FASTREBASE: HYBRID LOW-RANK STATES FOR EFFICIENT LINEAR-ATTENTION DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear attention replaces the growing key–value cache of Transformers with a fixed-size recurrent state, but repeatedly accessing this state can still incur substantial memory traffic during batched decoding. Low-rank approximation offers a natural way to reduce this cost. However, maintaining such representations online is challenging due to the cost of repeated singular value decompositions (SVDs). To address this, we introduce FastRebase, which maintains the recurrent states of linear attention models using a hybrid low-rank representation throughout inference without online SVDs. Our key insight is that rank allocation and subspace tracking can be decoupled: FastRebase determines a per-head rank profile offline and tracks only a sequence-dependent subspace during inference using GPU-efficient projections and small-matrix solves. Its hybrid state combines a low-rank history with a residual buffer of recent key–value associations, enabling periodic rebasing that amortizes the cost of subspace updates over multiple decoding steps. We implement FastRebase with custom Triton kernels and evaluate it on models with scalar or diagonal decay and delta-rule state transitions. FastRebase reduces recurrent-state traffic by 61.7–87.3%, while closely matching the dense baselines in zero-shot accuracy and long-context perplexity. At batch size 128 on an H100 GPU, the traffic reduction translates into 1.36–4.73× speedups for linear-attention recurrence and 1.18–2.31× for end-to-end decoding.

1 INTRODUCTION

Autoregressive large language model (LLM) inference requires repeatedly accessing and updating context-dependent state. In Transformers (Vaswani et al., 2017), this state is stored as a key–value (KV) cache whose size grows linearly with context length. Linear attention models (Katharopoulos et al., 2020) offer an attractive alternative: they summarize the context into a fixed-size recurrent state and therefore maintain a constant memory footprint with respect to context length. However, a fixed-size state is not necessarily inexpensive. For each linear attention head with key dimension d_k and value dimension d_v, the recurrent state H_t is represented as a dense $d_k \times d_v$ matrix. This state must be read, updated, and written back at every decoding step, incurring substantial memory traffic despite its constant size. As batch size increases and model-parameter traffic is amortized across requests, recurrent-state accesses can become a dominant inference bottleneck.

Recent approaches reduce recurrent-state size through static modifications to the model structure, such as pruning the shared key–query dimension d_k (Nazari & Rusch, 2026b) or permanently removing entire linear attention heads based on their state statistics (Sun et al., 2026). Because these pruning decisions are irreversible, aggressive static pruning can incur non-negligible quality degradation and may require recovery fine-tuning. Other recent work reduces writes to the recurrent state by deferring dense-state updates (Zou & Zhong, 2026). However, the dense recurrent state itself remains uncompressed. We therefore ask a different question: Can the evolving recurrent state of a pretrained linear attention model be maintained in a compact form throughout inference, without retraining or modifying the model itself?

The recurrent state of a linear attention head is formed by accumulating rank-one key–value associations under a structured state transition: $H_t = A_t H_{t-1} + k_t v_t^\top$. In decay-based models, A_t is typically scalar or diagonal (Sun et al., 2023; Yang et al., 2024a), whereas delta-rule models further incorporate rank-one erase or correction operators (Yang et al., 2024b; 2025) into the transition. By

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.

attenuating or overwriting previously stored associations, these transitions can limit the effective dimensionality of the accumulated state. Consistent with this intuition, prior work shows that recurrent states in trained linear attention models often exhibit effective ranks well below their maximum possible rank (Siems et al., 2025; Sun et al., 2026). This observation motivates maintaining the recurrent state in factorized form, $H_t \approx U_t W_t^\top$, thereby reducing recurrent-state memory traffic and allowing head outputs to be computed directly from the factors without materializing the dense state.

However, maintaining such a low-rank factorization online is challenging because the retained subspace must adapt to successive state updates. The appropriate rank also varies substantially across heads, layers, and model families, so a uniform rank either over-provisions highly compressible states or provides insufficient capacity for states with higher effective rank. A straightforward approach is therefore to jointly determine per-head ranks and their corresponding subspaces online using singular value decomposition (SVD). Repeated SVDs, however, can outweigh the savings from the compact representation. Even incremental SVD methods specialized for rank-one updates (Brand, 2002; 2006) require per-token basis orthogonalization and an SVD in a low-rank subspace.

Our key observation is that, for recurrent states, the two decisions underlying a low-rank approximation—how many directions to retain and which subspace they should span—can be decoupled. Specifically, we find that the rank required for each head can be determined offline and reused throughout inference, whereas using a fixed subspace substantially degrades model quality. At runtime, the problem is therefore reduced to tracking a sequence-dependent subspace at the calibrated rank, rather than estimating both the rank and the subspace online.

To realize this separation, we propose FastRebase, which combines a hybrid low-rank state representation with periodic subspace updates. We represent the state as $H_t \approx U_t W_t^\top + K_t^{\text{res}} (V_t^{\text{res}})^\top$, storing older key–value associations in low-rank factors while retaining recent updates in a residual buffer. This buffer not only delays truncation of strong contributions from recent tokens, but also amortizes the cost of subspace adaptation by folding multiple buffered updates into the low-rank history in a single subspace update, which we call a rebase. At each rebase, FastRebase updates the subspace by recycling the current value-side basis W_t using GPU-efficient projections and small-matrix solves, eliminating costly online SVDs.

We implement FastRebase with custom Triton kernels (Tillet et al., 2019) for the hybrid low-rank state representation and periodic rebasing. We integrate it into models with scalar or diagonal decay, as well as models with delta-rule state transitions. Our results show that the hybrid representation reduces recurrent-state memory traffic by 61.7–87.3% while largely preserving downstream-task accuracy and language-modeling quality. At batch size 128, FastRebase achieves speedups of 1.36–4.73$\times$ for linear-attention recurrence and 1.18–2.31$\times$ for end-to-end decoding on an H100 GPU.

2 Background and Motivation

2.1 Linear Attention

For a single linear attention head, let $q_t, k_t \in \mathbb{R}^{d_k}$ and $v_t \in \mathbb{R}^{d_v}$ denote the query, key, and value at token t. Its linear recurrence and output computation can be written as

$$H_t = A_t H_{t-1} + k_t v_t^\top, \qquad o_t^\top = q_t^\top H_t, \tag{1}$$

where $H_t \in \mathbb{R}^{d_k \times d_v}$ and $A_t \in \mathbb{R}^{d_k \times d_k}$ denote the recurrent state and state transition, respectively. The transition A_t determines how the previous state is transformed, while the rank-one term $k_t v_t^\top$ incorporates the current key–value association into the state.

Because linear attention represents the context using a fixed-size recurrent state, its memory capacity does not grow with context length (Schlag et al., 2021). To manage this finite memory, linear attention models employ structured state transitions that selectively forget or overwrite previously stored information. Decay-based models use scalar or diagonal decay to attenuate historical information (Sun et al., 2023; Yang et al., 2024a), whereas delta-rule models modify stored key–value associations by erasing part of the existing mapping before writing the new association (Yang et al., 2024b; 2025). These transitions can be written uniformly in the diagonal-plus-rank-one form

$$A_t = D_t + b_t a_t^\top, \tag{2}$$

where D_t is a scalar or diagonal decay operator and $b_t a_t^\top$ is an optional rank-one correction. Decay-only models are represented by $b_t a_t^\top = 0$, while models without decay are represented by $D_t = I$.

2.2 Linear-Attention Recurrence under Batched Decoding

To quantify the recurrence cost of linear attention during autoregressive generation, we profile per-token decoding latency across batch sizes for six 1.3B linear attention models on an H100 GPU. Figure 1 shows that the linear-attention recurrence accounts for an increasing fraction of decoding latency as batch size grows, reaching 62% on average at batch size 128.

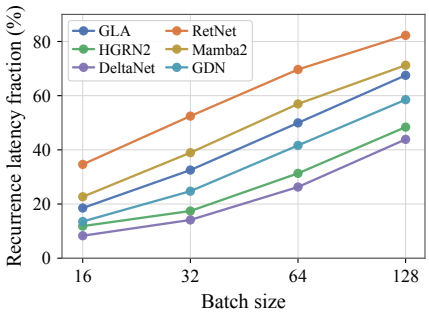

Figure 1: Fraction of decoding latency spent on linear-attention recurrence.

This trend arises from the different scaling characteristics of the underlying operations. Projection and feed-forward layers reuse their weights across sequences in a batch, amortizing the cost of weight accesses as the batch size increases. In contrast, recurrent states are sequence-specific and cannot be shared across the batch, so the associated memory traffic scales with batch size. As a result, linear-attention recurrence can dominate decoding latency at large batch sizes, motivating a compressed representation of the recurrent state to reduce its memory traffic.

2.3 Low-Rank Structure and Online Subspace Updates

We next examine whether the recurrent state can be compressed using a low-rank representation. For a recurrent state $H \in \mathbb{R}^{d_k \times d_v}$, consider its singular value decomposition (SVD)

$$H = L\Sigma R^\top, \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq 0. \tag{3}$$

Truncating the SVD to its top r singular components gives the best rank-r approximation in Frobenius norm (Eckart & Young, 1936).

We characterize the compressibility of a recurrent state using its normalized cumulative singular value energy ($c_H(r)$) and the corresponding η-energy rank ($r_\eta(H)$), defined as

$$c_H(r) = \frac{\sum_{j=1}^{r} \sigma_j^2}{\sum_j \sigma_j^2}, \qquad r_\eta(H) = \min\{r : c_H(r) \geq \eta\}. \tag{4}$$

Figure 2(a) shows that most of the singular value energy of recurrent states is concentrated in a small subset of singular directions, consistent with prior observations that recurrent states can have substantially lower effective rank (Siems et al., 2025; Sun et al., 2026). At the same energy-retention target η, however, the η-energy rank varies substantially across layers and heads, as shown in Figure 2(b). The varying rank requirements across heads suggest that uniform rank assignment is suboptimal, as it can over-provision lower-rank heads while under-provisioning higher-rank ones.

