on precision reduction and selective preservation. Section 5 shows that

Table 7: End-to-end throughput for a 1,024-token decode on Llama-3.1-8B using a 40 GB A100. Total time includes prefill and decode.

Method, T	prefill	decode/tok	total	eff tok/s
KV-Fold, 32K	14.0 s	36.3 ms	51.2 s	19.98
KV-Fold, 64K	44.9 s	37.5 ms	83.3 s	12.29
KV-Fold, 96K	95.5 s	46.8 ms	143.4 s	7.14
KV-Fold, 128K	164.7 s	56.1 ms	222.2 s	4.61
single-shot, 32K	3.6 s	36.1 ms	40.5 s	25.28
single-shot, 64K	9.9 s	37.5 ms	48.2 s	21.22
single-shot, 96K	19.1 s	46.8 ms	67.0 s	15.29
single-shot, 128K	OOM	OOM	OOM	OOM

uniform per-step quantization degrades retrieval gradually: int8 retains $13/14 = 93\%$ and int4 retains $24/33 = 73\%$ across the $T = 128$K retrieval sweep, while more aggressive perturbations degrade more sharply. Here we characterize this behavior in greater detail and study *selective* per-step quantization, which preserves a subset of cache positions at full precision while quantizing the remainder using a question-aware salience signal.

The motivation is that uniform quantization leaves measurable headroom at low bit widths. If positions most relevant to the eventual query can be localized, preserving those positions at full precision may improve retrieval while leaving most of the cache quantized.

We use an LSTM-gate analogy only to organize these interventions. Precision allocation acts like an output-style gate over retained state, while the two-tier design of Appendix C.9 additionally introduces retention and eviction decisions analogous to input and forget gates. These are functional analogies rather than architectural equivalences: the transformer is unchanged and no additional parameters are trained.

C.1 Quantization protocol

We use per-token symmetric round-trip quantization. For each cache position with key or value vector $z \in \mathbb{R}^{d_{\text{head}}}$ in bfloat16, we compute

$$s = \frac{\max_j |z_j|}{L}, \qquad L = 2^{n-1} - 1,$$

where $L = 7$ for $n = 4$ and $L = 3$ for $n = 3$. The quantized representation is

$$\hat{z} = s \cdot \text{clamp}(\text{round}(z/s), -L, L)\,,$$

cast back to bfloat16 storage. Each row of $d_{\text{head}} = 128$ elements shares one scale; K and V are quantized independently with separate scales. Quantization is applied after RoPE rotation because the cache stores post-rotation keys.

This is simulated quantization: round-trip conversion reproduces the precision loss of an int$n$ representation while retaining bf16 storage. It therefore isolates the effect of reduced numerical precision without measuring the memory savings or kernels of a production int$n$ KV-cache implementation.

In the per-step protocol, quantization is applied after each chunk's forward pass to the carried cache before it is consumed by the next chunk. Later chunks therefore attend to already quantized state, so quantization errors can propagate through subsequent computation. This is the protocol used for the int8, int4, and int3 results below.

C.2 The bit-width search: identifying a partial-failure regime

Selective preservation is most informative at an operating point where uniform quantization has measurable headroom: not so mild that almost all trials succeed, and not so destructive that preserving a subset of positions cannot help. We therefore surveyed bit widths at $T = 4$K and $T = 128$K.

At $T = 4$K, $d = 14$, uniform per-step int4 retains $5/5$ trials, leaving no observed failures to rescue. At the same scale, int2 retains $0/5$, and preserving half of the cache at bf16 also retains $0/5$. This

Table 8: Question-aware salience ranks at $T = 128$K, $d = 511$ on Llama-3.1-8B-Instruct over $N = 10$ trials. "Min rank" is the rank of the highest-salience needle token among all 131,072 cache positions.

Trials with needle in top 0.01% (≤ 13 positions)	$10/10$
Trials with needle in top 0.10% (≤ 131 positions)	$10/10$
Trials with needle in top 1.00% ($\leq 1{,}310$ positions)	$10/10$
Median needle min rank	~ 10
Maximum needle min rank observed	~ 45

suggests that, in this setting, corruption of the quantized fraction can remain large enough to prevent retrieval even when high-salience positions themselves are preserved.

We therefore move to $T = 128$K, where substantially more chunk transitions expose a broader partial-failure regime. At $d = 511$, uniform per-step int4 retains $7/10 = 70\%$ (Section C.3), while uniform per-step int3 retains $3/10 = 30\%$ (Section C.4). We use int3 as the main setting for selective preservation because it leaves substantial room for both rescue and failure.

