LAST LAYER KV CACHE IS ALL YOU NEED

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers store separate key-value (KV) states at every layer, causing memory to scale with network depth while preventing deep representations of the past from informing shallower future layers. We introduce LastKV, where all layers instead share the last-layer KV states of completed context. This simple change reduces long-term KV storage by approximately the number of layers while feeding deep past representations back into future computation, creating recurrence across both time and depth. To preserve parallel training, LastKV applies recurrence across chunks while retaining standard parallel attention within each chunk. Despite substantially compressing history, LastKV matches or outperforms standard Transformers across language modeling, state tracking, novel-view synthesis, and video generation.

1 INTRODUCTION

Transformers Vaswani et al. (2017) dominate sequence modeling largely because attention accesses past context directly, enabling parallel training unlike recurrent neural networks (RNNs) that relies on sequential state updates. However, each transformer layer stores its own key-value (KV) states for past tokens. For an L-layer model, the KV cache scales with both sequence length T and depth L, which imposes a large burden on GPU memory. Furthermore, information flows strictly layer-by-layer: each layer only retrieves past states at its own depth, never passing deeper past features into shallower future steps. Within a single forward pass, sequential computation per token is therefore bounded by network depth (Fan et al., 2020). For Transformers with saturated attention and floating-point values, Merrill et al. (2022) prove a constant-depth threshold-circuit upper bound.

These properties raise a natural question: *does every layer need its own history?* Once a token passes through the entire network, its deep representation is already fully computed. Reusing this representation across layers does more than cut redundant storage—it feeds completed deep features back into future token processing, turning shared history into a recurrent state.

We introduce **LastKV**, where every layer attends to the last-layer KV states of completed history (Figure 1). This single change offers two key benefits: (1) **memory reduction**, by sharing a single history across all layers so that the long-term cache for an L-layer model drops to roughly $1/L$ of a standard Transformer while keeping past tokens directly accessible; and (2) **recurrence across both depth and time**, by feeding last-layer representations of past tokens directly into all future layers, significantly increasing the effective computational depth along the sequence and improving its capability to solve complex tasks such as multi-step state tracking.

Applying recurrence token-by-token introduces sequential dependencies like an RNN, breaking parallel training. We therefore perform recurrence over *chunks*. Within each chunk, tokens are processed in parallel using standard attention; across chunks, all layers share the last-layer KV of completed history. This chunk-based design balances training parallelism and state recurrence. We additionally retain a small window of recent layer-specific KV through learned blending, which smooths the transition from local Transformer computation to shared long-term history.

Across language modeling, 3D novel-view synthesis, and video generation, LastKV delivers competitive or improved performance over standard Transformers while using a substantially more compact long-term KV representation. It improves long-context evaluation at both 124M and 760M scales, achieves higher reconstruction quality on Objaverse and DL3DV, and maintains strong downstream performance. These results suggest that a shared deep history can preserve the information needed for future computation while substantially reducing redundant KV storage across layers.

Beyond memory efficiency, recurrent feedback is an important source of the gains of LastKV. Ablations show that enabling feedback from deep historical representations improves performance during

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

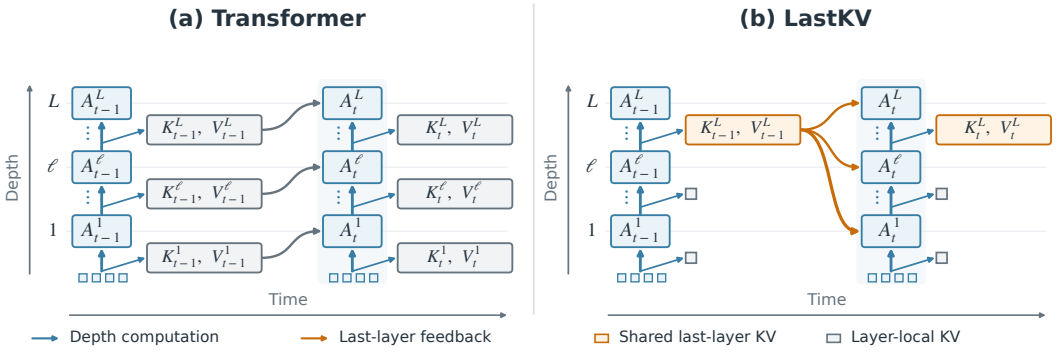


Figure 1: **LastKV allows recurrence across time and depth. (a) Standard Transformer:** At time step t, each layer ℓ computes its attention readout A_t^ℓ by retrieving the historical keys and values $K_{t-1}^\ell, V_{t-1}^\ell$ (gray arrows) of previous tokens at the same layer. **(b) LastKV:** Every layer ℓ at step t instead attends directly to the shared last-layer states K_{t-1}^L, V_{t-1}^L from step $t-1$ (orange arrows), routing deep completed representations of past tokens back into shallower layers of future tokens.

sequential processing, while synthetic state-tracking experiments demonstrate the benefit of repeatedly refining state over time. Our gradient analysis further shows that future supervision propagates through the shared last-layer KV into earlier computation, exhibiting BPTT-like credit assignment across both time and depth. Together, these results highlight a key property of LastKV: memory sharing and recurrence reinforce each other—a compact representation of the past also serves as a deep recurrent state for future computation.

2 Related Work

KV-cache reduction. Prior work reduces KV memory either across tokens or across layers. SnapKV (Li et al., 2024) selects a subset of prompt KV positions for each attention head, reducing the number of stored tokens. Cross-Layer Attention (Brandon et al., 2024) instead shares KV between neighboring layers, while YOCO (Sun et al., 2024) lets higher layers reuse KV produced by lower layers. LastKV differs in both direction and scope: all layers reuse the last-layer KV of completed history. Thus, it shares KV across the full network depth while retaining all historical token positions, and simultaneously feeds deep representations back into future shallow layers and creates recurrence across both time and depth.

Recurrent and feedback Transformers. Transformer-XL (Dai et al., 2019) caches layer-specific hidden states from previous segments as detached context, extending the receptive field without propagating gradients across segments. RMT (Bulatov et al., 2022) instead introduces dedicated memory tokens that are passed between segments and trained with BPTT, while AutoCompressors (Chevalier et al., 2023) compress previous segments into summary vectors used as soft prompts. Unlike these methods, LastKV introduces no additional memory tokens or compressed summaries: it directly reuses the ordinary last-layer KV of all historical tokens.

Feedback Transformer (Fan et al., 2020) similarly exposes high-level past representations to future shallow layers, but performs feedback token-by-token using a shared mixture of layer representations, limiting parallelism. Staircase Attention (Ju et al., 2021) instead repeatedly processes token states through a shared Transformer, with Cached Staircase retaining older states only as attention context. Block-Recurrent Transformers (Hutchins et al., 2022) maintain dedicated recurrent state vectors; their Block-Feedback variant broadcasts these states to all layers through additional cross-attention. LastKV achieves deep-to-shallow feedback using the existing KV cache itself and applies recurrence at the chunk level, achieving a balance between state recurrence and training parallelism.

3 Preliminaries

Consider an L-layer autoregressive Transformer operating on a sequence $x = (x_1, \ldots, x_T)$. At layer ℓ, token x_t is projected into a query, key, and value,

$$(q_t^\ell, k_t^\ell, v_t^\ell) = \mathrm{Proj}^\ell(h_t^{\ell-1}), \tag{1}$$

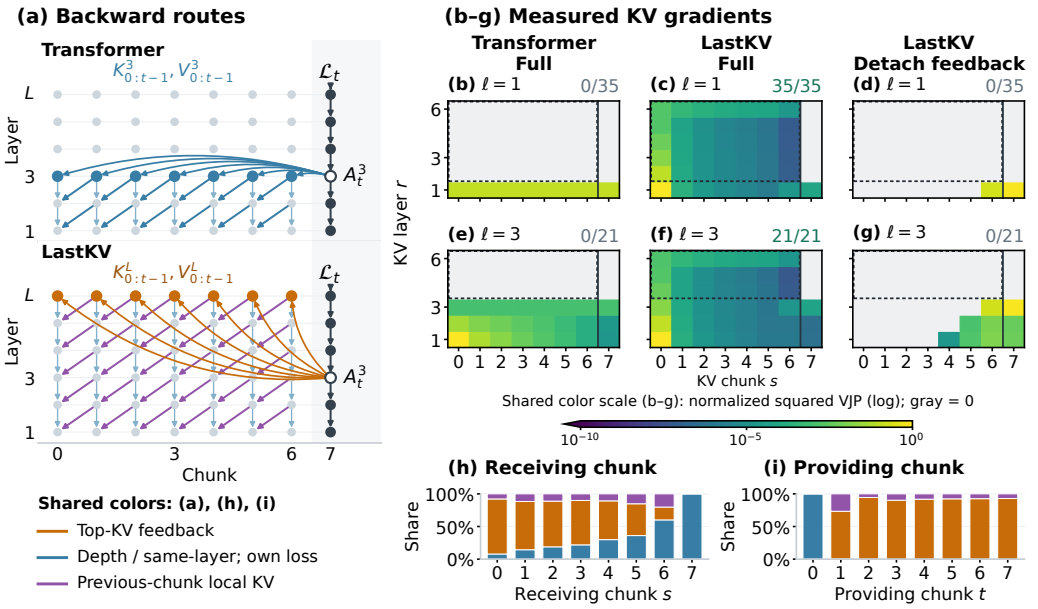


Figure 2: **Recurrence and gradient flow in LastKV.** (a) Backward paths from A_t^3—the output of layer-3 attention for chunk t—to historical KV: a standard Transformer reaches only same- or lower-depth history, while LastKV also reaches last-layer states $r = L$ through feedback. (b–d) VJP magnitudes from A_7^1 to historical KV for Transformer, LastKV, and LastKV with detached feedback (uses the same historical last-layer KV in the forward pass, but applies stop-gradient to it); (e–g) show the same comparison for A_7^3. Only LastKV with differentiable feedback exhibits nonzero gradients to higher-layer historical KV. (h) Decomposition of the gradients received by each historical chunk s: blue denotes supervision from the chunk's own loss through standard depth/same-layer paths, orange denotes feedback through the shared last-layer KV, and purple denotes the previous-chunk layer-local KV path. (i) The same decomposition from the perspective of the loss-providing chunk t, showing how supervision from later chunks propagates backward to earlier chunks. Together, these results show that the shared last-layer KV acts as a differentiable recurrent state, enabling BPTT-like credit assignment across both time and depth. Please refer to Appendix A for more details.

and attends to the current and previous tokens at the same layer:

$$A_t^\ell = \mathrm{Attn}\left(q_t^\ell, k_{\le t}^\ell, v_{\le t}^\ell\right). \tag{2}$$

Self-attention provides direct access to the past while enabling parallel training under a causal mask. However, its layerwise KV organization introduces two limitations:

Memory cost increasing linearly with model depth. Each layer stores its own KV states for all past tokens, resulting in a cache size of $\mathcal{O}(LTd_{\mathrm{KV}})$. Thus, the same sequence positions are stored independently across depth, making KV memory increasingly expensive for long contexts.