Beyond rank allocation, maintaining such a low-rank recurrent state online introduces a key challenge. A natural approach for incorporating each rank-one association $k_t v_t^\top$ is to update the truncated

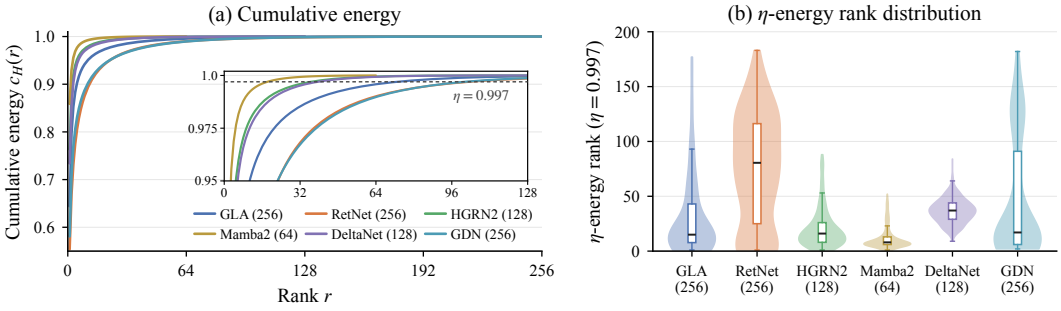

Figure 2: Low-rank structure and rank heterogeneity of recurrent states across layers and heads. (a) Per-model mean cumulative singular value energy. (b) Distribution of the η-energy rank at $\eta = 0.997$. Per-sequence cumulative-energy curves are averaged over 128 sequences sampled from WikiText-2 (Merity et al., 2017). Parenthesized values denote the rank limit $r_{\max} = \min(d_k, d_v)$.

SVD incrementally at every token. Incremental SVD methods (Brand, 2002; 2006) avoid repeatedly decomposing the full $d_k \times d_v$ state, but still require an $(r+1) \times (r+1)$ SVD for each rank-one update. The cost of such SVD-based subspace updates can offset the savings from the compact state representation (Section 5.1).

3 Hybrid Low-Rank Recurrent States

3.1 Hybrid State Representation

FastRebase represents the recurrent state of each linear attention head with low-rank factors for the compressed history and a residual buffer for the keys and values of recent tokens. Given a target rank r, this hybrid representation approximates the recurrent state $H_t \in \mathbb{R}^{d_k \times d_v}$ as

$$\widehat{H}_t = U_t W_t^\top + K_t^{\mathrm{res}} (V_t^{\mathrm{res}})^\top, \tag{5}$$

where $U_t \in \mathbb{R}^{d_k \times r}$ and $W_t \in \mathbb{R}^{d_v \times r}$ are the key- and value-side factors of the rank-r compressed history, while $K_t^{\mathrm{res}} \in \mathbb{R}^{d_k \times s_t}$ and $V_t^{\mathrm{res}} \in \mathbb{R}^{d_v \times s_t}$ store the keys and values of s_t recent tokens. Because contributions from older tokens may have been attenuated by state decay or partially overwritten by delta-rule updates, recent contributions can remain relatively strong. The residual buffer therefore keeps recent key–value associations uncompressed by retaining their keys and values.

The same residual buffer also enables periodic rebasing, in which multiple older associations are incorporated into the compressed history with a single subspace update. Let p denote the rebase interval and ℓ the number of most recent key–value associations retained after each rebase. Since FastRebase accumulates p new associations between rebases, the residual buffer contains $p + \ell$ associations before each rebase. The rebase then partitions the buffer and updates the hybrid state to $\widehat{H}'_t$ as follows:

$$\begin{aligned} K_t^{\mathrm{res}} &= \begin{bmatrix} K_t^{\mathrm{old}} & K_t^{\mathrm{keep}} \end{bmatrix}, \qquad V_t^{\mathrm{res}} = \begin{bmatrix} V_t^{\mathrm{old}} & V_t^{\mathrm{keep}} \end{bmatrix}, \\ U'_t (W'_t)^\top &\approx U_t W_t^\top + K_t^{\mathrm{old}} (V_t^{\mathrm{old}})^\top, \qquad \widehat{H}'_t = U'_t (W'_t)^\top + K_t^{\mathrm{keep}} (V_t^{\mathrm{keep}})^\top, \end{aligned} \tag{6}$$

where U'_t and W'_t denote the updated factors. At each rebase, the older p associations are folded into the compressed history, while the newest ℓ associations remain in the residual buffer. This amortizes the cost of subspace adaptation over p decoding steps.

3.2 Linear-Attention Recurrence on Hybrid States

The linear-attention recurrence in (1) preserves the hybrid representation between rebases. Substituting (5) into this recurrence gives

$$\widehat{H_t} = (A_t U_{t-1}) W_{t-1}^\top + \begin{bmatrix} A_t K_{t-1}^{\mathrm{res}} & k_t \end{bmatrix} \begin{bmatrix} V_{t-1}^{\mathrm{res}} & v_t \end{bmatrix}^\top = U_t W_t^\top + K_t^{\mathrm{res}} (V_t^{\mathrm{res}})^\top. \tag{7}$$

For the structured transition in (2), the low-rank factors and residual buffer are updated as

$$\begin{aligned} U_t &= D_t U_{t-1} + b_t \big(a_t^\top U_{t-1}\big), \qquad W_t = W_{t-1}, \\ K_t^{\mathrm{res}} &= \begin{bmatrix} D_t K_{t-1}^{\mathrm{res}} + b_t \big(a_t^\top K_{t-1}^{\mathrm{res}}\big) & k_t \end{bmatrix}, \quad V_t^{\mathrm{res}} = \begin{bmatrix} V_{t-1}^{\mathrm{res}} & v_t \end{bmatrix}. \end{aligned} \tag{8}$$

The transition A_t acts only on the key-side factor and the buffered keys, leaving the value-side factor and existing values in the buffer unchanged. The new key and value are then appended to the residual buffer. The hybrid-state update in (8) therefore introduces no additional approximation between rebases. We further reduce memory writes to U_t using lazy decay: decay factors are accumulated separately rather than applied to U_t at every step (Appendix A.1).

The output of each head can also be computed directly from the hybrid state:

$$\widehat{o}_t^\top = q_t^\top \widehat{H}_t = \big(q_t^\top U_t\big) W_t^\top + \big(q_t^\top K_t^{\mathrm{res}}\big) (V_t^{\mathrm{res}})^\top. \tag{9}$$

Thus, the primary benefit of the hybrid representation is that recurrent-state updates and output computation operate entirely on the low-rank factors and residual buffer without materializing the dense state. This formulation also preserves the chunkwise structure used for parallel sequence processing in linear attention models (Yang et al., 2024a;b). Only the inter-chunk state computations are performed in factorized form using (8)–(9), while intra-chunk computations remain unchanged.

With s_t associations in the residual buffer, the hybrid state representation comprises $(r+s_t)(d_k+d_v)$ elements. Because the observed η-energy ranks are well below the corresponding rank limits (Figure 2(b)) and periodic rebasing ensures $s_t \leq p + \ell$, the resulting recurrent-state size is substantially smaller than that of a dense $d_k \times d_v$ state. Consequently, during autoregressive decoding, the hybrid representation reduces recurrent-state memory traffic while also lowering the per-token arithmetic complexity from $O(d_k d_v)$ to $O((d_k + d_v)(r + s_t))$. The remaining challenge is to perform periodic rebasing without offsetting these savings with subspace update overhead.

4 FastRebase

Rebasing involves two decisions: selecting per-head ranks and updating the corresponding subspaces to incorporate buffered associations. We first formulate an adaptive-rank rebase that makes both decisions online by recomputing a basis for the dominant subspace, and then introduce FastRebase, which moves rank selection offline and recycles the value-side basis for online subspace updates.

4.1 Adaptive-Rank Rebase

We maintain the value-side factor W in (5) with orthonormal columns, $W^\top W = I_r$. Because the linear-attention recurrence leaves W unchanged as in (8), W remains orthonormal between rebases. This orthonormality reduces the rebase problem to a small augmented factorization, akin to the low-dimensional augmentation used in incremental SVD (Brand, 2002; 2006). Directly maintaining the recurrent state in the SVD form $H_t = L_t \Sigma_t R_t^\top$, however, would require reorthogonalizing the transformed left basis $A_t L_{t-1}$ at each state update. We instead leave U unconstrained, allowing the transition to be absorbed directly into U while W remains orthonormal.

We now derive the augmented factorization. To fold $K^{\text{old}}(V^{\text{old}})^\top$ into the compressed history, we first project V^{old} onto the value-side subspace spanned by W and compute the orthogonal residual:

$$P = W^\top V^{\text{old}}, \qquad E = V^{\text{old}} - WP. \tag{10}$$

Let $E = QR$ be a thin QR factorization. Substituting $V^{\text{old}} = WP + QR$ into the rebase in (6) yields

$$UW^\top + K^{\text{old}}(V^{\text{old}})^\top = \left(U + K^{\text{old}} P^\top\right) W^\top + K^{\text{old}} R^\top Q^\top = MZ^\top, \tag{11}$$

where the augmented factors are $M = [U + K^{\text{old}} P^\top \quad K^{\text{old}} R^\top]$ and $Z = [W \quad Q]$.

The adaptive-rank rebase requires an SVD of the merged history $MZ^\top$ to determine its η-energy rank and corresponding dominant value-side subspace. Since Z is orthonormal, the same nonzero singular values of the dense $d_k \times d_v$ matrix $MZ^\top$ can instead be computed from the smaller $d_k \times (r+p)$ matrix M. We implement this SVD through the eigendecomposition of the $(r+p)\times(r+p)$ Gram matrix $M^\top M = T \Lambda T^\top$ and select the new η-energy rank r' from the eigenvalues $\lambda_j = \sigma_j(M)^2$ using (4). Letting $T_{r'}$ contain the leading r' eigenvectors, the rebased factors are

$$U' = MT_{r'}, \qquad W' = ZT_{r'}. \tag{12}$$

However, even for small M, the SVD remains inefficient on GPUs because it exposes limited parallelism and requires multiple synchronization steps. Repeating this decomposition across layers and heads at each rebase therefore introduces considerable overhead throughout decoding (Section 5.1).

4.2 FastRebase via Value-Subspace Recycling

FastRebase eliminates online SVD by decoupling rank selection from subspace adaptation. For each head, we use an offline-calibrated rank r throughout inference while updating the value-side subspace online by recycling the current basis W.

Since W remains unchanged between rebases and each rebase introduces at most p new value-side directions through Q, W provides a natural warm start for the next subspace update. For the augmented factorization in (11), FastRebase replaces the SVD of the merged history $MZ^\top$ with one-step subspace iteration initialized from W. In the coordinates of Z, the current value-side basis W is represented by $J = \left[\begin{smallmatrix} I_r \\ 0 \end{smallmatrix}\right]$, where $W = ZJ$. Defining $G = M^\top M$, the subspace update is

$$\Phi = \text{orth}(GJ), \tag{13}$$

where $\mathrm{orth}(\cdot)$ denotes an orthonormal basis for the column space. Using W as a pivot, this update rotates the value-side subspace toward the new directions in Q to adapt it to the merged history.