C.3 Statistical characterization of the int4 baseline at $d = 511$

Across the $T = 128$K distance grid, uniform per-step int4 retains $24/33 = 73\%$. To characterize the deepest point separately, we run $N = 10$ trials at $d = 511$ with distinct keys, values, and PG-19 windows. Int4 retains $7/10 = 70\%$ (Wilson 95% CI $[40\%, 89\%]$), consistent with the aggregate distance-grid result. At lower distances, it retains $9/9$ across $d \in \{1, 127, 255\}$ with three trials at each distance.

These measurements suggest that the observed int4 failures are concentrated near the deepest tested operating point. Because the uniform int4 baseline already succeeds on most trials, int3 provides a more informative regime for studying selective preservation.

C.4 int3 as a partial-failure regime

At $n = 3$ bits, each quantized row uses the seven levels $\{\pm s, \pm 2s, \pm 3s, 0\}$. Cosine similarity between round-tripped int3 vectors and their bf16 originals is approximately 0.92, compared with approximately 0.99 for int4. At $T = 128$K and $d = 511$, uniform per-step int3 retains $3/10 = 30\%$ (Wilson 95% CI $[11\%, 60\%]$). At $d \in \{1, 127, 255\}$, three-trial checks yield $1/3$, $1/3$, and $2/3$, respectively.

The failures are therefore not clearly monotone in chain depth. In particular, retrieval can fail even at $d = 1$, suggesting that local corruption of the needle-bearing representation is sufficient to cause some failures without requiring long-horizon accumulation. We use this regime to test whether preserving query-relevant positions at bf16 can recover part of the lost retrieval accuracy.

C.5 Question-aware salience as a localization signal

To identify cache positions relevant to the query, we compute a post-hoc question-aware salience signal. After completing the bf16 fold, the question tokens are forwarded against the accumulated cache with eager attention and `output_attentions=True`. Per-position salience is the mean attention weight from question tokens to each earlier cache position, averaged across attention heads, query tokens, and layers.

To control transient memory at $T = 128$K, the question is processed in blocks of 16 tokens with `use_cache=True`; afterward, the cache is restored from a snapshot to remove appended question keys and values. The transient peak per block is approximately 2.5 GB.

The resulting signal is highly localized. At $T = 128$K with the needle in chunk 0 ($d = 511$), needle content appears among the highest-salience positions in all $N = 10$ trials (Table 8). This is expected for an oracle-like signal because the eventual question directly identifies the content being sought.

Table 9: Eviction-based selection at $T = 4$K with a fixed bank of $M = 1024$ positions and recency-debiased attention scoring. Each cell contains five trials. The $d = 7$ result remains $1/5$ for every tested α.

α	$d = 1$	$d = 7$	$d = 14$	$d = 15$
0.00	$5/5$	$1/5$	$0/5$	$0/5$
0.25	$3/5$	$1/5$	$1/5$	$0/5$
0.50	$0/5$	$1/5$	$1/5$	$1/5$
0.75	$0/5$	$1/5$	$2/5$	$2/5$
1.00	$0/5$	$1/5$	$4/5$	$4/5$

Two qualifications are important. First, the signal is oracle-like: it uses the eventual question, information unavailable to a selector that must decide what to retain during document processing. It should therefore be viewed as a reference for how well task-relevant positions can be localized once the query is known, not as an upper bound on all possible query-blind methods. Second, it applies naturally to settings where a long context is processed before a trailing question arrives, such as document retrieval or archival QA; it does not directly apply to open-ended streaming settings where retention decisions must be made before the query is known.

C.6 Why query-blind salience falls short in the tested setting

We also test whether chunk-time attention provides a useful query-blind retention signal. At $T = 4$K, we maintain a fixed bank of $M = 1024$ positions and retain the top-M entries according to attention from the current chunk. Because raw attention exhibits a strong recency bias, we calibrate a recency prior on a held-out PG-19 window and divide the scores by this prior raised to an exponent α. Thus $\alpha = 0$ recovers raw attention and $\alpha = 1$ applies the full debiasing correction. We test $\alpha \in \{0, 0.25, 0.50, 0.75, 1.00\}$ at $d \in \{1, 7, 14, 15\}$.

At $d = 7$, every tested value of α yields $1/5$ retention (Table 9). The behavior at other distances changes substantially with α: raw attention performs best at short distance, while stronger recency correction helps at larger distances. The invariant $1/5$ result at $d = 7$ therefore represents an empirical floor for this family of chunk-time attention selectors, not an information-theoretic bound.

These results suggest a limitation of the tested query-blind signal. Chunk-time queries are generated by the current chunk, not by the eventual question, and therefore need not assign distinctive attention to content that will only become relevant later. Recency correction can change which historical positions are emphasized, but in these experiments it does not reliably separate the needle from neighboring PG-19 content at all depths. This motivates the question-aware signal used above, while leaving open whether stronger query-blind selectors can do better.