Lack of recurrence. In a standard transformer, each layer ℓ only retrieves historical states from the same layer, $(k_{<t}^\ell, v_{<t}^\ell)$. Although deeper representations of past tokens have already been computed, they are not fed back into shallower layers processing future tokens. Consequently, information moves forward through the network depth only once, bounding the number of sequential transformations by the network depth L.

These two limitations motivate LastKV: we store the deepest KV representation of completed history once and feed it back to all future layers, simultaneously reducing KV storage and introducing recurrence across time and depth.

4 Method

LastKV replaces layer-specific KV caches with a shared cache formed from the last-layer KV of previously processed tokens. This simple change simultaneously reduces redundant KV storage and introduces recurrent feedback: deeply processed representations of the past are fed back into future computation. We first describe this core mechanism (Section 4.1), then show how chunking enables

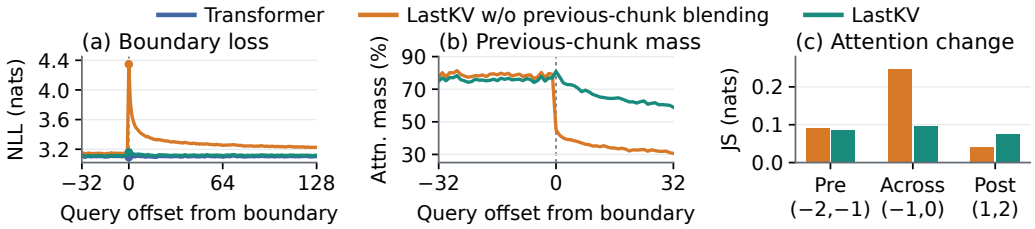


Figure 3: **Previous-chunk blending mitigates chunk-boundary artifacts.** (a) Next-token NLL aligned at chunk boundaries. LastKV without blending exhibits an obvious boundary spike, which is substantially reduced with blending. (b) Attention mass assigned to the immediately preceding chunk around the boundary; without blending, the abrupt switch from layer-specific to last-layer KV produces a sharper change in attention. (c) Jensen–Shannon divergence between the attention distributions of adjacent queries before, across, and after the boundary, showing that blending reduces the across-boundary discontinuity.

efficient parallel training, and introduce previous-chunk blending to smooth transitions between local and shared representations (Section 4.2).

4.1 Last-Layer KV Cache

The core idea of LastKV is to represent previously processed tokens using only their deepest KV representations. In a standard Transformer, layer ℓ attends to its own historical KV, $(k^{\ell}_{<t}, v^{\ell}_{<t})$. Instead, LastKV allows every layer to attend to the shared last-layer history, $(k^{L}_{<t}, v^{L}_{<t})$:

$$A^{\ell}_{t} = \mathrm{Attn}\big(q^{\ell}_{t}, [k^{L}_{<t};\, k^{\ell}_{t}], [v^{L}_{<t};\, v^{\ell}_{t}]\big)\,. \tag{3}$$

Here, brackets denote concatenation along the sequence dimension. All other operations, including the QKV projections, residual connections, and feed-forward layers, remain unchanged.

After token x_t is processed through all L layers, its last-layer KV (k^{L}_{t}, v^{L}_{t}) is added to the shared history and becomes directly accessible to every layer processing future tokens.

4.2 Chunk-Parallel Processing with Previous-Chunk Blending

The token-wise formulation in Equation 3 introduces a sequential dependency: the last-layer KV of token t must be computed before it can be consumed by future tokens. To retain Transformer-style parallelism, we apply the recurrence at the *chunk* level.

We partition the sequence into chunks of C tokens and denote the t-th chunk by $\mathcal{C}_t$. Let $Q^{\ell}_{\mathcal{C}_t}$, $K^{\ell}_{\mathcal{C}_t}$, and $V^{\ell}_{\mathcal{C}_t}$ denote the query, key, and value matrices of chunk $\mathcal{C}_t$ at layer ℓ, and define the shared last-layer history of all preceding chunks as

$$K^{L}_{\mathcal{C}_{<t}} = [K^{L}_{\mathcal{C}_0}; \ldots; K^{L}_{\mathcal{C}_{t-1}}], \qquad V^{L}_{\mathcal{C}_{<t}} = [V^{L}_{\mathcal{C}_0}; \ldots; V^{L}_{\mathcal{C}_{t-1}}]. \tag{4}$$

Within each chunk, all tokens are processed in parallel using standard causal attention, while the historical KV remains fixed:

$$A^{\ell}_{\mathcal{C}_t} = \mathrm{Attn}\big(Q^{\ell}_{\mathcal{C}_t},\, [K^{L}_{\mathcal{C}_{<t}}; K^{\ell}_{\mathcal{C}_t}],\, [V^{L}_{\mathcal{C}_{<t}}; V^{\ell}_{\mathcal{C}_t}]\big)\,. \tag{5}$$

Only after $\mathcal{C}_t$ has been processed through all L layers is its last-layer KV, $(K^{L}_{\mathcal{C}_t}, V^{L}_{\mathcal{C}_t})$, appended to the shared history and exposed to subsequent chunks. Thus, recurrence occurs across chunks rather than individual tokens. For a sequence of length T, this yields $\lceil T/C \rceil$ sequential recurrence steps instead of T token-wise steps, preserving parallel computation within each chunk.

Previous-chunk blending. At a chunk boundary, the representation used for nearby tokens changes abruptly. For tokens immediately before the boundary, layer ℓ attends to their layer-specific KV, while tokens immediately after the boundary access those same preceding positions through their last-layer KV. Thus, two adjacent query tokens on opposite sides of the boundary can observe different representations of nearly the same recent context. As shown in Figure 3, this sharp KV transition is associated with boundary artifacts, including elevated next-token loss and abrupt changes in attention patterns. To smooth this boundary transition, we retain the layer-specific KV of

the immediately preceding chunk and interpolate it with the corresponding last-layer representation. For chunk $\mathcal{C}_{t-1}$, we define

$$\widetilde{K}^{\ell}_{\mathcal{C}_{t-1}} = (1-\alpha^{\ell})K^{\ell}_{\mathcal{C}_{t-1}} + \alpha^{\ell}K^{L}_{\mathcal{C}_{t-1}}, \tag{6}$$

$$\widetilde{V}^{\ell}_{\mathcal{C}_{t-1}} = (1-\alpha^{\ell})V^{\ell}_{\mathcal{C}_{t-1}} + \alpha^{\ell}V^{L}_{\mathcal{C}_{t-1}}. \tag{7}$$

Here, α^{ℓ} is a learned sigmoid gate for each KV head, shared between keys and values and initialized to 0.5 (head indices are omitted for clarity).

With blending, chunk $\mathcal{C}_t$ at layer ℓ attends to the shared last-layer KV of older chunks, the blended KV of the immediately preceding chunk, and its own layer-specific KV for the current chunk:

$$A^{\ell}_{\mathcal{C}_t} = \mathrm{Attn}\Big(Q^{\ell}_{\mathcal{C}_t}, [K^{L}_{\mathcal{C}_{<t-1}}; \widetilde{K}^{\ell}_{\mathcal{C}_{t-1}}; K^{\ell}_{\mathcal{C}_t}], [V^{L}_{\mathcal{C}_{<t-1}}; \widetilde{V}^{\ell}_{\mathcal{C}_{t-1}}; V^{\ell}_{\mathcal{C}_t}]\Big). \tag{8}$$

Thus, only the most recent historical chunk retains layer-specific information, while older chunks are represented solely by their last-layer KV. The learned interpolation provides a gradual transition from layer-specific to shared representations near the chunk boundary. Blending substantially reduces boundary discontinuities in next-token loss and attention (Figure 3; protocols in Appendix B).

KV cache reduction. After processing $T \geq C$ tokens in complete chunks, LastKV retains the last-layer KV for all positions, together with layer-specific KV only for the most recent chunk. Its persistent KV cache therefore contains entries of

$$M_{\text{LastKV}} = T + (L-1)C, \tag{9}$$

compared with $M_{\text{Transformer}} = LT$ entries for a standard L-layer Transformer. These counts assume equal KV widths and deduplicated storage, excluding temporary buffers and training activations. Therefore, LastKV reduces the KV cache size by a factor of:

$$\frac{M_{\text{LastKV}}}{M_{\text{Transformer}}} = \frac{1}{L} + \left(1 - \frac{1}{L}\right)\frac{C}{T}, \tag{10}$$

which approaches $1/L$ as the number of chunks T/C increases. Thus, for long sequences, LastKV reduces persistent KV storage by approximately a factor of L.

Recurrence across both depth and time. In a standard Transformer, KV states remain layer-local: states produced at layer ℓ are reused only by the same layer at future positions. Thus, deep representations are never fed back into shallower layers for further processing.

In LastKV, once a chunk is processed through all L layers, its last-layer KV is added to the shared history and becomes accessible to every layer of subsequent chunks. A deep representation from one chunk can therefore be consumed by shallow layers of the next, transformed again through the network, and incorporated into the next shared state. This creates recurrence jointly across time and depth, allowing persistent information to undergo repeated rounds of computation as the sequence progresses. Its effective computational depth is therefore no longer strictly bounded by the network depth L, but can increase across recurrent chunk updates.

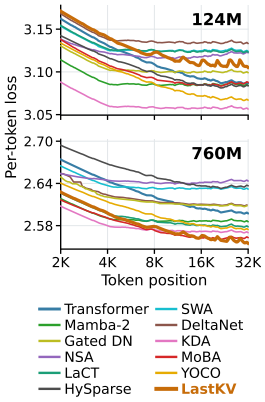
When the historical KV computation graph is retained, these state transitions are fully differentiable. Losses from later chunks backpropagate through the shared last-layer KV into the computation that produced earlier states, yielding BPTT-like credit assignment across both time and depth. Figure 2 probes this gradient connectivity in randomly initialized models.

5 Experiments

5.1 Language Modeling Pretraining

Long-Context Language Pretraining Under matched long-context pretraining, LastKV achieves the highest average RULER accuracy among the compared methods at both model scales and lowers tail-token loss relative to Transformer at the larger scale (Table 1). Appendix C gives the evaluation and KV configuration details.