To compute Φ without materializing the full Gram matrix G, FastRebase exploits the block structure of $M = [M_\parallel \; M_\perp]$ corresponding to $Z = [W \; Q]$. When $M_\parallel^\top M_\parallel$ is nonsingular, an orthonormal basis for the column space of GJ can instead be obtained by solving

$$S\big(M_\parallel^\top M_\parallel\big) = M_\perp^\top M_\parallel, \qquad \Phi = \mathrm{orth}\begin{bmatrix} I_r \\ S \end{bmatrix}, \tag{14}$$
$$\text{where} \quad M_\parallel = U + K^{\mathrm{old}} P^\top, \qquad M_\perp = K^{\mathrm{old}} R^\top.$$

In practice, we add a small diagonal loading λI to $M_\parallel^\top M_\parallel$ when solving (14) to handle rank-deficient or ill-conditioned cases. This block formulation also reduces the orthonormalization to an $r \times r$ Cholesky factorization of $I_r + S^\top S$, thereby enabling efficient normalization while avoiding the numerical sensitivity associated with forming $J^\top G^2 J$. Appendix A.2 provides the derivation and implementation details.

Applying the subspace update Φ to the augmented factors yields

$$U' = M\Phi, \qquad W' = Z\Phi. \tag{15}$$

Since both Z and Φ have orthonormal columns, W' remains an orthonormal rank-r basis.

4.3 Offline Per-Head Rank Calibration and Initialization

To obtain the per-head rank profile used by FastRebase, we perform a one-time offline calibration using dense recurrent states from a small set of N sequences. For head h, let $c_{h,n}(r)$ denote the normalized cumulative singular value energy of the state produced by calibration sequence n, as defined in (4). Given an energy-retention target η, we select the rank for head h as

$$\bar{c}_h(r) = \frac{1}{N}\sum_{n=1}^{N} c_{h,n}(r), \qquad r_h = \min\left\{r : \bar{c}_h(r) \geq \eta\right\}. \tag{16}$$

We characterize how state energy is distributed across ranks by averaging the energy-retention curves across sequences rather than averaging the per-sequence η-energy ranks, so that rank selection directly targets the desired average energy retention. The recurrent states and subspaces obtained during calibration are discarded, while only the resulting rank profile is retained throughout inference.

Given the calibrated rank profile, FastRebase initializes the recurrent state directly in the hybrid low-rank representation. Specifically, we set $U_0 = 0$ and $K_0^{\mathrm{res}} = V_0^{\mathrm{res}} = \varnothing$, while initializing W_0 as a random orthonormal basis with r_h columns. Because $U_0 W_0^\top = 0$, the choice of initial basis has no effect on the initial recurrent state. Consequently, FastRebase can begin inference directly with the hybrid representation without materializing a dense recurrent state.

5 Experiments

Models and Datasets. We evaluate FastRebase on six pretrained 1.3B linear attention models spanning scalar or diagonal decay and delta-rule transitions: GLA (Yang et al., 2024a), RetNet (Sun et al., 2023), HGRN2 (Qin et al., 2024), Mamba2 (Dao & Gu, 2024), DeltaNet (Yang et al., 2024b), and GatedDeltaNet (GDN) (Yang et al., 2025). To assess whether the compact recurrent-state representation preserves model quality, we report language-modeling perplexity on WikiText-2 (Merity et al., 2017) and zero-shot accuracy on seven downstream tasks (Appendix B). We further examine how FastRebase affects long-context behavior on 16K-token sequences sampled from PG-19 (Rae et al., 2020) and retrieval accuracy on six recall-intensive tasks (Arora et al., 2024).

FastRebase Implementation. We implement FastRebase on top of Flash Linear Attention (FLA) (Yang & Zhang, 2024), with custom Triton kernels for the hybrid-state recurrence and periodic rebasing via value-subspace recycling. For performance evaluation, we use the dense FLA implementations as baselines. Unless otherwise specified, we calibrate the per-head rank profile at $\eta = 0.997$ using 32 sequences of 1,024 tokens sampled from the WikiText-2 training split. The resulting ranks are lower-bounded by 4 and capped at half the corresponding rank limit, up to a maximum of 96. We retain $\ell = 4$ recent associations and use a rebase interval of $p = 16$.

Table 1: Comparison of state-size reduction, language-modeling quality, and zero-shot accuracy. For AdaptiveRebase, state size is computed using the mean rank after 1,024 tokens on WikiText-2.

Model	Method	State-size reduction ↑	Wiki. ppl ↓	LMB. acc ↑	PIQA acc_n ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc_n ↑	ARC-c acc_n ↑	BoolQ acc ↑	Avg. ↑
GLA	Dense	0%	17.69	46.03	70.73	48.89	53.43	48.78	27.73	57.86	50.49
	DRRQR	25%	2267.42	0.33	54.03	28.21	51.38	31.61	24.57	49.17	34.19
	AdaptiveRebase	78%	17.77	46.11	70.62	48.87	53.04	48.70	27.30	57.74	50.34
RetNet	Dense	0%	18.25	42.87	70.35	48.19	53.12	49.24	27.05	58.41	49.89
	DRRQR	25%	29.93	32.04	70.02	44.93	53.04	45.41	25.17	60.31	47.27
	AdaptiveRebase	65%	18.31	42.87	70.46	48.08	53.43	49.62	26.96	58.41	49.98
HGRN2	Dense	0%	16.97	50.20	70.95	51.73	53.67	52.10	28.16	60.09	52.41
	DRRQR	25%	251534.77	0.00	52.99	27.17	48.15	28.11	22.27	38.10	30.97
	AdaptiveRebase	47%	17.24	50.01	70.62	51.73	53.28	52.06	28.33	59.60	52.23
Mamba2	Dense	0%	13.19	64.68	73.67	59.80	60.54	60.44	33.53	63.52	59.45
	DRRQR	25%	1898.88	29.89	69.42	43.51	58.33	52.90	31.66	54.46	48.60
	AdaptiveRebase	47%	13.36	64.72	73.50	59.81	60.77	59.97	33.45	63.30	59.36
DeltaNet	Dense	0%	16.75	48.46	70.89	50.49	54.93	51.39	26.11	60.86	51.88
	DRRQR	25%	18.96	36.04	70.73	52.52	54.30	49.16	28.84	59.27	50.12
	AdaptiveRebase	25%	17.00	48.17	70.78	50.39	54.54	51.30	26.11	61.10	51.77
GDN	Dense	0%	16.83	51.48	74.10	60.19	61.64	67.55	40.61	61.53	59.59
	DRRQR	25%	17.35	47.99	73.78	60.30	61.33	66.75	39.85	58.32	58.33
	AdaptiveRebase	65%	16.92	51.27	73.99	60.24	61.72	67.76	40.53	61.38	59.56

5.1 Low-Rank State Approximation vs. Static Pruning

Comparison Setup. To examine the feasibility of low-rank state approximation, we compare the adaptive-rank rebase (AdaptiveRebase) with DRRQR (Nazari & Rusch, 2026b), a static pruning baseline without additional training. We evaluate DRRQR at a 25% state-size reduction, using the original implementation for DeltaNet and GDN and extending its channel-selection procedure to the remaining models while accounting for model-specific structures such as RoPE. Appendix B provides details on the AdaptiveRebase configuration and DRRQR implementation.

Quality and State-Size Reduction. Table 1 shows that AdaptiveRebase consistently preserves near-dense quality while reducing the logical size of the recurrent state by 25–78%, including both the low-rank factors and residual buffer. DRRQR with a fixed 25% state-size reduction remains relatively competitive on DeltaNet and GDN, but incurs substantially larger quality degradation on the remaining models. This contrast highlights the difficulty of applying static pruning consistently across diverse linear attention models. Moreover, during autoregressive decoding, recurrent-state memory traffic under static pruning decreases in proportion to the reduced state size, whereas the hybrid representation achieves a greater reduction in memory traffic than in logical state size through its factorized recurrence (Section 3.2).

AdaptiveRebase Overhead. Despite preserving near-dense quality, AdaptiveRebase incurs substantial overhead from periodic rebasing. Table 2 reports relative latency for 64 decoding steps after a 1,024-token prefill at batch size 32 on an H100 GPU. The hybrid state reduces linear-attention recurrence latency to 0.21–0.66× that of the dense baseline. However, even with a rebase interval of $p = 16$, adaptive-rank rebasing incurs 45.51–493.53× the latency of the dense-state recurrence, overwhelming these savings. As a result, total decoding latency increases to 16.57–84.69× that of the dense baseline. More detailed results in Appendix F show that the SVD accounts for most of the rebase overhead. These results motivate eliminating the costly online SVD while retaining the hybrid state representation.

Table 2: AdaptiveRebase latency relative to dense baselines on an H100 GPU. Rebase latency is normalized to dense-state recurrence.

Model	Hybrid-state recurrence	Adaptive-r rebase	Full decode
GLA	0.21×	45.51×	16.57×
RetNet	0.22×	73.27×	40.73×
HGRN2	0.43×	416.25×	83.83×
Mamba2	0.29×	203.19×	84.69×
DeltaNet	0.66×	493.53×	83.97×
GDN	0.52×	104.82×	29.68×

Table 3: Quality ablation of FastRebase. The final row corresponds to FastRebase. Acc. and ppl denote zero-shot accuracy averaged over the seven tasks and WikiText-2 perplexity, respectively.

	GLA		RetNet		HGRN2		Mamba2		DeltaNet		GDN	
Method	Acc. ↑	ppl ↓	Acc. ↑	ppl ↓	Acc. ↑	ppl ↓	Acc. ↑	ppl ↓	Acc. ↑	ppl ↓	Acc. ↑	ppl ↓
AdaptiveRebase	50.34	17.77	49.98	18.31	52.23	17.24	59.36	13.36	51.77	17.00	59.56	16.92
+ Rank calibration	50.36	17.76	50.04	18.30	52.32	17.16	59.42	13.28	51.81	17.03	59.68	16.88
+ Fixed subspace	43.79	27.74	44.65	25.97	45.65	22.92	52.36	17.50	45.59	22.08	53.65	20.18
+ Subspace recycling	50.39	17.78	50.00	18.31	52.20	17.26	59.34	13.31	51.84	17.08	59.69	16.91

5.2 FastRebase Quality and Efficiency

Quality Preservation. We conduct an ablation study to examine whether the design choices in FastRebase preserve the quality achieved by AdaptiveRebase. First, we replace online η-energy rank selection with the rank profile calibrated offline at $\eta = 0.997$, while retaining the SVD-based subspace update. At 1,024 tokens, the mean rank of the low-rank factors is 29.6 under AdaptiveRebase and 32.3 with the offline-calibrated rank profile. Table 3 shows that this change has negligible effect on either average zero-shot accuracy or WikiText-2 perplexity. Second, we further replace the online SVD with a fixed value-side basis computed from the calibration states (Appendix B). This substantially degrades performance on both metrics. In contrast, using the same rank profile, value-subspace recycling (Section 4.2) closely matches the near-dense quality achieved by AdaptiveRebase. These results support our key observation that rank allocation can be moved offline while the corresponding subspace must adapt to each sequence at runtime. FastRebase performs this adaptation without an online SVD.