C.7 Selective per-step quantization at int3

We next test selective preservation. Starting from the bf16 reference chain, we rank positions by question-aware salience and form a keep-mask covering the top fraction f. We then run a fresh per-step int3 chain: positions in the keep-mask remain at bf16, while all remaining positions are quantized after each chunk. The mask is shared across all 32 layers for a retained position.

At $f = 0.05$, the mask contains approximately 6,553 of 131,072 positions and includes all needle tokens in every trial. Selective preservation retains $5/10 = 50\%$ at $d = 511$, compared with $3/10 = 30\%$ under uniform int3 on the same paired trials. Two failures under uniform int3 are rescued, while no uniformly successful trial is broken. McNemar's exact two-sided test on the two discordant pairs gives $p \approx 0.50$, so this directional improvement is not statistically significant at $N = 10$.

Increasing the preserved fraction to $f = 0.10$ and $f = 0.25$ raises retention to $6/10 = 60\%$ in both cases. The improvement therefore saturates in this small sample rather than increasing monotonically with the preserved fraction. Moreover, selective preservation can break trials that uniform int3 retrieves: one such reversal occurs at $f = 0.10$ and two at $f = 0.25$.

Table 10: Per-step quantization retention at $T = 128$K, $d = 511$ on Llama-3.1-8B-Instruct over $N = 10$ paired trials. "Rescues" count trials where uniform int3 fails but selective preservation succeeds; "breaks" count the reverse.

Method	retention	Wilson 95% CI	Δ vs uniform	rescues	breaks
uniform per-step int4	$7/10 = 70\%$	$[40\%, 89\%]$	–	–	–
uniform per-step int3	$3/10 = 30\%$	$[11\%, 60\%]$	ref	–	–
selective int3, $f = 0.05$	$5/10 = 50\%$	$[24\%, 76\%]$	+20 pp	2	0
selective int3, $f = 0.10$	$6/10 = 60\%$	$[31\%, 83\%]$	+30 pp	4	1
selective int3, $f = 0.25$	$6/10 = 60\%$	$[31\%, 83\%]$	+30 pp	5	2

The data are consistent with two effects. Preserving a relevant position can remove local quantization error from its key and value, which can rescue retrieval. At the same time, mixing bf16 and int3 positions changes the relative attention scores across the cache and can alter the attention distribution in ways that occasionally hurt retrieval. We treat this as a plausible mechanism rather than a confirmed causal decomposition.

Across the three selective settings, rescues outnumber breaks, but the sample is small. McNemar's exact two-sided tests give $p \approx 0.50$ at $f = 0.05$, $p \approx 0.38$ at $f = 0.10$, and $p \approx 0.45$ at $f = 0.25$. Thus the appropriate conclusion is that question-aware selective preservation shows a directional improvement over uniform int3 in these experiments, but the evidence is not yet statistically conclusive.

C.8 Discussion and limitations

These experiments identify an int3 regime in which uniform per-step quantization fails on a substantial fraction of trials and question-aware selective preservation recovers some of those failures. The result complements the more robust int4 behavior reported in the main paper, while illustrating that aggressive quantization introduces a nontrivial interaction between precision allocation and attention.

Three limitations bound the result.

First, selection is question-aware and is therefore restricted to settings where the query is available after the long context has been processed. It does not solve query-blind retention for streaming generation.

Second, quantization is simulated by round-trip conversion while the cache remains stored in bf16. The experiments therefore measure retrieval under reduced precision but do not realize the corresponding memory reduction. Real int$n$ storage and kernels, such as those used by KIVI (Liu et al., 2024) or KVQuant (Hooper et al., 2024), remain complementary engineering directions.

Third, the selective experiments use only $N = 10$ paired trials per setting. The observed differences are directional but not statistically significant. In particular, the apparent saturation between $f = 0.10$ and $f = 0.25$, and the small break rate introduced by selective preservation, should be interpreted as empirical patterns rather than estimates of a stable optimum.

Overall, these results do not alter the main finding that identity KV-Fold preserves long-range retrieval and that moderate per-step quantization remains usable at large depth. Instead, they identify a more aggressive precision regime in which query-aware preservation can partially recover retrieval failures.

C.9 Two-tier memory: a proposed route toward constant-memory KV-Fold

The experiments above retain the main paper's linear-memory profile: KV-Fold carries state whose size grows with total context length, while selective preservation only changes precision allocation within that state. We therefore consider a separate, currently unimplemented two-tier design intended to explore constant-memory execution.