Short-Context Language Pretraining Under matched short-context pretraining, LastKV improves WikiText perplexity and mean zero-shot accuracy over Transformer and both adapted recurrent baselines at both model scales (Table 2). T^{2}MLR (Cai et al., 2026) and Full-bandwidth Transformer (FBT) (Wang et al., 2026) were developed as recurrent Transformers without targeting KV-cache reduction; here we adapt both to retain only last-layer historical keys and values. Appendix D details the training and evaluation protocols.



	124M						760M					
	PTL ↓	RULER ↑					PTL ↓	RULER ↑				
Method	30k+	4k	8k	16k	32k	avg.	30k+	4k	8k	16k	32k	avg.
Transformer												
Transformer (Vaswani et al., 2017)	3.0869	23.82	20.24	13.10	8.93	16.52	2.598	41.7	38.0	33.4	23.6	34.2
SWA (Beltagy et al., 2020)	3.124	26.9	14.7	9.9	6.4	14.5	2.633	45.2	22.6	12.7	8.6	22.3
Linear Attention												
Mamba-2 (Dao & Gu, 2024)	3.085	24.2	13.9	8.3	5.4	13.0	2.586	**48.4**	25.5	12.6	7.6	23.5
DeltaNet (Yang et al., 2024)	3.134	22.5	11.2	7.6	5.4	11.7	2.609	43.4	20.6	11.8	7.3	20.8
Gated DeltaNet (Yang et al., 2025)	3.099	23.8	12.9	8.3	4.3	12.3	2.608	40.4	23.1	13.7	8.8	21.5
Kimi Delta Attention (Kimi Team, 2025)	**3.057**	20.9	13.0	8.4	6.3	12.1	2.571	43.5	24.2	14.7	9.0	22.8
Sparse Attention												
NSA (Yuan et al., 2025)	3.1220	19.26	11.26	5.18	3.11	9.70	2.644	25.9	12.7	6.8	3.9	12.3
MoBA (Lu et al., 2025)	3.0872	22.70	17.95	11.44	6.71	14.70	2.563	45.4	**38.9**	32.0	20.7	34.2
Test-Time Training												
LaCT (Zhang et al., 2025)	3.123	13.8	11.4	5.9	2.9	8.5	2.579	43.7	24.0	14.3	10.9	23.2
Cross-Layer Share												
YOCO (Sun et al., 2024)	3.0672	22.70	17.95	11.40	8.53	15.14	2.575	35.8	28.1	23.2	18.5	26.4
HySparse (Gao et al., 2026)	3.0835	25.97	22.37	**17.71**	8.90	18.74	2.636	40.2	29.6	27.3	23.8	30.2
LastKV	3.1067	**26.99**	**23.14**	16.09	**13.97**	**20.05**	**2.556**	39.2	37.4	**33.4**	**27.0**	**34.3**

Table 1: Long-context pretraining. PTL is the 30K+ tail loss (↓); RULER accuracy is reported at each length and averaged in *avg.* (↑). Best scores are bold, selected before rounding; dashes indicate results not included in this comparison. Left: per-token loss by position within the 32K sequence (log-scaled axis, positions below 2K omitted), 124M top and 760M bottom, one curve per method.

Table 2: Short-context pretraining at 20 TPP on H100 GPUs: 124M (left) and 760M (right). PPL is WikiText word perplexity (↓); downstream scores are full-test zero-shot accuracies (↑), with normalized accuracy for ARC-E, ARC-C, and HellaSwag. Shaded *avg.* columns equally average ten accuracies, excluding PPL; LastKV averages are bold.

	124M												**760M**											
	PPL	Language			Knowledge			Commonsense Reading					PPL	Language			Knowledge			Commonsense Reading				
Method	↓	LAMBADA	HellaSwag	WinoGrande	ARC-E	ARC-C	MMLU	PIQA	SIQA	COPA	BoolQ	avg.	↓	LAMBADA	HellaSwag	WinoGrande	ARC-E	ARC-C	MMLU	PIQA	SIQA	COPA	BoolQ	avg.
Transformer																								
Transformer (Vaswani et al., 2017)	54.725	22.2	31.4	50.8	43.6	25.9	24.9	63.3	36.9	55.0	60.9	41.5	26.565	35.7	46.4	52.1	56.9	28.9	25.3	71.8	38.6	69.0	60.6	48.5
SWA512 (Beltagy et al., 2020)	60.182	22.5	31.6	51.9	44.7	25.3	24.8	64.2	37.2	62.0	59.6	42.4	28.842	35.5	47.3	53.2	58.7	33.2	26.3	72.0	38.5	70.0	58.2	49.3
Linear Attention																								
Mamba-2 (Dao & Gu, 2024)	54.279	20.8	32.3	50.1	45.6	26.8	25.7	65.8	36.4	57.0	58.7	41.9	25.663	36.0	49.2	52.8	61.2	32.9	26.7	73.4	39.2	75.0	55.7	50.2
DeltaNet (Yang et al., 2024)	58.343	17.4	30.9	51.6	45.1	24.9	23.9	64.4	36.8	58.0	59.3	41.2	27.067	34.5	46.8	52.8	63.0	32.3	26.7	71.8	39.7	70.0	53.6	49.1
Sparse Attention																								
NSA (Yuan et al., 2025)	61.191	22.9	32.2	52.4	45.3	26.0	24.1	66.3	37.2	56.0	60.1	42.3	35.850	32.3	45.4	52.5	60.8	31.1	24.4	72.1	39.5	72.0	59.0	48.9
MoBA (Lu et al., 2025)	54.718	21.8	31.6	50.3	44.2	25.3	23.9	66.1	36.9	59.0	58.1	41.7	–	–	–	–	–	–	–	–	–	–	–	–
Cross-Layer Share																								
YOCO (Sun et al., 2024)	52.205	23.2	31.9	50.9	44.5	25.4	23.7	65.0	37.7	58.0	58.7	41.9	26.666	37.7	46.7	51.7	58.2	31.4	27.5	72.4	39.8	68.0	55.8	48.9
HySparse (Gao et al., 2026)	53.737	21.6	31.8	52.0	45.7	26.1	24.4	64.1	36.4	62.0	57.7	42.2	25.841	37.6	47.2	51.6	60.1	30.1	25.8	71.3	38.6	73.0	58.2	49.4
Recurrent style cross-layer share (from top to bottom)																								
Jacobi recurrence																								
T^2MLR-LastKV (Cai et al., 2026)	57.841	18.8	30.7	49.8	42.7	24.0	25.3	62.8	36.7	54.0	61.7	40.6	28.071	34.1	44.7	53.5	57.8	32.3	26.8	70.6	38.2	67.0	58.0	48.3
FBT-LastKV (Wang et al., 2026)	57.205	18.7	31.2	50.8	43.5	24.3	25.6	63.2	36.5	60.0	55.2	40.9	29.741	32.7	43.0	52.2	55.9	29.8	26.7	69.5	37.2	68.0	54.3	46.9
Chunk recurrence																								
LastKV	52.642	23.1	31.7	51.6	44.6	26.3	24.9	65.1	36.9	61.0	60.2	**42.5**	25.912	37.6	47.8	54.1	58.8	31.3	25.8	70.8	39.5	75.0	57.8	**49.9**

5.2 State Tracking and Memory Retention

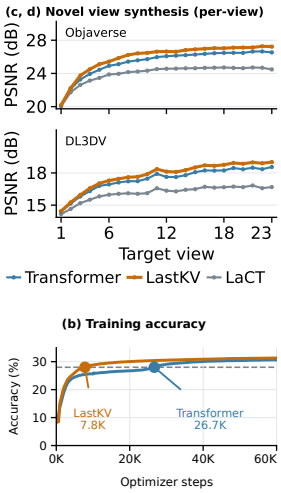
The synthetic suite evaluates five state-tracking and two memory-retention task families at 2K context (Table 3; panel a of Figure 4). LastKV achieves the best or tied-best performance on five of seven task families and numerically outperforms Transformer on all seven, supporting shared historical KV as a recurrent state for these tasks. Appendix E details the tasks, training, and evaluation. The training curve in Figure 4b also shows LastKV reaching the marked target accuracy in fewer optimizer steps.

5.3 Novel View Synthesis Pretraining

Figure 4e compares aggregate PSNR on Objaverse (GSO evaluation) and DL3DV; panels c,d report per-view PSNR. LastKV improves aggregate PSNR over Transformer on both datasets. Appendix F details training and evaluation.

(a) State tracking and memory retention

Method	State tracking: Fixed FSA	Mode switch	Gen. mod-sum	Group word	Perm. comp.	Memory retention: MQAR	Select. copy
Recurrent Neural Network							
RNN	33.0	50.6	2.0	19.3	12.7	0.2	0.2
Transformer							
Transformer	40.7	78.8	11.4	25.3	57.7	68.4	99.9
SWA	40.3	65.8	12.9	38.8	46.2	37.2	71.2
Linear Attention							
Mamba-2	40.3	73.0	5.9	**80.0**	43.8	100.0	0.5
DeltaNet	33.3	72.2	5.1	48.9	55.6	100.0	84.8
Gated DeltaNet	38.8	73.8	4.2	59.0	56.6	100.0	80.7
Kimi Delta Attention	42.2	75.7	3.8	49.7	57.9	100.0	92.8
Test-Time Training							
LaCT	42.1	88.9	**25.6**	27.6	59.5	100.0	99.8
Cross-Layer Share							
LastKV	**43.2**	**92.6**	21.3	27.6	**59.8**	**100.0**	**100.0**



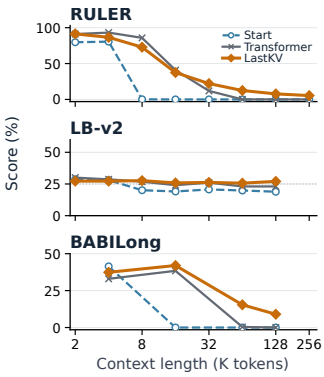
(e) Novel view synthesis

Method	Objaverse	DL3DV
Attention		
Transformer	25.36	17.41
Linear attention		
DeltaNet	23.94	15.82
Gated DeltaNet	24.32	16.05
Test-time training		
LaCT	23.99	16.14
Cross-Layer Share		
LastKV	**25.94**†	**17.80**

(f) Video generation

Size	Model	Qual.	Sem.	Total	Votes
1.3B	Transformer	83.1	76.5	81.8	85
	LastKV	**83.5**	**78.0**	**82.4**	**355**
14B	Transformer	82.2	72.7	80.3	114
	LastKV	**83.0**	**76.4**	**81.6**	326

Tables 3–4 and Figure 4: **(a, b) State tracking and memory retention.** (a) Token accuracy (%, ↑): the first five task columns measure state tracking and the last two memory retention, averaging four and three settings per task, respectively. (b) Training target-token accuracy, digitized from the supplied training plot. **(c–e) Novel view synthesis.** (c, d) Per-view PSNR on Objaverse (GSO evaluation) and DL3DV, respectively; (e) aggregate PSNR (↑). †The Objaverse LastKV entry and curve use logged in-training evaluation with random view draws; other NVS results use fixed-view full-test evaluation. **(f) Video generation.** VBench quality, semantic, and total scores (%, ↑), plus user-study votes, after teacher-forced fine-tuning. Protocols are specified in the text; Appendix Table 8 gives the full VBench breakdown.