Decoding Performance. We next evaluate whether FastRebase translates the reduced recurrent-state traffic of the hybrid representation into practical decoding speedups. Table 4 reports the analytical recurrent-state traffic and the measured speedups over the dense FLA implementations for both linear-attention recurrence and end-to-end decoding at batch size 128. For end-to-end decoding, we capture both FastRebase and the dense baseline using CUDA Graphs and measure their performance over 256 decoding steps. Across all six models, FastRebase reduces recurrent-state traffic by 61.7–87.3% and achieves a 2.27–7.93× speedup for the recurrence excluding rebasing overhead. By replacing the SVD-based update with GPU-efficient value-subspace recycling, FastRebase retains a 1.36–4.73× recurrence speedup even with periodic rebasing. Overall, FastRebase achieves a 1.18–2.31× end-to-end decoding speedup. Figure 3 further shows that the end-to-end benefit increases with batch size, as linear-attention recurrence accounts for a larger fraction of decoding time. Detailed batch-wise results are provided in Appendix F.

Table 4: FastRebase state traffic and speedups relative to the dense baselines on an H100 GPU.

Model	State traffic	Recurrence speedup		Decode speedup
		w/o rebase	w/ rebase	
GLA	12.7%	7.93×	4.73×	2.31×
RetNet	23.0%	4.77×	2.77×	2.19×
HGRN2	29.7%	3.73×	2.28×	1.47×
Mamba2	36.6%	3.95×	2.42×	1.78×
DeltaNet	38.3%	2.27×	1.36×	1.18×
GDN	21.7%	2.75×	1.81×	1.42×

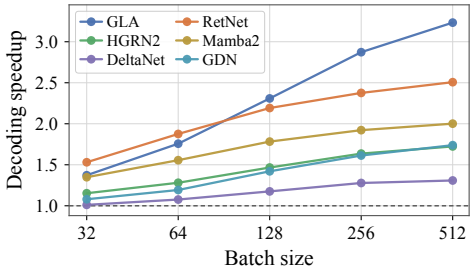

Figure 3: FastRebase end-to-end decoding speedup across batch sizes on an H100 GPU.

5.3 Long-Context and Recall-Intensive Evaluation

Long-Context Extrapolation. We evaluate pretrained checkpoints with a 2K-token training context (Appendix B). To examine whether FastRebase preserves their long-context extrapolation behavior, we evaluate perplexity on PG-19 at context lengths up to 16K tokens. As shown in Figure 4, except for Mamba2, FastRebase remains within 2.5% of the dense baseline across all evaluated context lengths. The dense Mamba2 baseline, however, exhibits a sharp increase in perplexity beyond

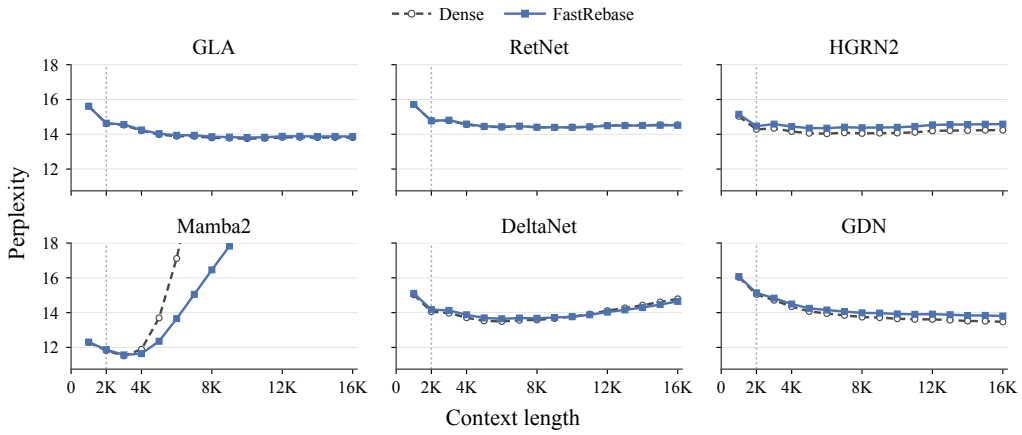

Figure 4: Context-length extrapolation on 64 sequences of 16K tokens from PG-19.

the 2K training context, as previously reported for the official checkpoints (Buitrago & Gu, 2025). Prior work has attributed this length-generalization failure to state overparameterization relative to the training context (Chen et al., 2025). FastRebase delays this degradation, suggesting that periodic low-rank compression may act as an implicit regularizer on the recurrent state. Overall, FastRebase preserves model quality under context-length extrapolation relative to the dense baselines.

Recall-Intensive Tasks. We further evaluate FastRebase on the six recall-intensive tasks (Arora et al., 2024). As reported in Table 5, FastRebase yields lower average accuracy than the dense baselines. To examine whether this degradation arises from the limited capacity of the low-rank state or from repeated online subspace adaptation, we consider a one-shot SVD initialization from the dense prefill state. Specifically, we perform prefill using the dense recurrence, apply a truncated SVD to the resulting state, and then switch to FastRebase for generation. With the same per-head rank profile calibrated at $\eta = 0.997$, the one-shot SVD substantially recovers retrieval accuracy. For DeltaNet, the higher-rank profile ($\eta = 0.999$) further narrows the remaining gap, consistent with the greater sensitivity of its non-decaying state to compression. The accuracy recovery with one-shot SVD using the same rank profile suggests that the observed retrieval degradation cannot be explained solely by limited hybrid-state capacity. Future work could investigate how to better preserve state information relevant to retrieval during online subspace adaptation, such as by incorporating query-side signals.

Table 5: Average accuracy on six recall-intensive tasks with a 2K-token context.

Model	Dense	FastRebase	One-shot SVD	
		η=.997	η=.997	η=.999
GLA	31.22	28.40	30.34	30.29
RetNet	28.50	27.36	28.37	28.36
HGRN2	26.98	24.80	26.74	26.75
Mamba2	39.68	33.04	38.78	38.77
DeltaNet	35.38	27.69	32.05	34.81
GDN	37.45	33.84	36.51	36.48

6 Conclusion

We introduce FastRebase, an online compression method for linear-attention decoding that maintains recurrent states in a hybrid low-rank form throughout inference. FastRebase decouples rank allocation from subspace tracking by combining offline per-head rank calibration with online value-subspace recycling, thereby eliminating costly online SVDs. A residual buffer for recent associations further amortizes subspace adaptation cost through periodic rebasing. Across six pretrained 1.3B linear attention models spanning decay-based and delta-rule transitions, FastRebase reduces recurrent-state traffic by 61.7–87.3% while preserving zero-shot accuracy and long-context perplexity. At batch size 128, this reduction translates into speedups of up to 4.73× for linear-attention recurrence and up to 2.31× for end-to-end decoding on an H100 GPU. Overall, these results demonstrate that online low-rank compression of linear-attention recurrent states can deliver practical end-to-end decoding speedups when sequence-dependent subspace adaptation is performed efficiently.

AI USE STATEMENT

In this work, we used generative AI tools to proofread the manuscript, provide code completion suggestions, and assist with code review for debugging purposes. All AI-assisted outputs were carefully reviewed and revised by the authors to ensure that they are correct and faithfully reflect our intentions. The research ideas, methodology, and experimental design were developed entirely by the authors. The authors take full responsibility for the research, implementation, and final manuscript.

REFERENCES

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proceedings of the VLDB Endowment*, 17(2):92–105, 2023. doi: 10.14778/3626292.3626294.

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: Closing the recall gap for recurrent language models. *arXiv preprint arXiv:2407.05483*, 2024. URL https://arxiv.org/abs/2407.05483.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision – ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pp. 707–720. Springer, 2002. doi: 10.1007/3-540-47969-4_47.

Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006. doi: 10.1016/j.laa.2005.07.021.

Ricardo Buitrago and Albert Gu. Understanding and improving length generalization in recurrent models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5856–5874. PMLR, 2025.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7da6e0e00702c60607a6ae05c802ef85-Abstract-Conference.html.

Yingfa Chen, Xinrong Zhang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Stuffed Mamba: Oversized states lead to the inability to forget. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=CdRauNXD1w.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

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 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1246.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi: 10.1007/BF02288367.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. ZipCache: Accurate and efficient KV cache quantization with salient token identification. In *Advances in Neural Information Processing Systems*, volume 37, pp. 68287–68307, 2024.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303, 2024.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, TianQi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV: Adaptive KV compression via trainable orthogonal projection. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d7b351608d824a4680344a02b180a947-Abstract-Conference.html.

Junqing Lin, Jingwei Sun, and Guangzhong Sun. DeltaLog: Deferred materialization of recurrent states for linear attention decoding. *arXiv preprint arXiv:2608.15533*, 2026. URL https://arxiv.org/abs/2608.15533.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems*, volume 36, pp. 52342–52364, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 2024.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1309.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Philipp Nazari and T. Konstantin Rusch. LinearAttentionPruning. GitHub repository, 2026a. URL https://github.com/camail-official/LinearAttentionPruning. Official code repository for "The Key to State Reduction in Linear Attention: A Rank-based Perspective".

Philipp Nazari and T. Konstantin Rusch. The key to state reduction in linear attention: A rank-based perspective. *arXiv preprint arXiv:2602.04852*, 2026b. URL https://arxiv.org/abs/2602.04852.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1144.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849, 2024. doi: 10.52202/079017-0970.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=y6SqbJfCSk.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-2124.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen Attention: Attention in low-rank space for KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.899.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via Householder products. In *Advances in Neural Information Processing Systems*, volume 38, pp. 153738–153782, 2025.