Proposed mechanism. The cache is divided into three fixed-size components: S attention sinks, a recent window of W positions, and a long-term bank of L positions. The total retained state is therefore $S + W + L$, independent of total context length. When a position exits the recent window, a

promotion rule may copy it into the long-term bank; when that bank is full, an eviction rule removes an existing entry.

Gate analogy. Using the organizational LSTM analogy above, promotion acts like an input decision, eviction like a forget decision, and optional precision allocation within the long-term bank like an output decision. This analogy describes the roles of the operations rather than an architectural equivalence to an LSTM.

At one endpoint, retaining only sinks and a fixed recent window resembles StreamingLLM (Xiao et al., 2024); at the other, retaining all positions recovers the full identity fold. Intermediate choices of W and L are intended to explore the memory–retrieval trade-off between these endpoints.

Promotion signal. The main unresolved issue is deciding which positions should enter the long-term bank before the eventual query is known. The negative result of Section C.6 shows that the tested snapshot-style chunk-attention selectors are not sufficient across all distances. One possible alternative is *cumulative attention exposure*: accumulate the attention received by a position while it remains in the recent window and promote positions with high integrated exposure. Whether this signal reliably identifies information needed later is an open empirical question.

Eviction within the long-term bank. Once the long-term bank is full, candidate eviction policies include least-recently-used and lowest-cumulative-exposure rules. The relative sizes of W and L provide an additional control over the intended trade-off between recent-context fidelity and retention of distant content.

Status. The two-tier protocol has not yet been implemented or evaluated. The existing experiments provide constraints on its design rather than evidence that it succeeds: question-aware salience demonstrates that task-relevant positions can be highly localized once the query is known, selective preservation shows that retaining a subset of state can improve an aggressively quantized cache, and the query-blind experiments show that the tested chunk-time attention signals are not reliable enough by themselves. Evaluating cumulative-exposure promotion and measuring the resulting retrieval–memory trade-off is left for future work.

D A MEASUREMENT PITFALL IN CACHE-BASED EVALUATION

An earlier version of Section 3 reported a drift plateau of 0.040–0.122 nats, depending on architecture, between the KV-FOLD and FULL conditions. The drift rose over the first several chain depths and then remained approximately flat; it was also insensitive to a bf16-to-fp32 precision change and stable across chunk sizes from 128 to 1,024. These properties initially made the plateau appear to be a robust feature of the recurrence. It was instead caused by an evaluation artifact. This appendix documents the cause, the forensic attribution, and the corrected protocol because the same pitfall can affect any evaluation harness that scores against a live cache in a separate forward pass.

The pitfall. In the HuggingFace transformers versions we tested, attention layers call `past_key_values.update(...)` when a cache object is supplied even when `use_cache=False`; the flag controls whether the cache is returned, not whether the supplied object is mutated. Thus a scoring call of the form `model(chunk, past_key_values=cache, use_cache=False)` can append the chunk's keys and values to the cache in place. We verified this behavior with a minimal two-layer model: cache length increased from 8 to 16 after a `use_cache=False` scoring forward and then to 24 after the subsequent cache-building forward.

Consequence in a chained evaluation loop. The original evaluation used two forwards per chunk: one to score the chunk against the accumulated cache and one to extend the cache. Because the scoring forward also mutated the cache, chunk c was appended twice. After chunk c, the cache therefore contained $(2c{+}1)C$ positions rather than $(c{+}1)C$: chunk 0 appeared once and chunks $1, \ldots, c$ twice, with each duplicate pair using identical RoPE positions.

For $c \geq 2$, the scored prefix consequently assigned duplicated attention mass to all intermediate chunks. The duplicated fraction of the prefix is

$$\frac{c-1}{2c-1},$$

Table 11: Forensic attribution at the original operating point (OLMoE-1B-7B-Instruct, $T = 4{,}096$, $C = 256$). "Original" is the stored output of the original evaluation ($N = 100$ windows); "verbatim" reruns the original code unchanged ($N = 40$); "clone-isolated" differs only in that the scoring forward receives a cloned cache.

depth	original	verbatim rerun	clone-isolated
1	-0.0007	$+0.0012$	$+0.0012$
2	$+0.0575$	$+0.0553$	-0.0022
3	$+0.0769$	$+0.0773$	-0.0009
5	$+0.1039$	$+0.0990$	$+0.0000$
7	$+0.1277$	$+0.1238$	-0.0032
9	$+0.1138$	$+0.1098$	-0.0022
11	$+0.1264$	$+0.1271$	$+0.0022$
13	$+0.1144$	$+0.1110$	-0.0001
15	$+0.1223$	$+0.1214$	-0.0016
plateau mean ($d \in [7, 15]$)	$+0.122$	$+0.1180$	-0.0008

which is 0 at depth 1, then rises through $1/3$, $2/5$, $4/9$, and approaches $1/2$. This mirrors the observed rise-and-plateau pattern. Because the corruption is structural, changing numerical precision does not remove it; because the same duplication occurs for every chunk size, varying C also preserves the qualitative effect. The earlier precision and chunk-size ablations therefore probed dimensions to which the artifact was largely insensitive.