Architecture	Knowledge: MMLU	MMLU-Redux†	ARC-Challenge	Super GPQA	Truthful QA-mc2	Coding: MBPP	MBPP+	Human Eval	Human Eval+
Start	57.13	51.42	59.35	14.29	45.87	15.10	3.35	10.87	7.77
Transformer	70.24	59.36	**62.45**	**22.75**	50.35	49.70	**75.32**	36.46	30.39
LastKV	**70.48**	**65.52**	61.42	21.89	**52.23**	**50.60**	74.27	**50.30**	**44.63**

Architecture	STEM & Reasoning: BBH	Wino Grande	GPQA-Diamond	HLE (CoT)	GSM8K	MATH-500	Instruction Following: IFEval	IFBench	*Avg.*
Start	39.41	76.58	22.64	4.23	20.78	7.20	9.56	6.73	26.60
Transformer	**68.45**	**78.57**	27.63	**4.87**	**71.24**	56.70	10.41	10.72	46.21
LastKV	67.56	76.62	**32.45**	2.63	71.12	**59.70**	**13.73**	**11.31**	**48.62**

Table 5: Selected 8B continued-pretraining results (%, ↑). Start is step 0; Transformer and LastKV are checkpoints at step 18,432 and use the same CPT training corpus. Left: per-length RULER, LB-v2, and BABILong results; lines connect measured points. Right: 17 selected benchmarks. *Avg.* is their equal-weight mean. Bold marks the best reported value in each column, including ties at displayed precision. Protocols and caveats (MMLU-Redux †; restricted-test MBPP+) are in Appendix H.

5.4 Video Generation Fine-Tuning

We fine-tune 1.3B and 14B models with teacher forcing and evaluate them with VBench (Table 4; panel f of Figure 4). Evaluation uses 946 prompts with five samples each; checkpoints are taken at 20k and 10k iterations, respectively. LastKV improves the total score from 81.76 to 82.41 at 1.3B and from 80.29 to 81.64 at 14B. At 14B, quality increases from 82.17 to 82.95 and the semantic score increases from 72.74 to 76.40. We also invite 44 participants for a user study. Each user is given random 10 pairs of videos, where each pair of videos are generated by 1.3B (or 14B) transformer/LaskKV model finetuned with teacher forcing, to opt for the one they prefer. At 1.3B scale, LaskKV receives 355 votes compared to 85 received by transformer; while at 14B scale, LastKV receives 326 votes compared to 114 received by transformer. Results are shown in Table 4.

5.5 LLM Continued Pretraining

Continued pretraining markedly improves LastKV's length extrapolation and coding over the starting checkpoint (Table 5). With the same training corpus and 8K window, LastKV surpasses Trans-

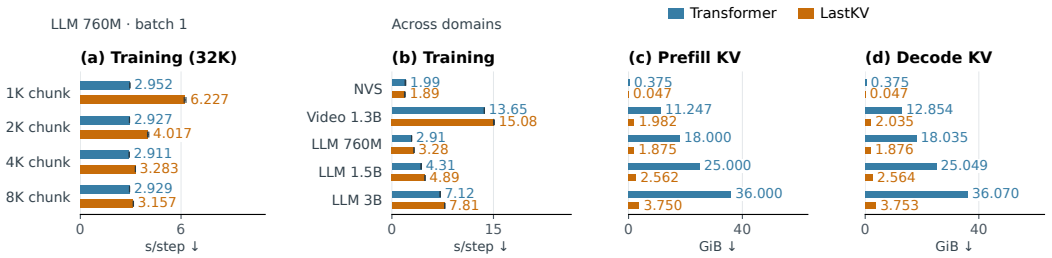
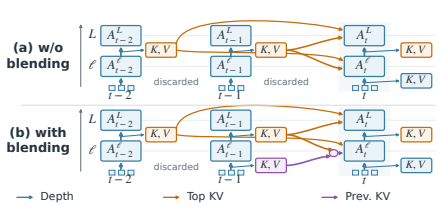
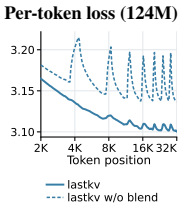


Figure 5: **Measured efficiency.** (a) 760M training at 32K, batch one; only LastKV varies chunk size. (b) Training across domains. (c,d) Physical self-attention KV after prefill and cached decoding. LLMs in (b–d) use 32K for training and 128K for inference, with C4K. Blue: Transformer; orange: LastKV. KV excludes weights/workspaces; whiskers span repetitions or block/scene medians.





Task	Metric	LastKV	w/o blend
LLM long	Loss ↓	3.102	3.141
	Avg. ↑	17.52	12.17
LLM short	PPL ↓	52.642	52.838
	Avg. ↑	42.54	42.33

Figure 6: **Previous-chunk blending.** Left: historical KV available at t, (a) without and (b) with blending (circles). Chunks $s \leq t-2$ retain only last-layer KV; earlier attention reads are omitted. Center: 124M per-token loss for LastKV (solid) and LastKV w/o blend (dashed) by token position (log-scaled axis, positions below 2K omitted, 768-token running mean); the periodic peaks without blending fall at the 4K chunk boundaries. Right: 124M long- and short-context comparisons using direct last-layer KV reuse.

former at far longer contexts and on most coding benchmarks, with a higher overall average. These coding gains align with its strong state-tracking results (Section 5.2): coding also requires reasoning about variables and program-state updates. Appendix H details the evaluation protocols and aggregation.

5.6 Efficiency

Figure 5 compares training time and physical KV storage after prefill and cached decoding on A100-SXM4-80GB GPUs. Each Transformer–LastKV pair uses matched hardware, input workloads, and precision. The LLM comparisons in panels (b–d) cover 760M, 1.5B, and 3B, using 32K context for training and 128K for inference, with a 4K chunk for LastKV. Both training methods enable activation checkpointing. NVS and video prefill encode observed views and supplied latent blocks, respectively; their decode KV is measured after target-view rendering and next-block generation. Appendix I specifies the phase boundaries, model initialization, repetitions, and absolute measurements.

LastKV substantially reduces physical KV storage in both inference phases while keeping training time close to Transformer in the main configurations. Larger chunks reduce 760M training time (Figure 5a). Physical KV storage counts shared allocations once and includes retained local buffers.

5.7 Ablation Studies

Previous-chunk blending. Blending improves language-modeling quality and downstream accuracy, with larger gains in long-context retrieval (Figure 6). Without blending, the previous chunk is accessed only through last-layer KV; with blending, its layer-specific KV are mixed in as described under Previous-chunk blending in Section 4.2. We report 32K-context tail loss and RULER average, and short-context WikiText PPL and ten-task average, all at 124M under the protocols in Appendix J.1, complementing the boundary diagnostics in Figure 3.

Chunk size. We vary $C/T \in \{1/32, 1/16, 1/8, 1/4\}$ in state tracking and 124M short- and long-context LLMs; at 32K context, these ratios correspond to $C = 1{,}024, 2{,}048, 4{,}096, 8{,}192$. Figure 7a,b reports the LLM evaluations; in (a), tail loss averages zero-based positions 30,720–32,766. The data, training-token budget, and last-layer source are fixed; previous-chunk blending is retained,

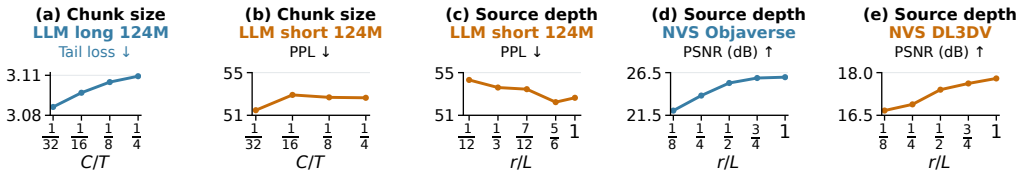
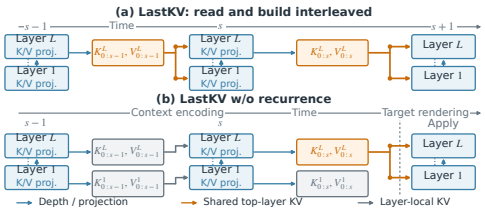


Figure 7: **Chunk-size and source-depth ablations.** (a,b) 124M long- and short-context LLM chunk-size sweeps; (c) 124M short-context LLM source-depth sweep; (d,e) NVS source depth on Objaverse and DL3DV. In (b,c), circles show WikiText PPL. C128 and source layers 0/3/9 use H200 runs; other short-context points use H100 runs. C512 is an archived control; C1024 and the last-layer reference reuse Table 2. In (a), circles show tail PTL over zero-based positions 30,720–32,766. Source depth is r/L, with one-based source layer r; the source-layer indices above are zero-based.



Task	Metric	LastKV	w/o recurrence
Video	Avg ↑	81.90	81.36
NVS (Obj./DL3DV)	PSNR ↑	25.97/17.80	25.31/17.32
LLM short	PPL ↓	52.64	53.22
	Avg ↑	42.54	42.47
LLM long	Loss ↓	3.10	3.10
	Avg ↑	17.52	9.43

Figure 8: **Recurrence ablation.** (a) In LastKV, each completed chunk immediately shares its last-layer KV with all layers of subsequent chunks, so deep historical representations recurrently participate in constructing future context. (b) The nonrecurrent control instead constructs the context with standard layer-local attention and shares the resulting last-layer KV only when processing the final target. Thus, both variants provide the target with last-layer historical KV, while only LastKV recurrently reuses it during context construction. **Right:** recurrent feedback consistently improves performance across video generation, novel-view synthesis, and short- and long-context language modeling. Video uses 1.3B models and LLMs use 124M models; PPL/Avg denotes WikiText perplexity/ten-task accuracy, and Loss/Avg denotes 30K+ tail loss/mean RULER.

so its window grows with C. At 32K context, $C = 1{,}024$ gives the lowest tail loss (3.0862). For short-context LLMs at $T = 4{,}096$, Figure 7b reports WikiText PPL, reusing the archived C512 control and the C1024 control from Table 2. Short-context PPL is lowest for the smallest tested chunk and varies little among the larger chunks.