Ao Sun, Hongtao Zhang, Heng Zhou, Yixuan Ma, Yiran Qin, Tongrui Su, Yan Liu, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, and Renqing He. State rank dynamics in linear attention LLMs. *arXiv preprint arXiv:2602.02195*, 2026. URL https://arxiv.org/abs/2602.02195.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to Transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. URL https://arxiv.org/abs/2307.08621.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19. Association for Computing Machinery, 2019. doi: 10.1145/3315508.3329973.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Songlin Yang and Yu Zhang. FLA: A Triton-based library for hardware-efficient implementations of linear attention mechanism. GitHub repository, 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523. PMLR, 2024a.

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, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4904fad153f6434a7bcf04465d4be2cc-Abstract-Conference.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472.

Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. LoRC: Low-rank compression for LLMs KV cache with a progressive compression strategy. *arXiv preprint arXiv:2410.03111*, 2024. URL https://arxiv.org/abs/2410.03111.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023.

Yuxuan Zhu, David H. Yang, Mohammad Mohammadi Amiri, Keerthiram Murugesan, Tejaswini Pedapati, and Pin-Yu Chen. OjaKV: Context-aware online low-rank KV cache compression. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 10161–10178. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.494.

Longwei Zou and Lin Zhong. KVBuffer: IO-aware serving for linear attention. *arXiv preprint arXiv:2605.19049*, 2026. URL https://arxiv.org/abs/2605.19049.

A Additional Details of FastRebase

A.1 Hybrid-State Recurrence and Lazy Decay

Section 3.2 shows that the hybrid representation is preserved by the linear-attention recurrence between rebases. Here, we provide additional details on the factorized recurrence for the evaluated models, including the implementation of delta-rule corrections and the lazy-decay scheme that reduces memory writes to the key-side low-rank factor U_t. These implementation details introduce no additional approximation beyond the periodic rebases.

Hybrid-State Recurrence. The diagonal-plus-rank-one transition $A_t = D_t + b_t a_t^\top$ in (2) leaves W_{t-1} and the existing buffered values V_{t-1}^{res} unchanged while transforming U_{t-1} and the buffered keys K_{t-1}^{res} as in (8). The current key–value pair is then appended to the residual buffer. For decay-only models, where $b_t a_t^\top = 0$, this reduces to applying the scalar or diagonal decay D_t to U_{t-1} and K_{t-1}^{res}. In contrast, delta-rule models additionally apply the rank-one correction $b_t a_t^\top$, which erases part of the previously stored associations before the current association is written.

Delta-Rule Specialization. In DeltaNet and GDN, the delta-rule update writes the correction along the same key direction as the new association. Specifically, $b_t = k_t$ in (1)–(2). The state update can therefore be rewritten as

$$\begin{aligned} H_t &= (D_t + k_t a_t^\top) H_{t-1} + k_t v_t^\top \\ &= D_t H_{t-1} + k_t \big(a_t^\top H_{t-1} + v_t^\top\big) \\ &= D_t H_{t-1} + k_t \widetilde{v}_t^\top, \qquad \text{where } \widetilde{v}_t = v_t + H_{t-1}^\top a_t. \end{aligned} \tag{17}$$

Using the hybrid representation, the corrected value can be computed directly from the factors as

$$\widetilde{v}_t = v_t + W_{t-1}(U_{t-1}^\top a_t) + V_{t-1}^{\text{res}}\big((K_{t-1}^{\text{res}})^\top a_t\big). \tag{18}$$

The recurrence then becomes

$$\begin{aligned} U_t &= D_t U_{t-1}, & W_t &= W_{t-1}, \\ K_t^{\text{res}} &= \begin{bmatrix} D_t K_{t-1}^{\text{res}} & k_t \end{bmatrix}, & V_t^{\text{res}} &= \begin{bmatrix} V_{t-1}^{\text{res}} & \widetilde{v}_t \end{bmatrix}. \end{aligned} \tag{19}$$

In this form, the rank-one correction is absorbed into the value of the newly appended association. Decay-only and delta-rule models therefore share the same hybrid-state update form: the existing value-side components remain unchanged, while only the decay D_t is applied to U_{t-1} and K_{t-1}^{res}.

Lazy Decay. We further reduce memory writes to U_t by accumulating the decay factors separately rather than applying D_t to U_t at every step. Let τ denote the most recent rebase step and define the accumulated decay as

$$\Gamma_t = D_t D_{t-1} \cdots D_{\tau+1}. \tag{20}$$

Between rebases, the logical key-side factor is therefore $U_t = \Gamma_t U_\tau$, while U_τ remains unchanged in memory. Because each D_t is scalar or diagonal, we instead absorb the accumulated decay into q_t for output computation and into a_t for the value correction in delta-rule models:

$$q_t^\top U_t = (\Gamma_t q_t)^\top U_\tau, \qquad U_{t-1}^\top a_t = U_\tau^\top (\Gamma_{t-1} a_t). \tag{21}$$

We incorporate the accumulated decay only when forming the merged history at each rebase, then reset the decay accumulator.

Lazy decay is applied only to U_t. Unlike U_t, buffered keys are inserted at different steps and therefore require different accumulated decays. Although scaling each new key by the inverse accumulated decay at its insertion step allows all buffered keys to share a single global decay factor, we found this rescaling to be numerically unstable when the accumulated decay becomes small. A stable alternative is to retain the insertion-time decay for each buffered key, but for diagonal decay this requires an additional d_k-dimensional vector per key and largely eliminates the memory-traffic benefit. We therefore decay K_t^{res} explicitly at each step.

A.2 VALUE-SUBSPACE RECYCLING

We provide the derivation and implementation details of the value-subspace recycling update in Section 4.2. Recall the augmented factorization in (11), $M = [M_{\|}\ M_{\perp}]$ and $Z = [W\ Q]$. As defined in (13), the subspace update obtained by one-step subspace iteration initialized from W is

$$\Phi = \mathrm{orth}(GJ), \qquad \text{where } G = M^{\top}M \text{ and } J = \begin{bmatrix} I_r \\ 0 \end{bmatrix}.$$

Using the block structure of M,

$$GJ = \begin{bmatrix} M_{\|}^{\top}M_{\|} \\ M_{\perp}^{\top}M_{\|} \end{bmatrix}. \tag{22}$$

When $M_{\|}^{\top}M_{\|}$ is nonsingular, GJ can be written as

$$GJ = \begin{bmatrix} I_r \\ S \end{bmatrix} (M_{\|}^{\top}M_{\|}), \tag{23}$$

where S satisfies

$$S(M_{\|}^{\top}M_{\|}) = M_{\perp}^{\top}M_{\|}. \tag{24}$$

Because right multiplication by the nonsingular matrix $M_{\|}^{\top}M_{\|}$ does not change the column space,

$$\mathrm{col}(GJ) = \mathrm{col}\begin{bmatrix} I_r \\ S \end{bmatrix}, \tag{25}$$

which gives the block formulation in (14).

In practice, when solving $S(M_{\|}^{\top}M_{\|}) = M_{\perp}^{\top}M_{\|}$, the coefficient matrix $A = M_{\|}^{\top}M_{\|}$ can be rank-deficient or ill-conditioned. We therefore use the diagonally loaded matrix

$$A_{\epsilon} = A + \epsilon\alpha I_r, \qquad \alpha = \max\left(\frac{\mathrm{tr}(A)}{r}, 1\right), \tag{26}$$

with $\epsilon = 10^{-4}$ (corresponding to $\lambda = \epsilon\alpha$ in Section 4.2), and solve

$$SA_{\epsilon} = M_{\perp}^{\top}M_{\|} \tag{27}$$

using a Cholesky factorization of A_{ϵ} followed by two triangular solves. This yields a regularized approximation to the one-step subspace-iteration update that remains well-defined even when A is rank-deficient.

With the resulting S, we construct the subspace update Φ as

$$\Phi = \begin{bmatrix} I_r \\ S \end{bmatrix} N, \tag{28}$$

where $N \in \mathbb{R}^{r\times r}$ is a normalization matrix satisfying $\Phi^{\top}\Phi = I_r$. Since

$$\Phi^{\top}\Phi = N^{\top}(I_r + S^{\top}S)N,$$

we compute the Cholesky factorization

$$I_r + S^{\top}S = LL^{\top} \tag{29}$$

and set $N = L^{-\top}$. This normalization ensures $\Phi^{\top}\Phi = I_r$, reducing the orthonormalization to a Cholesky factorization. Together, the block formulation and Cholesky normalization avoid the materialization of the squared-Gram matrix $J^{\top}G^2J$ and its associated numerical sensitivity during orthonormalization. Instead, N is computed from $I_r + S^{\top}S$, which is positive definite even when S is rank-deficient.

Substituting the normalized basis Φ into the factor update in (15) gives

$$U' = M\Phi = (M_{\|} + M_{\perp}S)N, \qquad W' = Z\Phi = (W + QS)N. \tag{30}$$

Without diagonal loading, the block solve in (14) can be viewed as a least-squares problem:

$$S^{\top} = \arg\min_{X} \left\| M_{\perp} - M_{\|}X \right\|_F^2.$$

Thus, $S^{\top}$ gives the least-squares coefficients for approximating $M_{\perp}$ using $M_{\|}$, which in turn determine how the new value-side directions Q are incorporated into W' through $W + QS$.

Overall, given the augmented factors M and Z, the subspace update requires only the block products involving $M_{\|}$ and $M_{\perp}$, two $r \times r$ Cholesky factorizations, and triangular solves, without materializing the full $(r + p) \times (r + p)$ Gram matrix or performing an SVD (via eigendecomposition).

A.3 FastRebase Decoding Algorithm

Algorithm 1 summarizes one FastRebase decoding step, combining the hybrid-state recurrence (Appendix A.1) with value-subspace recycling (Appendix A.2). We describe a single linear attention head; the same procedure applies to each head at its calibrated rank. For simplicity, $\widetilde{v}_t$ denotes the value appended to the residual buffer: $\widetilde{v}_t = v_t$ for decay-only models and the corrected value defined in (18) for DeltaNet and GDN. In Algorithm 1, $|K|$ denotes the number of associations stored in K.

Algorithm 1 One FastRebase decoding step for a single linear attention head.