Forensic attribution. We reran the original evaluation at its original operating point (OLMoE-1B-7B-Instruct, $T = 4{,}096$, $C = 256$) under three protocols: the original code unchanged ($N = 40$ windows), a *clone-isolated* variant in which the scoring forward receives a deep copy of the cache, and a *single-forward* variant that scores and extends the cache in one forward. The expected outcomes were fixed before running the comparison.

The verbatim rerun reproduces the plateau: the mean drift over depths 7–15 is $+0.1180$ nats, close to the stored original value of $+0.122$. Instrumented cache lengths follow exactly $(2c{+}1) \cdot 256$. Under clone isolation, the plateau collapses to a mean drift of -0.0008 nats, with maximum per-depth $|\text{drift}| = 0.0051$; the single-forward variant agrees with the clone-isolated result to four decimal places. A remaining bookkeeping asymmetry—the chunk-boundary token is included in the FULL condition but excluded from the KV-FOLD condition—contributes at most 0.0032 nats. Table 11 reports the per-depth comparison.

Scope of the correction. The affected measurements are the earlier drift values, the associated precision ablation, and the chunk-size comparison; all evaluated the duplicated-context protocol rather than identity KV-Fold as defined in Section 2. The other experiments in this paper are unaffected: the retrieval studies, StreamingLLM comparison, operational-frontier and amortization measurements, and the quantization and selective-preservation experiments construct the chain with a single forward per chunk and do not pass the live cache into a separate scoring forward.

The corrected drift measurements appear in Table 3. A complementary logit-level check on Qwen2.5-1.5B further isolates the residual numerical effect: under the corrected protocol, identity KV-Fold and the monolithic forward agree in fp32 to a maximum absolute logit difference of 5.9×10^{-4}, with identical greedy decisions at every tested position. In bf16, the residual discrepancy remains approximately depth-flat, with a late-to-early band ratio of 1.04. Together, these results attribute the previously reported plateau to the evaluation protocol rather than to accumulation intrinsic to the identity fold.

Recommendations. For cache-based evaluation, the safest approach is to score and extend the cache in the same forward, since that forward already provides the hidden states and logits needed for evaluation. If a separate scoring forward is necessary, the cache should be cloned or otherwise isolated before use. We additionally recommend asserting the expected cache length after every chunk, for example

```
cache.get_seq_length() == (c+1)*C,
```

as a simple protocol invariant. More generally, evaluation code should treat the mutation semantics of cache objects as part of the library contract rather than assuming that `use_cache=False` implies immutability.

A second instance of the same pattern. In related measurement work, we encountered a distinct interface pitfall: in the transformers version tested, the final entry of a decoder's returned `hidden_states` is already post-normalization, so applying the model's final normalization again inflates measured hidden-state differences. Neither behavior is necessarily a library bug; both arise when an evaluation harness assumes semantics different from those of the library interface. These cases motivate explicitly checking the mutation, normalization, and shape invariants on which intervention-style evaluations depend.

E CACHE-AS-MESSAGE PROTOCOL

This appendix details the cross-context transfer experiment of Section 5 (Table 2).

Producer. Llama-3.1-8B-Instruct processes a $T = 32{,}768$-token PG-19 window using KV-Fold with $C = 256$ (128 chunks). The needle sentence ("*The magic number for [key] is [value].*") is inserted at the start of chunk 1, placing it at chain depth $\sim$126 from the question. The producer uses eager attention so that attention weights are available for salience extraction; this changes runtime but not the underlying model computation. The completed cache is snapshotted to host memory.

Salience. The question is forwarded once against the full producer cache with `output_attentions=True`. Per-position salience is the mean attention from question tokens to each of the T cache positions, averaged over layers, heads, and question tokens, using the signal of Appendix C.5. Because this forward mutates the live cache, its result is discarded and every transfer condition is constructed from the host snapshot.

Transmission and consumer. For transmitted fraction f, the shared-position selector retains the top fT positions by salience, always unioned with the $S = 4$ attention-sink positions. Random and most-recent sets of matched size serve as controls. A fresh consumer cache is formed by gathering keys and values at the selected positions, in sorted position order, for every layer. Unless otherwise stated, keys retain the RoPE rotations associated with their original absolute positions. The consumer uses the same checkpoint and receives only the selected KV state and the question, whose position identifiers begin at T; it never processes the document tokens. Answers are greedily decoded for 30 tokens and scored by exact match of the first 5-digit number.