KV source depth. We replace historical K_s^L, V_s^L with K_s^r, V_s^r, retaining all receiving layers and using the same source for previous-chunk blending (Figure 7c–e). Short-context PPL improves with deeper sources but rises slightly at the final layer. Deeper sources consistently improve NVS reconstruction on both datasets, with the last layer performing best.

Recurrent feedback. To isolate the effect of recurrence from using last-layer KV itself, we compare LastKV with a nonrecurrent control that exposes the target to the same last-layer historical KV but does not use it while constructing the context (Figure 8). Specifically, LastKV immediately feeds each completed chunk's last-layer KV back to subsequent chunks, whereas the control constructs the context using standard layer-local attention and shares last-layer KV only for the final target. Removing recurrent feedback consistently degrades performance across video generation, novel-view synthesis, and both short- and long-context language modeling (Figure 8).

6 Conclusion

We introduced LastKV, a novel method that shares the last-layer KV of historical tokens across all transformer layers. This both reduces redundant KV storage across depth and feeds deep historical representations back into future computation, creating recurrence across time and depth. We apply recurrence at the chunk level, preserving full parallelism within each chunk while introducing recurrent feedback across chunks, thereby retaining much of the training efficiency of standard Transformers. Across language modeling, state tracking, novel-view synthesis, and video generation, LastKV matches or improves standard Transformers and other state-of-the-art baselines.

AI USE STATEMENT

We used large language models to assist with revising and polishing the manuscript. The authors take responsibility for the final content of this paper, including its scientific claims and reported results.

REPRODUCIBILITY STATEMENT

Section 4 specifies the historical-KV construction, chunked recurrence, and previous-chunk blending, and Section 5 describes the comparisons and metrics. The appendices provide task definitions, training settings, and evaluation protocols. Appendices A and B detail the gradient and boundary diagnostics. Appendix I documents hardware, precision, activation checkpointing, workload definitions, phase boundaries, and repetitions for the efficiency measurements, distinguishing trained-checkpoint runs from runtime tests using randomly initialized models.

REFERENCES

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. URL https://arxiv.org/abs/2004.05150.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention, 2024. URL https://arxiv.org/abs/2405.12981.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/47e288629a6996a17ce50b90a056a0e1-Abstract-Conference.html.

Ziyang Cai, Xingyu Zhu, Yihe Dong, Yinghui He, and Sanjeev Arora. T^2MLR: Transformer with temporal middle-layer recurrence. *arXiv preprint arXiv:2607.15178*, 2026. URL https://arxiv.org/abs/2607.15178.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285/.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 10041–10071, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*, 2020. URL https://arxiv.org/abs/2002.09402.

Yizhao Gao, Jianyu Wei, Qihao Zhang, et al. HySparse: A hybrid sparse attention architecture with oracle token selection and KV cache sharing. *arXiv preprint arXiv:2602.03560*, 2026. URL https://arxiv.org/abs/2602.03560.

Ailin Huang, Ang Li, Aobo Kong, et al. Step 3.5 Flash: Open frontier-level intelligence with 11b active parameters. *arXiv preprint arXiv:2602.10604*, 2026. URL https://arxiv.org/abs/2602.10604.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 33248–33261, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/d6e0bbb9fc3f4c10950052ec2359355c-Abstract-Conference.html.

Da Ju, Stephen Roller, Sainbayar Sukhbaatar, and Jason Weston. Staircase attention for recurrent processing of sequences. *arXiv preprint arXiv:2106.04279*, 2021. URL https://arxiv.org/abs/2106.04279.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation, 2024. URL https://arxiv.org/abs/2404.14469.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, et al. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025. URL https://arxiv.org/abs/2502.13189.

William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You Only Cache Once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024. URL https://arxiv.org/abs/2405.05254.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Xi Wang, Ziyang Cai, Zheng Zhan, Harry Dong, Ying Fan, Gustavo de Rosa, Tim Pearce, and John Langford. Full-bandwidth transformer. *arXiv preprint arXiv:2608.08888*, 2026. URL https://arxiv.org/abs/2608.08888.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115491–115522, 2024. URL https://arxiv.org/abs/2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Jingyang Yuan, Huazuo Gao, Damai Dai, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Tianyuan Zhang, Sai Bi, Yicong Hong, et al. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025. URL https://arxiv.org/abs/2505.23884.

A GRADIENT PROBE PROTOCOL

Connectivity probes (Figure 2(b–g)). We compare Transformer and LastKV with six layers, hidden size 128, four attention heads, and vocabulary size 512. Seeds 42, 43, and 44 use matched common weights and identical 4,096-token random inputs within each seed. LastKV uses $C = 512$, top-layer-only long-term history, and the previous-chunk blend with gates initialized to 0.5. These diagnostics use random initialization with zero optimizer steps; they measure connectivity, not pretraining quality.

For each target layer, we project the last 32 attention-output positions onto a fixed random-sign direction to obtain a scalar probe. We differentiate it with respect to the K projection outputs before rotary encoding and the V projection outputs at every layer and token. Each cell sums squared vector–Jacobian products (VJPs) over K, V, channels, and a 512-token source bin. We normalize by the total over all source layers and positions separately for each target and seed, then average the three seeds. Panels (b–g) select targets 1 and 3. Their historical boxes contain $7 \times 5 = 35$ and $7 \times 3 = 21$ higher-layer source cells, respectively. Counts require nonzero values in all seeds, independently of the color scale's 10^{-10} lower display limit.

The feedback-detached control detaches both older top-layer history and the top-layer component of the previous-chunk blend, while retaining the local component. All forward logits match the full-graph version exactly in these runs. Ordinary Transformer final-loss gradients can also reach historical KV at every layer; the intermediate-layer probes isolate the additional dependencies introduced by LastKV.

Measurements use BF16 and the actual attention and feed-forward implementations, with immutable concatenation replacing writable cache storage for a reliable autograd reference. The native evaluation path with in-place cache reuse produced matching logits but inconsistent gradients. For the first seed, the native training path with activation recomputation matched the reference closely on embedding, first-layer K, and last-layer V parameter VJPs: all cosine similarities exceeded 0.99996, and the largest relative L_2 difference was 0.807% for $C = 512$. This component check does not establish full training equivalence. Retaining or recomputing historical activations is necessary for cross-chunk backpropagation; the inference-cache count in equation 10 is not a bound on training memory.

Receiver-side attribution (Figure 2(h)). We use the same model dimensions, chunk size, and three seeds for a separate LastKV probe with random initialization and random tokens. Each chunk loss $\mathcal{L}_s$ averages next-token cross entropy over its 512 positions; the objective is $\frac{1}{8}\sum_{s=0}^{7}\mathcal{L}_s$. Here we differentiate generated post-RoPE K_s^ℓ and projected V_s^ℓ using FP32 CPU reference kernels, unfused normalization and feed-forward operations, and immutable caches.

For each receiving chunk s, let g_s collect the gradients of all its generated keys and values. Differentiable aliases identify individual cache-consumer edges without changing logits. We sum signed adjoints from later consumers separately for top-layer history and the next chunk's local KV branch, then backpropagate each sum through the receiver's KV producers. This operation is performed separately for each s; causality prevents a path from leaving for an earlier chunk and returning. Together with the derivative of $\mathcal{L}_s/8$, these give

$$g_s = g_s^{\mathrm{own}} + g_s^{\mathrm{top}} + g_s^{\mathrm{prev}}, \qquad g_s^{\mathrm{future}} = g_s^{\mathrm{top}} + g_s^{\mathrm{prev}}. \tag{11}$$

The future gradient is also computed independently from $\frac{1}{8}\sum_{t>s}\mathcal{L}_t$. Both identities hold with maximum per-node relative L_2 error below 7×10^{-7}; alias insertion leaves logits unchanged, and earlier losses have exactly zero gradients to later chunks.

Colors follow Figure 2(a), but attribution groups supervision rather than counting schematic arrows: blue is the own-loss contribution, and orange/purple retain their boundary-entry label while propagating within the receiver. The purple local branch includes layer L, where local and source KV alias the same tensors; their two gate-weighted uses remain separate branches. For each seed and receiver, bars divide each component's squared norm by the sum of the three squared norms over all layers, K/V, and positions, then average seeds. These shares exclude cross-component inner products and are not additive fractions of $\|g_s\|^2$. The own-loss share increases from 7.9% at $s = 0$ to 60.1% at $s = 6$ and 100% at $s = 7$ in this probe; the values characterize random-initialization routing, not trained-model performance.

Provider-side attribution (Figure 2(i)). Here a providing chunk t supplies the loss $\mathcal{L}_t/8$, rather than the cached keys and values. For this loss alone, we repeat the boundary-entry decomposition separately at each receiver $s \le t$. Write the resulting vectors as $g_{t\to s}^{\rho}$ for $\rho \in \{\mathrm{own}, \mathrm{top}, \mathrm{prev}\}$. The own component is nonzero only at $s = t$; the other components enter earlier receivers through their top-layer or local KV and include propagation through those receivers' producers. A provider bar normalizes $\sum_{s\le t}\|g_{t\to s}^{\rho}\|^2$ across ρ, then averages seeds. In contrast, receiver bars square after summing the signed contributions over loss chunks. Thus the two panels summarize the same branch definitions but are not transposes of a matrix of squared norms; both exclude cross-component inner

products. The first provider has only its own chunk to supervise, while later providers can also train historical KV. Each loss's three signed components reconstruct its full gradient; summing them over losses also recovers the independently computed receiver components, with maximum per-node relative L_2 errors below 7×10^{-7}. The rerun reproduces the original receiver, per-loss, and total squared norms exactly.

B PREVIOUS-CHUNK BLENDING ANALYSIS

We expand on the Previous-chunk blending discussion in Section 4.2 and Figure 3. LastKV w/o previous-chunk blending uses top-layer KV for all completed chunks; LastKV additionally applies the blending in equations 6–7. Both use *reusekv*, with 12 layers, hidden size 768, 12 attention heads, RoPE base 10^6, and no attention-output gate. We evaluate step-10,240 checkpoints in the 124M model tier. Their native sequence length is $T = 32{,}768$ with $C = 4{,}096$, giving eight chunks and seven internal boundaries. These are separately trained checkpoints, not an intervention on one fixed model.

KV change and storage. Rearranging equations 6–7 makes the attenuation of the immediate KV change explicit:

$$\widetilde{K}^{\ell}_{t-1} - K^{\ell}_{t-1} = \alpha_{\ell}\big(K^{L}_{t-1} - K^{\ell}_{t-1}\big), \qquad \widetilde{V}^{\ell}_{t-1} - V^{\ell}_{t-1} = \alpha_{\ell}\big(V^{L}_{t-1} - V^{\ell}_{t-1}\big). \tag{12}$$

Queries and attention weights can still change, so this does not guarantee continuous attention.