Input: Calibrated rank r, rebase interval p, residual keep size ℓ
Input: Hybrid state $(U, W, K^{\mathrm{res}}, V^{\mathrm{res}})$ with lazy-decay accumulator Γ
Input: Current decoding inputs $q_t, k_t, \widetilde{v}_t$ and D_t

$\Gamma \leftarrow D_t \Gamma$
$K^{\mathrm{res}} \leftarrow [D_t K^{\mathrm{res}} \;\; k_t]$
$V^{\mathrm{res}} \leftarrow [V^{\mathrm{res}} \;\; \widetilde{v}_t]$
$\widehat{o}_t \leftarrow W\big(U^\top(\Gamma q_t)\big) + V^{\mathrm{res}}\big((K^{\mathrm{res}})^\top q_t\big)$
if $|K^{\mathrm{res}}| = p + \ell$ **then**
  $[K^{\mathrm{old}} \;\; K^{\mathrm{keep}}] \leftarrow K^{\mathrm{res}}$ ▷ $|K^{\mathrm{old}}| = p,\ |K^{\mathrm{keep}}| = \ell$
  $[V^{\mathrm{old}} \;\; V^{\mathrm{keep}}] \leftarrow V^{\mathrm{res}}$
  $P \leftarrow W^\top V^{\mathrm{old}}$
  $E \leftarrow V^{\mathrm{old}} - WP$
  $(Q, R) \leftarrow \mathrm{QR}(E)$
  $M_{\|} \leftarrow \Gamma U + K^{\mathrm{old}} P^\top$
  $M_{\perp} \leftarrow K^{\mathrm{old}} R^\top$
  $A \leftarrow M_{\|}^\top M_{\|}$
  $\alpha \leftarrow \max(\mathrm{tr}(A)/r,\, 1)$
  $A_\epsilon \leftarrow A + \epsilon \alpha I_r$ ▷ $\epsilon = 10^{-4}$
  Solve $S A_\epsilon = M_{\perp}^\top M_{\|}$ for S ▷ Cholesky solve
  Factorize $I_r + S^\top S = L L^\top$ ▷ Cholesky factorization
  $N \leftarrow L^{-\top}$
  $U \leftarrow (M_{\|} + M_{\perp} S) N$
  $W \leftarrow (W + Q S) N$
  $K^{\mathrm{res}} \leftarrow K^{\mathrm{keep}}$
  $V^{\mathrm{res}} \leftarrow V^{\mathrm{keep}}$
  $\Gamma \leftarrow I$
end if
return $\widehat{o}_t,\ (U, W, K^{\mathrm{res}}, V^{\mathrm{res}}),\ \Gamma$

B Experimental Details

Model Checkpoints. We evaluate six pretrained 1.3B linear attention models using the publicly available checkpoints listed in Table 6. All checkpoints were pretrained with a 2K-token context.

Table 6: Pretrained model checkpoints used in our experiments.

Model	Checkpoint
GLA	`fla-hub/gla-1.3B-100B`
RetNet	`fla-hub/retnet-1.3B-100B`
HGRN2	`fla-hub/hgrn2-1.3B-100B`
Mamba2	`state-spaces/mamba2-1.3b`
DeltaNet	`fla-hub/delta_net-1.3B-100B`
GDN	`m-a-p/1.3B-100B-GatedDeltaNet-pure`

Tasks. For zero-shot evaluation, we use LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Easy/Challenge (Clark et al., 2018), and BoolQ (Clark et al., 2019). In Tables 1 and 12, acc_n denotes length-normalized accuracy. For the recall-intensive evaluation, we use the six tasks from Arora et al. (2024): DROP (Dua et al., 2019), FDA (Arora et al., 2023), NQ (Kwiatkowski et al., 2019), SQuAD (Rajpurkar et al., 2018), SWDE (Lockard et al., 2019), and TriviaQA (Joshi et al., 2017).

AdaptiveRebase. For AdaptiveRebase, we use the same settings as FastRebase in Section 5, except for the energy-retention target η. Unlike FastRebase, whose rank profile is calibrated once from dense recurrent states, AdaptiveRebase repeatedly estimates the retained energy from previously compressed states. We therefore use a higher energy-retention target to avoid excessive rank reduction. We set $\eta = 0.999$ for all models except DeltaNet, for which we use $\eta = 0.9999$. DeltaNet requires a stricter threshold because its non-decaying recurrent state is more sensitive to successive compression.

DRRQR. DRRQR (Nazari & Rusch, 2026b) selects a subset of the d_k channels for DeltaNet and GDN by applying strong rank-revealing QR factorization to Q/K activations collected from calibration samples. Since diagonal-decay models such as GLA have channel-wise decay activations along the d_k dimension, we also include these activations in the calibration inputs for channel selection. Because Q, K, and decay activations have different scales, we normalize each activation block by its RMS before applying DRRQR. For HGRN2, the key-equivalent input gate and the decay are jointly determined by the forget gate projection (Qin et al., 2024); we therefore use the input-gate activation in place of the key activation for channel selection. Meanwhile, directly removing the pruned coordinates would also require modifying model-specific channel-wise structures, such as RoPE in RetNet and the per-channel lower bounds for decay in HGRN2. To avoid introducing additional architectural modifications, we retain the original recurrent-state shape for these models and mask the projection channels corresponding to the pruned coordinates for quality evaluation. We calibrate DRRQR on 128 sequences of 2,048 tokens from FineWeb-Edu (Penedo et al., 2024), following the original setup (Nazari & Rusch, 2026a), without recovery fine-tuning.

Fixed Subspace. We construct a fixed value-side subspace that minimizes normalized reconstruction error over the dense calibration states for a given rank. For head h, let $H_{h,n}$ denote the dense recurrent state for calibration sequence n. We form the normalized Gram matrix

$$G_h^{\mathrm{cal}} = \frac{1}{N} \sum_{n=1}^{N} \frac{H_{h,n}^{\top} H_{h,n}}{\|H_{h,n}\|_F^2},$$

and set W_h^{fixed} to the leading r_h eigenvectors of G_h^{cal}, where r_h is the calibrated rank in Section 4.3. The normalization prevents calibration sequences with larger state norms from dominating the shared subspace, consistent with the normalized energy used for rank calibration. During inference, W_h^{fixed} remains unchanged: at each rebase, the older buffered values are projected onto this fixed subspace and their orthogonal components are discarded. We use the same calibration data and rebase settings as FastRebase in Section 5.

FastRebase. The rebase interval p controls the trade-off between hybrid-state recurrence and periodic rebasing. Larger p reduces the frequency of rebasing, but increases the average residual-buffer size and consequently the latency of the recurrence. Table 7 shows this trade-off at batch size 128. Increasing p from 8 to 16 nearly halves the rebase latency and reduces the total recurrence-path latency across all evaluated models. At $p = 32$, the rebase latency decreases further, while the recurrence latency increases; for Mamba2, this increase outweighs the additional rebase savings, resulting in a higher total latency. We use a shared rebase interval of $p = 16$ throughout the evaluation for consistency across models, although model-specific tuning could further reduce total latency.

Table 7: Latency breakdown (ms) over 64 decoding steps at batch size 128 on an H100 GPU.

	$p = 8$			$p = 16$			$p = 32$		
Model	Recurrence	Rebase	Total	Recurrence	Rebase	Total	Recurrence	Rebase	Total
GLA	55.33	77.43	132.76	58.55	39.60	98.15	65.39	24.93	90.32
RetNet	188.80	277.69	466.49	196.00	141.02	337.02	210.83	87.55	298.38
HGRN2	58.16	76.23	134.39	61.52	38.94	100.46	69.87	28.28	98.15
Mamba2	205.75	287.98	493.73	237.07	149.53	386.60	303.34	121.58	424.92
DeltaNet	98.07	135.33	233.40	101.02	68.44	169.46	107.58	45.45	153.03
GDN	162.23	172.04	334.27	167.11	86.65	253.76	178.76	54.37	233.13

Although a residual keep size $\ell > 0$ increases the average residual-buffer size, it can help preserve model quality by keeping recent key–value associations uncompressed at each rebase, as these associations have been less affected by subsequent state transitions. As shown in Table 8, increasing ℓ from 0 to 4 reduces WikiText-2 perplexity for five of the six models, with the largest improvement for HGRN2 from 17.390 to 17.257. However, further increasing ℓ to 8 provides only small additional reductions in perplexity. We therefore use $\ell = 4$ for all evaluated models, retaining the quality benefit of uncompressed recent associations while limiting the recurrence overhead. A nonzero ℓ also provides an asynchronous window of ℓ decoding steps for rebasing because each rebase excludes the most recent ℓ associations. This could allow periodic rebasing to overlap with subsequent recurrence computation by offloading either operation to the CPU or another accelerator. We leave such asynchronous execution for future work.

Table 8: WikiText-2 perplexity across residual keep sizes ℓ. Lower is better.

Model	$\ell = 0$	$\ell = 4$	$\ell = 8$
GLA	17.799	17.782	17.778
RetNet	18.314	18.309	18.304
HGRN2	17.390	17.257	17.222
Mamba2	13.318	13.313	13.313
DeltaNet	17.092	17.084	17.080
GDN	16.905	16.906	16.906
Mean	16.803	16.775	16.767

C Rank Calibration

C.1 Effect of the Energy-Retention Target

The energy-retention target η controls the rank assigned to each head in FastRebase. To examine the trade-off between the calibrated rank and model quality, we further evaluate $\eta = 0.995$ and 0.999. Table 9 reports the mean calibrated rank, average zero-shot accuracy, and WikiText-2 perplexity for $\eta \in \{0.995, 0.997, 0.999\}$ under the calibration setup in Section 5. The results show that zero-shot accuracy remains largely unchanged across all models, whereas WikiText-2 perplexity generally decreases with larger η, particularly for HGRN2 and DeltaNet. However, increasing η from 0.997 to 0.999 substantially increases the calibrated ranks for several models while providing comparatively modest additional quality gains. Overall, $\eta = 0.997$ retains most of the quality achieved at $\eta = 0.999$ with substantially lower calibrated ranks for several models.

Table 9: Effect of the energy-retention target η on calibrated rank and model quality. Mean r denotes the mean calibrated rank across layers and heads, and Acc. and ppl denote average zero-shot accuracy over the seven downstream tasks and WikiText-2 perplexity, respectively.