Conditions and trials. A pilot run ($N = 10$) evaluates the full cache, question-aware selection at $f \in \{0.005, 0.01, 0.05\}$, and random and recency controls at $f = 0.01$. The main run ($N = 20$) extends the question-aware sweep to $f \in \{0.005, 0.002, 0.001, 0.0005\}$, adds SnapKV-style per-head selection at $f \in \{0.005, 0.001, 0.0005\}$, evaluates random and recency controls at $f = 0.005$, and includes the compacted condition described below. Within each trial, all conditions share the same producer cache and question; trials use distinct keys, values, and PG-19 windows.

Results detail. Question-aware shared-position transfer succeeds in $20/20$ trials at 167, 69, and 36 retained positions, and in $19/20$ at 20 positions. Median needle coverage is 12, 11, 9, and 5 tokens, respectively. Thus perfect transfer persists even after the selected set contains only part of the needle span, while performance begins to degrade at the smallest tested shared-position budget. The pilot agrees on all overlapping conditions (combined: full cache $30/30$; question-aware $f = 0.005$, $30/30$).

Random selection covers at most one needle token in any main-run trial, and recency covers none because the needle lies far outside the recent window; both achieve $0/20$ retrieval. Their decoded outputs typically state that the requested information is unavailable, contrasting with the malformed outputs observed under aggressive quantization in Appendix C.2. Position selection removes entries while leaving retained keys and values unchanged, so these failures reflect missing rather than numerically corrupted state.

Selector comparison (SnapKV-style). For each layer and KV head, we compute question attention averaged within each GQA group, max-pool along positions with kernel 7 as in SnapKV (Li et al., 2024), and retain the per-head top-k positions together with the sinks. Each $(\text{layer}, \text{head})$ therefore uses its own position set, and the consumer cache is constructed by per-head gather. Relative to the

shared-position selector, this requires per-layer-per-head index metadata rather than one global index set.

Transfer succeeds in $20/20$ trials at every tested budget, including 20 positions per head. The median across the 256 (layer, head) retained sets contains no needle tokens despite perfect transfer. This is consistent with task-relevant information being concentrated in a subset of heads rather than being uniformly represented across them. A direct per-head coverage analysis is left for future work.

Position compaction. The compacted condition re-indexes the sorted survivors to contiguous positions $0, \ldots, m-1$. For an entry originally at position p_i and moved to position i, we update the stored RoPE-rotated key using the positional offset $\Delta_i = i - p_i$:

$$k' = k\cos(\Delta_i\omega) + \mathrm{rotate_half}(k)\sin(\Delta_i\omega),$$

where ω is the inverse-frequency vector; values are unchanged, and the question begins at position m. Transfer remains $20/20$, showing that the selected KV state can be re-addressed into the consumer's positional frame.

Paraphrase arm. Ten additional trials replace the needle and question with profession synonym pairs sharing no profession word. For example, the needle says "*The vault combination chosen by the apiarist is [value].*" while the question asks about "a beekeeper," with analogous pairs such as cartographer/mapmaker and vintner/winemaker. Salience is computed from the paraphrased question. The full-cache condition succeeds in $10/10$ trials, as do question-aware selection at $f = 0.005$ and $f = 0.01$, with median needle coverage 14 of approximately 15 tokens. This indicates that the selector need not depend on exact lexical overlap between the question and the stored sentence.

NLL transfer. For each main-run trial, we score the next 256 document tokens using four contexts: full cache, question-aware selection at $f = 0.005$, recency selection at $f = 0.005$, and no cache. The first three continue from position T; the isolated condition begins from position 0. Mean NLL over 20 trials is 2.324 for the full cache, 2.538 for question-aware selection, 2.591 for recency, and 2.796 in isolation. Relative to the 0.472-nat isolated-to-full gap, the 167-position message recovers approximately 55% under question-aware selection and 44% under recency.

This differs from the exact-retrieval result at the same bandwidth: question-aware selection achieves $20/20$ retrieval while recency achieves $0/20$. Thus next-token prediction degrades more gradually under aggressive state reduction, whereas exact task retrieval depends strongly on which positions are retained. A sinks-only condition would help separate the contribution of the four attention sinks to the NLL recovery and is left as an additional calibration.

Qwen replication status. Applying the same protocol to Qwen2.5-7B-Instruct at $T = 16{,}384$ ($N = 10$) reveals a low full-cache consumer ceiling: the full-cache condition succeeds in only $5/10$ trials. Conditional on that ceiling, question-aware $f = 0.005$ also succeeds in $5/10$ trials and agrees with the full-cache outcome in $9/10$ trials; $f = 0.01$ succeeds in $6/10$, including one trial in which the pruned cache succeeds while the full cache fails. Random and recency selection both achieve $0/10$.