For equation 10, we count equal-width, deduplicated keys and values at a completed chunk boundary, with $T \geq C$ divisible by C. The shared last-layer history contains T positions, and the recent window adds $(L-1)C$ positions from the remaining layers. Current-chunk buffers and training activations are excluded. Cache savings do not change dense attention's quadratic dependence on T; throughput also depends on kernels and cache layout.

Archived loss measurements. The original Books3 evaluations process 81,920 windows per model. Let $\bar{n}_p$ denote the recorded mean negative log-likelihood (NLL) at query/logit position p, predicting token $p+1$. A native boundary is $b = kC$, $k = 1, \ldots, 7$; query $b-1$ predicts the boundary token.

The first-query and first-32-query local excesses are

$$\Delta_1(b) = \bar{n}_b - \frac{1}{32}\sum_{i=1}^{32} \bar{n}_{b-i}, \qquad \Delta_{32}(b) = \frac{1}{32}\sum_{i=0}^{31} \bar{n}_{b+i} - \frac{1}{32}\sum_{i=1}^{32} \bar{n}_{b-i}. \tag{13}$$

Table 6 averages each excess over the seven boundaries. Full-sequence NLL averages all 32,767 valid query positions. LastKV w/o previous-chunk blending and LastKV use the same recorded Books3 data glob and nominal evaluation protocol. The Transformer reference uses a different Books3 copy whose byte identity and ordering have not been verified.

Model	Mean NLL	Δ_1	Δ_{32}
Transformer	3.1083	-0.0088	-0.0022
LastKV w/o previous-chunk blending	3.1643	$+1.2024$	$+0.2773$
LastKV	3.1206	$+0.0463$	$+0.0142$

Table 6: Archived Books3 NLL and boundary-local excess, in nats. These descriptive contrasts are not independent-seed estimates or isolated causal effects of blending.

Matched attention diagnostic. Separately, we run LastKV w/o previous-chunk blending and LastKV on four identical, consecutive 32K windows from the official `full_books3` source. We tokenize documents from the first Arrow shard in stored order with the evaluation tokenizer, default BOS insertion, and no EOS insertion, then concatenate and split into windows. This subset involves two books; their boundary lies outside every inspected chunk boundary's ± 32-position neighborhood. It provides a mechanism diagnostic, not corpus-wide attention statistics. Transformer attention is not measured.

At each boundary b, we fix the key positions $[b-C, b)$ and sum attention probabilities assigned to them by queries around b. This keeps the same target chunk when crossing the boundary. We

Table 7: KV construction comparison for LastKV, extending Table 1 with full-context mean PTL in addition to the 30k+ tail mean (both lower is better). *reusekv* keeps the last layer's keys and values verbatim, *sharekv* passes the retained feature through one projection pair shared by all layers, and *perlayerkv* lets each layer apply its own key and value projections to it. The *perlayerkv* row at 124M and *sharekv* row at 760M are used in Table 1. The best available value in each metric column is in bold, selected before rounding; dashes denote unavailable costs.

	Cost		124M							760M						
	Compute	Memory	PTL ↓		RULER ↑					PTL ↓		RULER ↑				
Method	↓	↓	mean	30k+	4k	8k	16k	32k	avg.	mean	30k+	4k	8k	16k	32k	avg.
LastKV																
reusekv	–	–	3.121	3.102	**27.3**	21.9	11.7	9.2	17.5	**2.575**	**2.553**	36.6	35.4	27.9	22.8	30.7
sharekv	–	–	**3.112**	**3.092**	24.4	22.6	15.2	12.1	18.6	2.578	2.556	39.2	**37.4**	**33.4**	**27.0**	**34.3**
perlayerkv	–	–	3.125	3.107	27.0	**23.1**	**16.1**	**14.0**	**20.1**	2.582	2.560	**41.7**	36.3	30.8	22.6	32.8

average over heads, layers, windows, and boundaries. The resulting mass changes from 79.15% to 45.12% for LastKV w/o previous-chunk blending and from 76.84% to 80.93% for LastKV between queries $b-1$ and b. Probability mass does not measure the contribution of the corresponding value vectors to the attention output.

For distribution changes, we first average probabilities over heads, restrict both distributions to the common prefix $p_{\text{key}} < b-1$, and renormalize. We compute Jensen–Shannon divergence with natural logarithms separately for each window, layer, and boundary before averaging. The query pairs $(b-2, b-1)$, $(b-1, b)$, and $(b+1, b+2)$ yield respectively $(0.0913, 0.2462, 0.0409)$ nats for LastKV w/o previous-chunk blending and $(0.0862, 0.0973, 0.0746)$ for LastKV. All pairs use the same prefix, excluding changes caused solely by newly available key positions; adjacent queries still differ in content.

Instrumentation and reproducibility. We capture actual post-RoPE, post-blend Q/K tensors and reconstruct sampled probabilities in FP32 while retaining FlashAttention for model outputs. The observer leaves first-window hidden states and NLL exactly unchanged. Reconstructed probability–value products agree with sampled kernel outputs within 0.2% maximum relative L_2 error. Boundary queries within ± 32 are sampled individually; complete key distributions for ± 2 are stored in FP16. Repeated boundaries are not independent training seeds, and this four-window diagnostic must not be pooled with archived losses.

The original PTL records are Transformer, LastKV w/o previous-chunk blending, and LastKV. Attention inference uses commit `7ebfe91a`. These observations support reduced boundary disruption; they do not establish that blending alone causes the difference or eliminates chunk structure.

C Long-Context Pretraining by KV Construction

We pretrain 124M and 760M models from scratch with 32K context, using the same data and token budget across methods at each scale.

PTL averages valid next-token losses at zero-based positions 30,000–32,766 in a 32,768-token context. RULER is evaluated at 4K, 8K, 16K, and 32K, with equal weight across these four lengths in the reported average. The main LastKV row in Table 1 uses *perlayerkv* at 124M and *sharekv* at 760M; Table 7 compares all three KV constructions. Figure 9 illustrates their projection and sharing patterns.

D Short-Context Evaluation Protocol

Table 2 reports selected 124M and 760M methods at 20 tokens per parameter (TPP), using 8 H100 GPUs and per-GPU batch size 16. All runs use sequence length 4,096, 128 warmup updates, and seed 42. At 124M, the budget uses a shared 134.1M-parameter reference; gradient accumulation is 1: the global batch contains 128 sequences, or 524,288 tokens per update. Under the warmup–stable–decay (WSD) schedule, training ends at update 5,116 (2,682,257,408 tokens), with the final 512 updates decaying the learning rate from 10^{-3} to zero using $1-\sqrt{p}$, where $p \in [0,1]$ is progress through the decay phase.

At 760M, gradient accumulation is 6, giving a global batch of 768 sequences and 3,145,728 tokens per update. The shared reference has 777,856,512 parameters; training ends at update 4,946 (15,558,770,688 tokens), with 4,451 stable updates followed by 495 decay updates under the same schedule. The table reports selected H100 configurations. At both model scales, the table reports LastKV with chunk size 1024 and previous-chunk blending.

Model configurations. For the 124M cohort, Mamba-2, DeltaNet, MoBA, and YOCO use no sliding-window attention (SWA). DeltaNet enables Q/K/V short convolutions with kernel size 4. YOCO uses six gated-retention layers and six full shared cross-attention layers, with Transformer initialization: Linear/Embedding weights have normal standard deviation 0.02, and the embedding scale is 1. NSA retains its native local window of 16. HySparse uses the NSA-based configuration with SWA512 and a nominal 1:3 full-to-sparse pattern; forcing the final layer to full attention gives four full and eight sparse layers. LastKV uses chunk size 1024 with previous-chunk $(t-1)$ blending; our T^{2}MLR-LastKV and FBT-LastKV adapt T^{2}MLR (Cai et al., 2026) and Full-bandwidth Transformer (Wang et al., 2026) by retaining only last-layer historical keys and values.

Evaluation. Evaluation uses zero-shot prompts, the full evaluation splits, a maximum context of 4,096 tokens, batch size 1, and no KV cache. WikiText uses word perplexity; LAMBADA uses continuation accuracy. ARC-E, ARC-C, and HellaSwag use normalized accuracy, while the other seven downstream tasks use accuracy. The reported average equally weights the ten accuracy scores and excludes WikiText perplexity. Figure 7 reuses the C512/C1024 control perplexities, and Figure 8 reuses the C1024 control average under this protocol. The additional 124M ablation models are trained on eight GPUs with the same per-GPU training batch size, training budget, seed, and evaluation protocol. C128, source layers 0/3/9, and the three unblended KV constructions use H200 GPUs; C256, source layer 6, and the no-recurrence variant use H100 GPUs. Source-layer indices are zero-based. These ablations are reported separately from the main results in Table 2.

E SYNTHETIC PRETRAIN

Synthetic tasks. We evaluate five state-tracking task families and two memory-retention families. All tasks are generated online as causal language-modeling sequences, with input/action tokens and output-state tokens placed in semantically distinct vocabulary ranges.

Fixed FSA. Each task instance uses a fixed random transition function $\delta : [S] \times [A] \to [S]$, shared across training and evaluation. Given an initial state s_0 and a sequence of uniformly sampled actions a_t, the target after every action is

$$s_{t+1} = \delta(s_t, a_t).$$

The initial state is included as the first input token, and every subsequent transition is supervised. We average results over $(S, A) \in \{(32, 8), (64, 8), (64, 32), (96, 8)\}$. The transition table is generated with seed 42.