	$\eta = 0.995$			$\eta = 0.997$			$\eta = 0.999$		
Model	Mean r	Acc. ↑	ppl ↓	Mean r	Acc. ↑	ppl ↓	Mean r	Acc. ↑	ppl ↓
GLA	24.47	50.42	17.79	27.14	50.39	17.78	32.32	50.42	17.78
RetNet	57.05	49.95	18.31	62.06	50.00	18.31	69.32	49.93	18.31
HGRN2	16.96	52.14	17.35	19.82	52.20	17.26	26.20	52.24	17.16
Mamba2	9.58	59.51	13.33	11.07	59.34	13.31	14.24	59.34	13.30
DeltaNet	28.84	51.84	17.43	36.54	51.84	17.08	51.55	51.84	16.84
GDN	35.47	59.78	16.91	37.24	59.69	16.91	40.63	59.66	16.91

C.2 Effect of Per-Head Rank Allocation

The calibrated rank profile assigns different ranks across heads according to their singular value energy distributions. To examine the benefit of this per-head allocation, we compare the profile calibrated at $\eta = 0.997$ with a uniform profile of comparable mean rank. As shown in Table 10, the per-head profile achieves slightly higher zero-shot accuracy and lower WikiText-2 perplexity across all models, despite using lower mean ranks. These results show that per-head rank allocation preserves model quality more effectively than uniform rank assignment.

Table 10: Effect of per-head rank allocation on model quality at $\eta = 0.997$. For each model, the uniform profile assigns every head the ceiling of the mean calibrated rank.

	Per-head			Uniform		
Model	Mean r	Acc. ↑	ppl ↓	Mean r	Acc. ↑	ppl ↓
GLA	27.14	50.39	17.78	28.00	50.34	18.62
RetNet	62.06	50.00	18.31	63.00	49.83	18.43
HGRN2	19.82	52.20	17.26	20.00	52.12	17.88
Mamba2	11.07	59.34	13.31	12.00	59.10	13.92
DeltaNet	36.54	51.84	17.08	37.00	51.80	17.10
GDN	37.24	59.69	16.91	38.00	59.57	17.45

C.3 Calibration Robustness

Calibration-Set Size N. We next examine the sensitivity of the rank profile to the number of calibration sequences N. For each model and calibration-set size N, we construct rank profiles from 200 random subsets of the WikiText-2 training split and compute the mean absolute error (MAE) of the layer–head ranks relative to the reference profile calibrated using 128 sequences. As shown in Figure 5, both the rank-profile MAE and its variation across calibration subsets decrease consistently as N increases. At $N = 32$, the calibrated profiles are close to the 128-sequence reference: for every model, at least 75% of the sampled subsets have a rank-profile MAE below 1, with relatively small variation across subsets. We therefore use 32 calibration sequences throughout our evaluation.

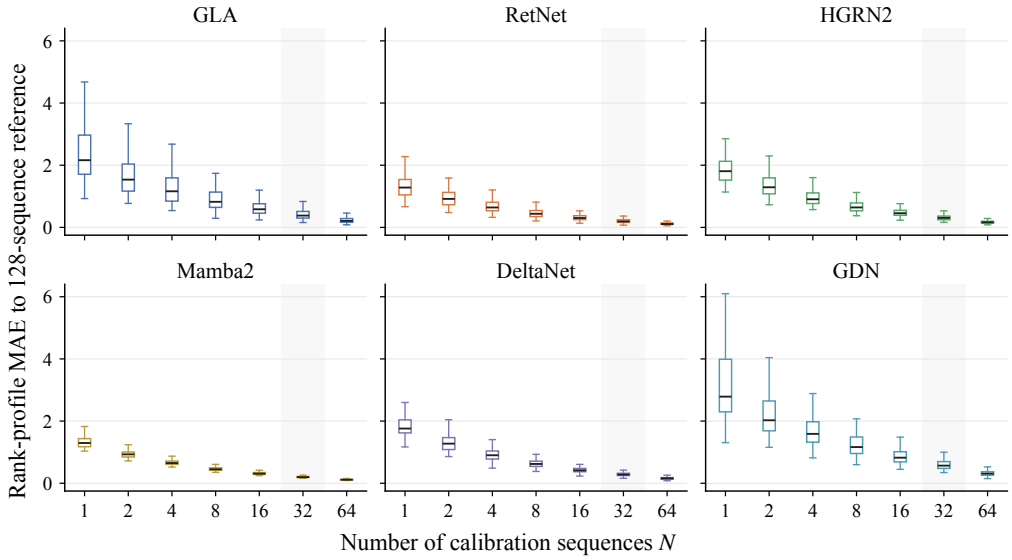

Figure 5: Sensitivity of the calibrated per-head rank profile to the number of calibration sequences. The shaded band indicates $N = 32$, the calibration-set size used throughout our evaluation.

Calibration Dataset. We further examine whether the calibrated rank and the resulting model quality depend on the choice of calibration dataset. Using the same calibration setup as in Section 5, we construct rank profiles from FineWeb-Edu and compare them with the WikiText-2 profiles used for the FastRebase evaluation. As shown in Table 11, the mean calibrated rank differs by less than one between the two datasets for every model, while the corresponding zero-shot accuracy and WikiText-2 perplexity remain largely unchanged. These results indicate that the choice of calibration dataset has little effect on the mean rank or resulting model quality.

Table 11: Effect of the calibration dataset on calibrated rank and model quality.

	WikiText-2			FineWeb-Edu		
Model	Mean r	Acc. ↑	ppl ↓	Mean r	Acc. ↑	ppl ↓
GLA	27.14	50.39	17.78	27.92	50.44	17.78
RetNet	62.06	50.00	18.31	62.11	49.86	18.31
HGRN2	19.82	52.20	17.26	19.59	52.19	17.28
Mamba2	11.07	59.34	13.31	11.06	59.28	13.32
DeltaNet	36.54	51.84	17.08	36.92	51.84	17.09
GDN	37.24	59.69	16.91	37.58	59.73	16.91

D Related Work

Online Subspace Updates. Incremental SVD methods incorporate successive low-rank updates by augmenting the orthonormal bases of the current factorization and solving a smaller decomposition problem (Brand, 2002; 2006). Our adaptive-rank rebase follows a similar augmentation strategy, but linear-attention state transitions generally do not preserve the orthonormality of the key-side basis. Directly maintaining the recurrent state in SVD form would therefore require reorthogonalizing the transformed key-side basis after each state transition. We instead leave the key-side factor unconstrained and maintain an orthonormal value-side basis, allowing state transitions to be absorbed into the key-side factor without reorthogonalization. Building on this formulation, FastRebase determines per-head ranks through offline calibration and replaces the SVD-based subspace update with value-subspace recycling.

Context Compression for Autoregressive Decoding. The growing key–value (KV) cache of Transformer-based language models has motivated a broad range of inference-time compression approaches, including token eviction (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024) and quantization (Liu et al., 2024; Hooper et al., 2024; He et al., 2024). A complementary line of work exploits redundancy along the feature dimension through low-rank representations. Eigen Attention performs attention in a reduced subspace (Saxena et al., 2024), while Palu and LoRC apply low-rank factorizations to the key and value projection matrices to cache lower-dimensional representations (Chang et al., 2025; Zhang et al., 2024). MatryoshkaKV instead learns orthogonal projections with adaptive compression rates across layers and heads (Lin et al., 2025). These low-rank compression methods determine their low-rank factorizations or projection subspaces before inference and keep them fixed at runtime. In contrast, OjaKV adapts the projection subspaces online to account for sequence-dependent variation in key and value representations (Zhu et al., 2026). However, the additional cost of online subspace updates leads to higher decoding latency than the full-KV baseline in their experiments.

Unlike Transformers, which maintain a token-wise KV cache, linear-attention models summarize context into a fixed-size recurrent state. Prior work reduces recurrent-state size through static pruning, such as removing channels along the shared key–query dimension d_k (Nazari & Rusch, 2026b) or entire heads (Sun et al., 2026). Other approaches reduce recurrent-state writes through deferred dense-state updates or materialization while leaving the state itself uncompressed and retaining the dense-state read cost (Zou & Zhong, 2026; Lin et al., 2026). In contrast, FastRebase maintains the evolving recurrent state itself in a hybrid low-rank representation and adapts its subspace online. This reduces both recurrent-state computation and memory traffic by performing state updates and output computation directly on the compressed representation. To our knowledge, FastRebase is the first method to demonstrate end-to-end decoding speedups for linear attention models through online low-rank compression of recurrent states.

E Additional Quality Results

Detailed Zero-Shot Results. Table 12 provides the detailed results corresponding to the quality ablation in Table 3. The fixed-subspace variant substantially degrades zero-shot accuracy across the evaluated models, with particularly large drops on LAMBADA. In contrast, the effects of offline rank calibration and subspace recycling remain small across individual zero-shot tasks: relative to AdaptiveRebase, the final FastRebase configuration changes accuracy by at most 0.55 percentage points across all model–task pairs, with both small increases and decreases.

Table 12: Detailed zero-shot results and WikiText-2 perplexity for the FastRebase quality ablation.

Model	Method	Wiki. ppl ↓	LMB. acc ↑	PIQA acc_n ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc_n ↑	ARC-c acc_n ↑	BoolQ acc ↑	Avg. ↑
GLA	AdaptiveRebase	17.77	46.11	70.62	48.87	53.04	48.70	27.30	57.74	50.34
	+ Rank calibration	17.76	46.17	70.73	48.92	52.72	48.70	27.73	57.58	50.36
	+ Fixed subspace	27.74	9.02	68.99	45.27	53.91	44.70	26.79	57.83	43.79
	+ Subspace recycling	17.78	45.97	70.78	48.90	52.96	48.65	27.73	57.74	50.39
RetNet	AdaptiveRebase	18.31	42.87	70.46	48.08	53.43	49.62	26.96	58.41	49.98
	+ Rank calibration	18.30	42.91	70.46	48.15	53.67	49.45	26.88	58.75	50.04
	+ Fixed subspace	25.97	10.54	69.37	45.65	54.22	46.80	25.94	60.03	44.65
	+ Subspace recycling	18.31	42.89	70.46	48.13	53.75	49.41	26.79	58.56	50.00
HGRN2	AdaptiveRebase	17.24	50.01	70.62	51.73	53.28	52.06	28.33	59.60	52.23
	+ Rank calibration	17.16	50.20	70.84	51.67	53.35	52.15	28.33	59.69	52.32
	+ Fixed subspace	22.92	20.36	68.82	46.73	51.38	47.52	26.71	58.01	45.65
	+ Subspace recycling	17.26	50.22	70.78	51.72	52.88	52.15	28.16	59.51	52.20
Mamba2	AdaptiveRebase	13.36	64.72	73.50	59.81	60.77	59.97	33.45	63.30	59.36
	+ Rank calibration	13.28	64.76	73.34	59.97	60.93	60.14	33.45	63.36	59.42
	+ Fixed subspace	17.50	29.59	72.80	55.95	59.67	57.24	33.02	58.23	52.36
	+ Subspace recycling	13.31	64.53	73.56	59.93	60.69	60.27	33.53	62.84	59.34
DeltaNet	AdaptiveRebase	17.00	48.17	70.78	50.39	54.54	51.30	26.11	61.10	51.77
	+ Rank calibration	17.03	48.32	71.11	50.54	54.30	51.35	26.19	60.86	51.81
	+ Fixed subspace	22.08	15.33	68.72	46.47	54.22	47.14	25.85	61.41	45.59
	+ Subspace recycling	17.08	48.46	70.89	50.54	53.99	51.47	26.28	61.25	51.84
GDN	AdaptiveRebase	16.92	51.27	73.99	60.24	61.72	67.76	40.53	61.38	59.56
	+ Rank calibration	16.88	51.41	74.21	60.21	61.96	67.72	40.53	61.71	59.68
	+ Fixed subspace	20.18	24.96	72.47	57.13	60.14	64.10	37.37	59.36	53.65
	+ Subspace recycling	16.91	51.37	74.16	60.20	61.80	67.76	40.78	61.74	59.69