Because the full-cache consumer itself is unreliable under this harness, these measurements do not provide a clean cross-architecture replication and are excluded from Table 2. One possible source is the consumer prompt construction, which appends a chat-template block after raw document-state processing, but we leave the diagnosis unresolved.

Scope. For the shared-position selector, perfect transfer persists down to 36 of 32,768 document positions, while the 20-position budget succeeds in $19/20$ trials. Per-head selection succeeds at the smallest tested budget of 20 positions per head. These experiments therefore establish that task-relevant information can be transferred through a very sparse KV message once the receiver's question is available.

The selection mechanism remains receiver-driven: question-aware salience is computed against the full producer cache before transmission. The query-blind experiments of Appendix C.6 show that the particular chunk-time attention selectors tested there do not reach this operating point; they do not establish that query-blind selection is impossible. Open questions include sinks-only NLL calibration, per-head coverage, the Qwen consumer harness, and transfer under producer–consumer mismatch, including different prompts or mixture-of-experts routing regimes.

Table 12: Needle-in-a-haystack retrieval on Qwen2.5-7B-Instruct. FULL uses a monolithic forward, ISOLATED answers without haystack context, and KV-FOLD uses the identity fold. Twenty trials are run at each distance.

Distance d	FULL	ISOLATED	KV-FOLD	ratio
1	100% (20/20)	0% (0/20)	100% (20/20)	1.00
15	100% (20/20)	0% (0/20)	100% (20/20)	1.00
31	100% (20/20)	0% (0/20)	100% (20/20)	1.00
62	100% (20/20)	0% (0/20)	100% (20/20)	1.00
overall	100% (80/80)	0% (0/80)	100% (80/80)	1.00

Table 13: Multi-needle retrieval. K independent (key, value) pairs are inserted at evenly spaced positions and queried separately. Identity KV-Fold retrieves $175/176$ needles overall, with all needles recovered in $35/36$ trials.

Model	T	K	trials	per-needle	all-correct	ISOLATED
Qwen2.5-7B-Instruct	16K	2	10	20/20 (100%)	10/10	0/20
Qwen2.5-7B-Instruct	16K	4	10	39/40 (97.5%)	9/10	0/40
Qwen2.5-7B-Instruct	16K	8	10	80/80 (100%)	10/10	0/80
Llama-3.1-8B-Instruct	128K	4	3	12/12 (100%)	3/3	0/12
Llama-3.1-8B-Instruct	128K	8	3	24/24 (100%)	3/3	0/24
combined			36	175/176 (99.4%)	35/36	0/176

F RETRIEVAL PROTOCOL AND ADDITIONAL RETRIEVAL RESULTS

Protocol. For each trial:

1. Sample a 16,384-token PG-19 window.
2. Sample a key from a fixed list of uncommon English words (*e.g.*, "amaranth", "obsidian", "halcyon") and a uniformly random 5-digit value.
3. Insert the sentence "*The magic number for [key] is [value].*" at a controlled chunk position.
4. After all 64 data chunks, ask: "*Earlier in the document, what was the magic number associated with [key]? Reply with only the number.*"
5. Greedily decode 30 tokens, extract the first 5-digit number, and compare it with the gold value.

We test four needle placements, parameterized by the number of chunk transitions d between the needle and the question: $d \in \{1, 15, 31, 62\}$. Here $d = 1$ places the needle immediately before the question, while $d = 62$ places it near the beginning of the 64-chunk context. Each placement is evaluated over 20 independent trials on Qwen2.5-7B-Instruct.

Multi-needle retrieval. We strengthen the single-needle protocol by inserting K independent (key, value) pairs at evenly spaced chunk positions and querying each key separately. This follows the multi-needle style of long-context evaluation used in RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2024), and tests whether multiple distinct facts remain simultaneously accessible from the carried KV state.

On Llama-3.1-8B at $T = 128$K, we test $K \in \{4, 8\}$ with three trials per setting. Identity KV-Fold retrieves all $36/36$ needles, and every trial recovers all K inserted facts; the ISOLATED condition retrieves $0/36$. On Qwen2.5-7B at $T = 16$K, a larger sweep with $K \in \{2, 4, 8\}$ and 10 trials per setting retrieves $139/140$ needles, with one miss for $K = 4$ at $d = 42$. Across both models, retrieval is therefore $175/176$ per needle and $35/36$ for complete recovery of every needle in a trial.