Mode Switching. The model maintains a binary latent state. The input alphabet contains SET-A, SET-B, SWITCH, and four semantically equivalent filler tokens. The state update is

$$s_{t+1} = \begin{cases} 0, & x_t = \text{SET-A}, \\ 1, & x_t = \text{SET-B}, \\ 1 - s_t, & x_t = \text{SWITCH}, \\ s_t, & x_t \text{ is a filler token.} \end{cases}$$

We supervise the state after every token. We define H as the expected number of tokens between explicit set events, assigning probability $1/(2H)$ to each of SET-A and SET-B. The evaluated settings are

$$(H, p_{\text{switch}}) \in \{(256, 0.15), (4096, 0.15), (1024, 0.05), (1024, 0.30)\}.$$

Generalized Mod-Sum. Inputs are sampled uniformly from $x_t \in \{0, \ldots, K-1\}$, and the latent state is the running modular sum

$$s_t = (s_{t-1} + x_t) \bmod N, \qquad s_0 = 0.$$

The model predicts the updated residue after every input token. Input symbols and residue labels use disjoint token ranges, so, for example, an input value of zero is distinct from an output state of zero. We average over

$$(N, K) \in \{(64, 4), (64, 16), (128, 16), (128, 32)\}.$$

Finite-Group Word Problem. The latent state is a running product in the non-abelian solvable group $A_4 \times \mathbb{Z}_5$, which has 60 elements. Each non-reset input is sampled uniformly from all 60 group elements, and the state evolves as

$$g_t = g_{t-1}x_t, \qquad g_0 = e.$$

A dedicated RESET token periodically maps the state back to the identity e. We supervise the complete 60-way group state after every input, including reset tokens. Difficulty is controlled by the number B of group multiplications between resets, with

$$B \in \{32, 64, 128, 256\}.$$

Permutation Composition. The model tracks $R = 16$ permutation-valued registers in the symmetric group S_N. In each round, $U = 16$ distinct registers are selected. Each selected register receives D updates of the form

$$\pi_r \leftarrow \gamma_j \circ \pi_r,$$

where γ_j is chosen from a fixed set of G non-commuting generators. The first two generators are an N-cycle and the transposition $(0\,1)$; additional generators are fixed deterministically. After the updates, every selected register is queried with a randomly sampled element e, and the model must output $\pi_r(e)$. Only query outputs are supervised. We evaluate

$$\begin{aligned}(N, R, U, D, G) \in \{ \\ (9, 16, 16, 4, 2),\ (9, 16, 16, 8, 2), \\ (7, 16, 16, 4, 8),\ (7, 16, 16, 8, 8)\}.\end{aligned}$$

Vanilla MQAR. A sequence begins with a support set of K distinct key–value bindings,

$$(k_1, v_1), \ldots, (k_K, v_K),$$

where keys and values are drawn without replacement from disjoint vocabulary ranges of size V. The remainder of the sequence contains repeatedly shuffled key queries, and the model predicts the corresponding value. The current answer is never visible to the logit predicting it, although gold answers are teacher-forced and may therefore be visible to later queries. Loss is applied only at answer positions. We use

$$(V, K) \in \{(512, 64), (512, 72), (512, 96)\}.$$

Target-Free Selective Copy. The source region contains random symbols from a vocabulary of size V. We select K source positions uniformly without replacement and mark them using a disjoint marked-token range. After a separator, the model must reproduce the selected symbols in their original source order. Every answer slot in the input contains the same placeholder token, rather than a teacher-forced gold answer, preventing earlier targets from serving as an autoregressive retrieval scaffold. Loss is applied only to the K answer positions. We evaluate

$$(V, K) \in \{(512, 64), (512, 128), (1024, 128)\}.$$

Training and evaluation settings. All experiments use sequences of length 2,048 and train for 60,000 optimization steps on eight GPUs with a per-GPU batch size of 24, giving a global batch size of 192 sequences. We use AdamW with learning rate 3×10^{-4}, weight decay 0.1, and a 1,000-step linear warmup from zero followed by a constant learning rate. Model initialization, training data, and evaluation data are controlled by seed 42. Dense state-tracking tasks apply causal cross-entropy supervision after every transition; Permutation Composition and both memory tasks supervise only their query or answer positions. Evaluation is performed at the training length of 2,048 tokens. Each reported table entry is the arithmetic mean of token accuracy over the settings listed for that task family.

F NOVEL VIEW SYNTHESIS CONFIGURATION

Training. Transformer and LastKV in Figure 4c–e share an 8-layer, width-512 backbone, data, and training budget (global batch 128; 24,476 object-level and 20,046 scene-level steps). DeltaNet and Gated DeltaNet use the same data and budget, without short convolutions and parameter-matched to LaCT.

Evaluation. Fixed-view evaluation uses 24 views per scene with teacher-forced sequential conditioning, averaging PSNR over target views 1–23 and then over all 1,019 GSO objects or 140 DL3DV scenes. Transformer, LaCT, and DL3DV LastKV use this protocol for their per-view curves in Figure 4c,d. The Objaverse LastKV entry and curve instead use logged in-training evaluation at step 24,000 with random view draws.

G Video Generation Configuration

The teacher-forcing Transformer baseline in Table 4 is the `tf_flex` run at both scales. The LastKV rows use `akv2542` with direct KV reuse at both scales. The corresponding checkpoints are at 20k and 10k iterations, respectively.

Table 8 reports the complete VBench scores, including the individual evaluation dimensions omitted from the main-text summary. The 14B LastKV aggregates and individual dimensions come from the same step-10,000 evaluation.

Table 8: Video generation fine-tuning, evaluated with VBench (higher is better; values are percentages). The first three rows are aggregate scores; the remaining rows report raw individual-dimension scores. Evaluation generates 4,730 videos (946 prompts × 5 samples), with each dimension evaluated on its designated subset. The 1.3B teacher-forcing results use 20k-iteration checkpoints and the 14B results use 10k-iteration checkpoints.

	1.3B		**14B**	
	Teacher forcing		Teacher forcing	
VBench metric	Transformer	LastKV	Transformer	LastKV
Quality	83.1	83.5	82.2	83.0
Semantic	76.5	78.0	72.7	76.4
Total	81.8	82.4	80.3	81.6
Subject consistency	93.2	93.3	93.4	94.6
Background consistency	95.0	95.5	95.3	95.1
Temporal flickering	97.3	97.3	96.6	97.8
Motion smoothness	98.5	98.4	98.4	98.9
Dynamic degree	81.1	79.2	68.1	73.3
Aesthetic quality	61.6	62.2	61.1	61.6
Imaging quality	64.1	66.7	66.9	62.8
Object class	91.5	93.5	86.2	92.1
Multiple objects	72.8	74.6	66.3	75.5
Human action	95.6	96.6	95.0	96.0
Color	88.0	87.4	80.3	86.5
Spatial relationship	66.1	69.6	68.3	71.7
Scene	52.9	53.6	48.9	47.2
Appearance style	21.0	22.3	20.3	21.6
Temporal style	24.0	24.2	22.2	23.0
Overall consistency	25.8	25.8	24.4	25.4

H Continued-Pretraining Setup and Evaluation

This appendix documents the training setup, selected benchmarks, and long-context curves in Table 5.

Training data and setup. Transformer and LastKV use the same 1.7T-token corpus for continued pretraining, constructed from the source mixture in Table 9, and both train with an 8K sequence length. The comparison uses 8B models: Start is the initial checkpoint at step 0, while Transformer and LastKV are evaluated at step 18,432. Start has a native 4K training window; the two continued-pretrained models have 8K training windows, so their evaluations beyond 8K measure length extrapolation. The 1.7T-token figure describes the corpus size.

Evaluation protocols and caveats. ARC-Challenge uses 25-shot evaluation. GPQA-Diamond uses the CoT protocol. SuperGPQA uses the official fixed five-shot CoT prompt and parser; its

Source	Documents (%)	Text (%)
nemotron-cc-v2-high-quality-synthetic	28.31	14.93
nemotron-cc-v2-high-quality	13.70	12.62
nemotron-cc-v2-diverse-qa	11.83	8.37
dolma-stack-edu	7.88	6.74
nemotron-cc-v2.1-high-quality-synthetic	6.18	6.01
nemotron-cc-v2-translated-diverse-qa	4.60	5.38
nemotron-cc-v2.1-...translated...synthetic	4.42	3.52
nemotron-pretraining-sft-code	3.63	2.73
nemotron-cc-v2.1-...translated-to-english	3.17	2.41
nemotron-pretraining-sft-general	3.13	4.58
finepdfs	2.42	15.62
Other sources	10.73	17.09

Table 9: Shared continued-pretraining corpus composition. Document and text shares are reported separately, without renormalizing the listed sources. Other sources is the unlisted remainder; ellipses preserve abbreviated source names in the mixture record.

examples are embedded in the prompt. HLE (CoT) uses all 2,158 text questions and the Explanation/Answer/Confidence response format, judged in one pass by `gpt-5.2-2025-12-11` with low reasoning effort, rather than the original official judge. The archived Start, Transformer, and LastKV correct counts are 89, 103, and 56; their 6, 6, and 2 unjudged or parse-failed cases count as incorrect. Transformer has 16.5% BOS nonanswers on MMLU-Redux; this formatting effect also enters the average. MBPP uses the official 3-shot Austin protocol. MBPP+ preserves the source's 0-shot "MBPP+ (instruct)" results, which execute only one base assertion per task, not the extended Plus tests; these values are not standard MBPP+ results. HumanEval and HumanEval+ use plain 3-shot completions with the three examples from the Step 3.5 report (Huang et al., 2026), the latter scored on the EvalPlus extended tests. These are 3-shot variants, not the original 0-shot HumanEval protocol; the HumanEval+ prefix follows the report's HumanEval examples because its HumanEval+ prompt is not printed.

The source also flags possible few-shot example leakage in a legacy MMLU-Redux template. Current cells identify `mmlu_redux_stepproto`; these score transcriptions do not independently establish that the warning has been resolved.

Long-context evaluation and aggregation. The curves in Table 5 report the September 23, 2026 source snapshot. RULER uses 13 tasks per length, with per-task example limits of 100 through 128K and 50 at 256K. Start's measured 8K/16K/32K/64K scores are 0.15/0.00/0.00/0.00, recovered from the source tooltips although the webpage hides these out-of-window results. The Start and Transformer 128K/256K runs use RoPE extrapolation. The completed September 23 rerun gives LastKV at step 18,432 a RULER 256K score of 5.38; the source's August legacy value is not substituted.

LB-v2 is LongBench v2 with 503 multiple-choice questions, left truncation to the stated context, and answer-loglikelihood scoring, not generated-answer evaluation. Its 64K/128K runner differs from the shorter-context runner. The source reports an approximately 0.6-point bf16 cross-runner discrepancy. The dotted 25% reference is uniform four-choice guessing. BABILong reports few-shot qa1–qa5 means, with 100 questions per task. Only the measured 4K/16K/64K/128K points are connected; 2K/8K/32K are unavailable. The September 23 BABILong 128K scores for Start, Transformer, and LastKV are 0.00/0.00/9.00. Missing points are not zero-filled.

Table 5's *Avg.* is the equal-weight mean of its 17 right-hand benchmark values, rounded to two decimals. The plotted context lengths do not contribute additional weight to that mean.

I Training and inference efficiency

I.1 Main comparison

Figure 5 uses A100-SXM4-80GB GPUs, with each method pair measured on the same physical GPU. No other workload runs on the assigned GPUs during a measurement; other GPUs on the node may be active. Table 10 gives the same absolute training times and physical self-attention KV allocations as Figure 5b–d, with identical displayed precision. Training includes the loss forward,

full-history backward, and optimizer update. Prefill builds persistent historical KV; LLM decoding consists of genuine single-token cached calls. Checkpoint loading, input transfer, compilation warmup, and correctness diagnostics are excluded from timing. CUDA synchronization brackets wall-time measurements. We retain all timed repetitions and report the median of repetitions or block/scene medians; plotted whiskers span the corresponding values, not confidence intervals.