Position-Wise PG-19 Perplexity. To further examine local quality variations with context length, we report position-wise perplexity on PG-19 in Figure 6. We partition each sequence into non-overlapping 1K-token intervals and compute perplexity within each interval; thus, each point reflects language-modeling quality within the corresponding interval rather than cumulatively over all preceding tokens as in Figure 4. The position-wise results show that FastRebase closely tracks the dense baseline across all positions for GLA and RetNet, while HGRN2 and GDN exhibit modest gaps that remain relatively stable with context length. Excluding Mamba2, FastRebase remains within 3.8% of the dense baseline in every 1K-token interval. As discussed in Section 5, Mamba2 remains an exception, with perplexity increasing sharply beyond its training context. For DeltaNet, the position-wise perplexity of the dense baseline increases with context length, while FastRebase closely tracks this trend throughout the evaluated context. Overall, the position-wise results are consistent with the cumulative trends in Figure 4.

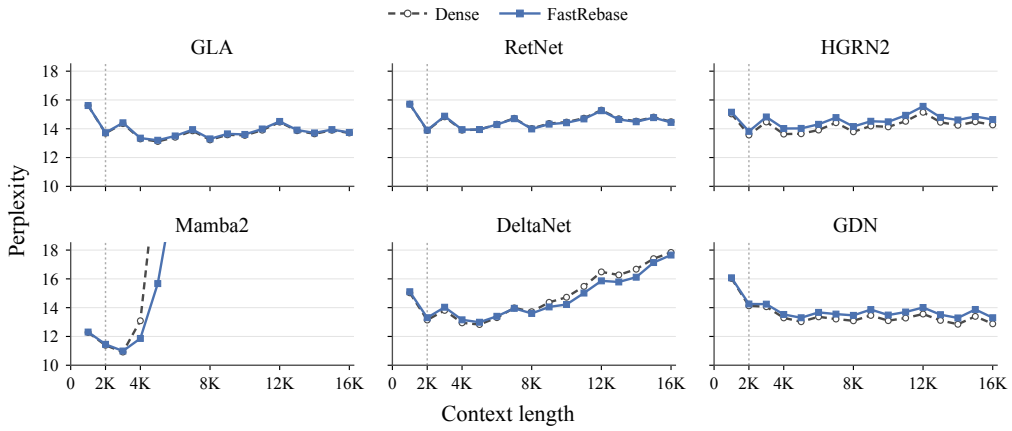

Figure 6: Position-wise perplexity on 64 sequences of 16K tokens from PG-19, computed over non-overlapping 1K-token intervals. For Mamba2, values exceeding the y-axis limit are not shown.

Detailed Results on Recall-Intensive Tasks. Table 13 provides the per-task results corresponding to the recall-intensive evaluation in Table 5. The degradation under FastRebase is task-dependent, with larger gaps on FDA and SWDE and generally smaller changes on the remaining tasks. One-shot SVD with the $\eta = 0.997$ rank profile substantially narrows these gaps for most models, while DeltaNet benefits further from the higher-rank profile at $\eta = 0.999$.

Table 13: Detailed accuracy results on the six recall-intensive tasks (Arora et al., 2024). FastRebase uses the rank profile calibrated at $\eta = 0.997$. For One-shot SVD, a truncated SVD is applied to the dense prefill state before FastRebase generation. TQA denotes TriviaQA.

Model	Method	η	DROP	FDA	NQ	SQuAD	SWDE	TQA	Avg.
GLA	Dense	–	18.74	24.43	22.43	35.30	30.46	55.98	31.22
	FastRebase	0.997	19.36	13.71	20.18	34.53	27.09	55.51	28.40
	One-shot SVD	0.997	19.36	20.35	22.01	35.14	29.43	55.75	30.34
		0.999	19.41	20.16	21.79	35.20	29.33	55.86	30.29
RetNet	Dense	–	19.98	19.89	16.28	34.53	26.90	53.44	28.50
	FastRebase	0.997	19.60	18.17	15.52	33.89	24.93	52.07	27.36
	One-shot SVD	0.997	20.08	19.80	15.93	34.20	26.71	53.50	28.37
		0.999	20.22	19.89	15.81	34.06	26.71	53.44	28.36
HGRN2	Dense	–	19.50	12.53	18.97	32.89	23.06	54.92	26.98
	FastRebase	0.997	19.21	7.08	17.17	32.25	19.03	54.03	24.80
	One-shot SVD	0.997	19.79	11.81	18.72	32.72	22.77	54.62	26.74
		0.999	19.26	12.08	18.78	32.75	22.68	54.92	26.75
Mamba2	Dense	–	21.99	35.42	28.19	42.06	47.24	63.15	39.68
	FastRebase	0.997	20.70	17.44	24.17	40.11	33.36	62.44	33.04
	One-shot SVD	0.997	21.56	32.06	27.43	41.95	46.02	63.63	38.78
		0.999	21.90	32.33	27.27	42.16	46.11	62.86	38.77
DeltaNet	Dense	–	20.56	38.69	24.39	35.30	36.74	56.58	35.38
	FastRebase	0.997	19.17	11.26	22.05	31.47	26.90	55.27	27.69
	One-shot SVD	0.997	20.36	24.52	23.19	34.36	33.74	56.10	32.05
		0.999	20.94	35.42	24.74	35.24	36.36	56.16	34.81
GDN	Dense	–	22.52	39.24	24.77	40.28	33.27	64.63	37.45
	FastRebase	0.997	22.23	27.34	22.74	39.57	26.99	64.16	33.84
	One-shot SVD	0.997	22.42	35.06	24.07	40.24	32.24	65.05	36.51
		0.999	22.33	35.24	23.92	40.31	32.24	64.81	36.48

F Additional Performance Results

AdaptiveRebase Latency Breakdown. Table 14 further breaks down the rebase overhead of AdaptiveRebase reported in Table 2. Across all six models, the SVD-based subspace update accounts for 99.3–99.8% of the total rebase latency, while the matrix multiplications, QR factorization, and remaining operations contribute only a small fraction. These results show that the large AdaptiveRebase overhead reported in Section 5 is dominated by the online SVD, which FastRebase replaces with value-subspace recycling.

Table 14: Latency breakdown (ms) of periodic rebasing in AdaptiveRebase over 64 decoding steps at batch size 32 on an H100 GPU. The SVD-based subspace update is implemented via eigendecomposition (Section 4.1).

Model	Matmul	QR	Other	SVD	Total	SVD (%)
GLA	4.88	7.78	21.35	4846.83	4880.84	99.3
RetNet	9.33	11.53	38.59	16169.16	16228.61	99.6
HGRN2	5.30	7.84	23.47	21105.20	21141.81	99.8
Mamba2	24.19	26.69	77.21	44669.08	44797.17	99.7
DeltaNet	6.49	7.85	27.32	22816.42	22858.08	99.8
GDN	7.43	7.67	31.87	11054.01	11100.98	99.6

Detailed Batch-Wise Performance. Detailed performance results across batch sizes are reported in Table 15. The end-to-end benefit of FastRebase increases with batch size for two complementary reasons. First, the reduced per-sequence recurrent-state traffic allows FastRebase to benefit from additional concurrency before reaching memory-bandwidth saturation, leading to higher recurrence speedups at larger batch sizes. Second, as discussed in Section 2.2, weight accesses are amortized across the batch, whereas recurrent-state memory traffic scales with batch size. Consequently, linear-attention recurrence accounts for a larger fraction of decoding latency as batch size increases, allowing the recurrence speedups to translate into greater end-to-end gains. At batch size 512, FastRebase achieves end-to-end decoding speedups of 1.31–3.23× across the six models.

Table 15: FastRebase speedups relative to the dense baselines across batch sizes on an H100 GPU.

		Batch size				
Model	Scope	32	64	128	256	512
GLA	Recurrence (w/o rebase)	4.91×	6.54×	7.93×	8.59×	8.97×
	Recurrence (w/ rebase)	2.67×	3.83×	4.73×	5.17×	5.34×
	End-to-end decode	1.37×	1.76×	2.31×	2.87×	3.23×
RetNet	Recurrence (w/o rebase)	3.98×	4.50×	4.77×	4.87×	4.92×
	Recurrence (w/ rebase)	2.23×	2.57×	2.77×	2.83×	2.88×
	End-to-end decode	1.53×	1.88×	2.19×	2.38×	2.51×
HGRN2	Recurrence (w/o rebase)	2.66×	3.40×	3.73×	3.97×	4.04×
	Recurrence (w/ rebase)	1.63×	2.06×	2.28×	2.42×	2.45×
	End-to-end decode	1.15×	1.28×	1.47×	1.64×	1.72×
Mamba2	Recurrence (w/o rebase)	3.54×	3.86×	3.95×	3.96×	4.02×
	Recurrence (w/ rebase)	2.15×	2.34×	2.42×	2.44×	2.47×
	End-to-end decode	1.35×	1.56×	1.78×	1.92×	2.00×
DeltaNet	Recurrence (w/o rebase)	1.53×	1.78×	2.27×	2.51×	2.69×
	Recurrence (w/ rebase)	1.02×	1.14×	1.36×	1.52×	1.56×
	End-to-end decode	1.01×	1.08×	1.18×	1.28×	1.31×
GDN	Recurrence (w/o rebase)	1.92×	2.24×	2.75×	3.10×	3.37×
	Recurrence (w/ rebase)	1.19×	1.48×	1.81×	1.99×	2.14×
	End-to-end decode	1.08×	1.19×	1.42×	1.61×	1.74×