Decode and per-chunk timing. Once the final cache has been constructed, decode uses the same cache representation regardless of prefill route. Measured per-token latency ranges from 36.3 to 56.1 ms/token between 32K and 128K (Appendix B.1). At $T = 128$K, the measured end-to-end rate is approximately 4.6 decoded tokens/s for a 1,024-token response, and prefill falls below 25% of total wall-clock only beyond approximately 8,800 decoded tokens.

Table 14: Identity KV-Fold and StreamingLLM at $T = 128$K. NLL is measured over three PG-19 windows and retrieval over three trials per distance. StreamingLLM uses a 1024-position bounded cache. *One KV-Fold trial at $d = 511$ produced no extractable 5-digit answer; together with the separate $12/12$ result at the same setting in Section 4.1, retrieval is $14/15$.

Method	NLL	peak GPU	total wall	d=1	d=31	d=255	d=511
KV-Fold	2.46 ± 0.12	35.6 GB	166 s	$3/3$	$3/3$	$3/3$	$2/3^*$
StreamingLLM	2.66 ± 0.17	16.6 GB	22 s	$3/3$	$0/3$	$0/3$	$0/3$

Within the fold, per-chunk wall-clock increases with accumulated cache size, from approximately 0.103 to 0.335 s over the measured operating points, consistent with the expected $O(C \cdot \text{cache_size})$ attention cost. The complete 128K chain takes approximately 171 s.

G STREAMINGLLM COMPARISON IN DETAIL

StreamingLLM (Xiao et al., 2024) provides a bounded-cache reference point by retaining a small set of attention sinks together with a sliding window of recent tokens; LM-Infinite (Han et al., 2024) uses a related long-context strategy. We compare this operating point directly with identity KV-Fold.

Both methods are evaluated on Llama-3.1-8B-Instruct at $T = 128$K and $C = 256$. StreamingLLM retains 4 sink tokens and a 1020-token recent window, for a total capacity of 1024 positions. Because all tested positions lie within the model's native context range, we use position-preserving eviction without additional RoPE re-anchoring. We measure next-token NLL over three PG-19 validation windows and needle retrieval at $d \in \{1, 31, 255, 511\}$ with three trials per cell.

The two methods occupy different memory–retrieval operating points. StreamingLLM uses approximately 0.13 GB of KV state independent of total context length in this configuration, compared with 17.18 GB for the full 128K KV cache, and completes the measured prefill in 22 s rather than 166 s. At the same time, its retained window no longer contains the needle at $d \in \{31, 255, 511\}$, where retrieval is $0/3$ in each cell. Both methods retrieve $3/3$ at $d = 1$.

Identity KV-Fold retains the complete state and retrieves $3/3$ at $d = 31$ and $d = 255$, and $2/3$ at $d = 511$ in this stress test. The latter result is from a separate small-sample experiment and should not be conflated with the main retrieval sweep; combined with the $12/12$ result of Section 4.1, the corresponding retrieval count is $14/15$. The measured NLL is also lower for the full retained state (2.46 ± 0.12 versus 2.66 ± 0.17), while StreamingLLM provides substantially lower memory use and prefill latency.

H ADDITIONAL LOSSY-STATE AND CHANNEL RESULTS

Selective preservation. Question-aware salience—the mean attention from question tokens to each cache position when the question is evaluated against the full cache—localizes needle content within the top 0.01% of 131,072 positions in all 10 trials. Preserving the top fraction f at bf16 while quantizing the remaining state at int3 increases retrieval from $3/10$ under uniform int3 to $5/10$ at $f = 0.05$ and $6/10$ at $f = 0.10$. The paired results are directional but not statistically significant at $N = 10$ (McNemar $p \approx 0.4$–0.5). These results are consistent with preserving query-relevant state helping some trials, while the remaining quantized state can still alter the attention distribution enough to prevent retrieval.

Semantic selection. The selector does not require exact lexical overlap. In a paraphrase arm using profession synonyms—for example, a needle containing *apiarist* queried using *beekeeper*—question-aware salience retains a median 14 of approximately 15 needle tokens within the top 0.5% of positions. Both the full-cache condition and the selected-cache condition retrieve $10/10$ answers.

Task information versus general context. The sparse message does not preserve all of the document's predictive information. When scoring the next 256 document tokens, the 167-position question-aware message recovers approximately 55% of the 0.47-nat isolated-to-full NLL gap, while

a recency-selected message of the same size recovers approximately 44%. In contrast, exact task retrieval at the same bandwidth is $20/20$ for question-aware selection and $0/20$ for recency.

Thus the experiments show a sharp distinction between targeted retrieval and general next-token prediction: a sparse, query-aware KV message can preserve the designated fact even though it retains only part of the broader contextual information present in the full cache.