Table 10: Absolute training times and physical KV allocations for Figure 5b–d. LLM training uses 32K contexts; inference uses a 128K prefix, with 4K chunks for LastKV. Decode KV follows one target-view query for NVS, one additional three-latent block for video, and 128 additional tokens for LLMs.

Workload	Method	Train (s/step)	Prefill KV (GiB)	Decode KV (GiB)
NVS	Transformer	1.99	0.375	0.375
	LastKV	1.89	0.047	0.047
Video 1.3B	Transformer	13.65	11.247	12.854
	LastKV	15.08	1.982	2.035
LLM 760M	Transformer	2.91	18.000	18.035
	LastKV	3.28	1.875	1.876
LLM 1.5B	Transformer	4.31	25.000	25.049
	LastKV	4.89	2.562	2.564
LLM 3B	Transformer	7.12	36.000	36.070
	LastKV	7.81	3.750	3.753

LLM training. All three LLM scales use $T = 32{,}768$, with $C = 4{,}096$ for LastKV. The 760M, 1.5B, and 3B pairs use one GPU and batch one. Both methods enable nonreentrant activation checkpointing, use the same fused linear cross-entropy kernel, and retain gradients through historical KV. Parameters, gradients, and AdamW moments are BF16; there are no distributed FP32 master weights. All training state remains on the GPU. Reported training times use three fresh model/optimizer blocks with five warmup and ten timed steps per block. The 760M pair reloads nongated step-40,960 checkpoints; 1.5B and 3B use seed-42 matched random initialization with identical common parameters. These larger-model runs measure runtime rather than trained-model quality. Separate untimed checks verify checkpoint recomputation and gradients from losses restricted to final-chunk queries to every preceding input chunk, accounting for the causal label shift. The 3B rerun uses deterministic FlashAttention only in untimed gradient-parity checks to remove backward reduction noise; the native attention policy is restored before full-context checks and timing.

760M chunk-size sweep. Figure 5a fixes batch one and varies $C \in \{1{,}024, 2{,}048, 4{,}096, 8{,}192\}$ for training at $T = 32{,}768$. Training reloads the same nongated checkpoints with a runtime chunk-size override, using the activation-checkpointed full-step protocol above. Each setting has a separately measured same-GPU Transformer reference and reports the median of three block medians. Table 11 reports the exact sweep plotted in Figure 5a. The C4K sweep pair and the 760M pair in panel (b) are independent measurements under the same protocol.

Table 11: 760M training chunk-size sweep for Figure 5a at $T = 32{,}768$ and batch one, with activation checkpointing enabled for both methods. Each chunk setting has a separately measured full-attention Transformer reference on the same GPU. Times are seconds per complete optimizer step (↓).

LastKV chunk	Transformer (s/step)	LastKV (s/step)
1K	2.952	6.227
2K	2.927	4.017
4K	2.911	3.283
8K	2.929	3.157

NVS training. The 28M, eight-layer DL3DV pair uses step-20,000 checkpoints and three frozen real scenes, each with 24 views at 256×256. The LastKV checkpoint uses last-layer feedback (`vgap8`); the Transformer projects layer-local normalized states with the shared last-layer projection. Both use FP32 parameters and AdamW state, BF16 autocast, native activation checkpointing, and the same Flash SDPA backend under PyTorch 2.9.1+cu128. Each model uses three fresh checkpoint/optimizer blocks, with five warmup and twenty timed steps per block; we report the median of

the three block medians. Training retains the native loss, full-history backward, gradient clipping, and optimizer update.

Video training. The 1.3B-class pair uses step-20,000 EMA checkpoints of Transformer and no-sink `AKV254`, each with 1.419B actual parameters and 30 layers. The input has 21 latent frames, grouped into seven three-latent blocks of 4,680 tokens each (32,760 tokens total). Both methods use FP32 parameters, gradients, and AdamW states, BF16 autocast, and activation checkpointing under PyTorch 2.7.1+cu126. Transformer uses its native two-forward implementation; training includes clean/noisy passes, timestep sampling, the native weighted flow loss, full-history backward, gradient clipping, and the optimizer update. We use three consecutive groups of five warmup and ten timed steps, preserving optimizer state across groups, and report the median of the three group medians.

LLM inference. We use matched seeded architecture fixtures with BF16 weights and self-attention KV, PyTorch 2.8.0+cu128, and FlashAttention 2.8.3. All three primary LLM configurations use multi-head attention. LastKV stores one shared last-layer history and two bounded per-layer local banks, retaining previous-chunk blending with fixed $\alpha = 0.5$. On one GPU, physical allocation includes $T + 2LC$ KV positions of the corresponding head width. All measured inference uses GPU-resident weights and KV, with no CPU offload. Transformer prefill uses one full-prompt FlashAttention call per layer, including persistent cache writes. All three LLM inference pairs use batch one and a 128K prefix, with a 4K chunk for LastKV. Decode KV is recorded after consuming 128 fixed tokens following this prefix, remaining within the next chunk. Decode KV counts the physical cache after these calls, including capacity rounded to 256-token pages; prefill KV uses the exact 128K prefix capacity. Each inference case has one warmup and three measured repetitions. Fixed-input correctness comparisons are preserved with the measurements. The full-prompt Transformer control is checked against an independent functional reference with separate positional encoding and direct KV construction. At 128K, the independent reference streams its KV through CPU memory and disk solely during untimed validation; all measured inference remains GPU-resident.

NVS prefill. The eight-layer DL3DV models use the same step-20,000 checkpoints and three fixed real scenes as the NVS training comparison above. Prefill encodes 23 observed views at 256×256, excluding target rendering and loss, with capacity reserved for one additional view. KV allocation is checked on all three scenes, with one warmup and three repetitions per scene. The shared allocation preserves native KV updates and held-out rendering outputs exactly in separate validation runs. Training uses PyTorch 2.9.1+cu128 with the Flash SDPA compatibility backend; prefill uses PyTorch 2.8.0+cu128 and FlashAttention 2.8.3. Both retain FP32 parameters with BF16 autocast, and prefill self-KV is BF16. Shared last-layer storage is counted once across the eight layers. For decode KV, we render the held-out twenty-fourth view from this prefix. The query uses the reserved final-view tail as scratch without committing new historical KV, so post-query physical storage is unchanged. Native query/render outputs and all layer histories agree exactly with the shared-cache implementation on all three scenes.

Video prefill. We use the recovered 1.3B-class, step-20,000 EMA checkpoints, with 1.419B actual parameters. The supplied latent input contains 21 frames, grouped into seven three-latent blocks of 4,680 tokens each. Prefill performs clean-cache commits at timestep zero for both classifier-free-guidance branches; it excludes denoising, T5, and VAE execution. Latent and text tensors are synthetic with seed 42; these calls measure cache storage rather than video quality. Prefill uses PyTorch 2.8.0+cu128 and FlashAttention 2.8.3, with FP32 parameters and BF16 autocast. Self-attention KV is BF16; the two guidance branches and native cross-attention caches (FP32 keys, BF16 values) remain separate. Cross-attention KV is excluded from the self-KV panel. The shared-cache adapter preserves the native 1,560-token blending argument while reserving a conservative 4,680-token recent bank. The training workload described above includes noisy passes and backward, whereas these prefill calls only construct clean historical KV.

Video decode KV. Starting from the same 21-latent prefix, we generate one additional three-latent block using the native FlowUniPC scheduler with 50 steps, shift 5, and guidance scale 5. Both guidance branches run at each denoising step, followed by a clean-cache commit for each branch. We count physical self-KV after these commits, with capacity for all 24 latents (37,440 tokens), including the retained local banks. The next-block noise uses seed 4242; the prefix, checkpoint, precision, and cache-sharing adapter match prefill. One warmup and three measured repetitions

produce identical KV allocations and generated outputs while preserving the existing prefix. T5, VAE, weights, and cross-attention KV are excluded from the self-KV measurement.

J Additional ablation studies

J.1 KV construction

Figure 9 illustrates three historical KV constructions; Table 12 compares their performance. *reusekv* directly reuses K^L_{t-1}, V^L_{t-1}, projected from normalized H^{L-1}_{t-1}. The other two project the final output H^L_{t-1}: *sharekv* uses one additional shared K/V projection pair; *perlayerkv* uses each receiving layer's own projections. All supply history to every depth, including L; the schematics omit current-chunk attention and blending. NVS uses in-training evaluations on Objaverse and full-test evaluation on DL3DV (Figure 4). Long-context LLM reports 30K+ tail loss and RULER average (Table 1); short-context LLM reports WikiText PPL and ten-task average at 124M, with C1024, 20 TPP, and blending (Table 2). Video reports teacher-forced VBench averages. Among the three constructions, *reusekv* gives the best short-context LLM PPL and average.

The three-chunk schematics in Figure 6 illustrate *reusekv*, omitting current-chunk attention and intermediate computations. Its table uses only direct last-layer KV reuse: LastKV scores reproduce the blended *reusekv* entries of Table 12; the short-context w/o blend scores come from a separate unblended H200 run under the same 124M budget and full-test protocol. The long-context w/o blend control uses direct KV reuse at the same 32K context and 4K chunk size, with previous-chunk blending disabled.

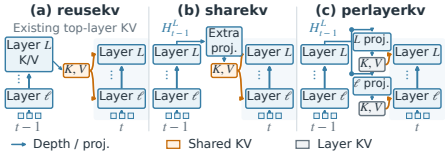


Figure 9: **KV construction.** (a) *reusekv*, (b) *sharekv*, and (c) *perlayerkv*, each at two times. Table 12 compares their NVS, LLM, and video results.

Table 12: KV construction results for NVS, LLMs, and video (↓ for loss/PPL; ↑ otherwise).

	NVS PSNR		LLM long 124M		LLM long 760M		LLM short 124M		Video 1.3B	Video 14B
Method	Obj	DL3DV	Loss	Avg.	Loss	Avg.	PPL	Avg.	Avg.	Avg.
reusekv	25.9	17.8	3.102	17.5	**2.553**	30.7	**52.6**	**42.5**	**81.9**	80.5
sharekv	**26.0**	17.8	**3.092**	18.6	2.556	**34.3**	53.5	41.7	81.8	**81.5**
perlayerkv	25.7	**17.8**	3.107	**20.1**	2.560	32.8	53.9	42.1	81.7	79.3