SketchSSM: Write to the Full State, Read from a Compact Sketch

Anonymous authors
Paper under double-blind review

Abstract

Hybrid-attention models replace most softmax attention layers with linear attention, reducing KV-cache growth and enabling larger decode batches where recurrent-state access becomes a major bottleneck. ReplaySSM amortizes state updates by buffering keys and values, but each new query still requires a full-state read even though the state remains unchanged between state updates. We observe that low-rank state-weighted query approximation accurately preserves state-read outputs. Although future queries are unknown, the basis vectors used to approximate them can be fixed offline. Based on this observation, we introduce SketchSSM, which preserves full-state updates while approximating reads. At each state update, SketchSSM reads the full state once to precompute outputs for these basis vectors, storing them in a compact sketch. Each subsequent decode step combines the sketch vectors with query-dependent coefficients to reconstruct the output without a full-state read. Across four Mamba-2-, GDN-, and KDA-based models, SketchSSM reduces state-access traffic by approximately 10× while largely preserving average accuracy across four decode benchmarks and recall on four RULER retrieval tasks. On one NVIDIA B300, linear-attention kernel speedups over the standard vLLM baseline reach 7.78×, 5.22×, and 5.20× for Mamba-2, GDN, and KDA, respectively, with up to 2.64× higher decode throughput on Nemotron 3 Super.

1 Introduction

Recent hybrid attention models replace most softmax attention layers with linear attention while retaining high accuracy (NVIDIA et al., 2026; Qwen Team, 2026b; GLM-5-Team et al., 2026). Replacing KV caches with fixed-size recurrent states reduces cache memory, enabling larger decode batches with higher throughput. However, in this large-batch regime, reading and updating the entire recurrent state every decode step becomes a major decode bottleneck as shown in Fig. 1. A common approach to reducing this traffic is to reduce the recurrent state size. Quantization lowers state precision (Tianqi et al., 2025; Chiang et al., 2025; Zhang et al., 2026), while pruning removes state entries (Menezes & Kyrillidis, 2026; Nazari & Rusch, 2026). However, reducing the state size introduces state approximation errors that propagate and accumulate through subsequent decode steps, resulting in substantial accuracy degradation, as we show in Section 5.1.

To reduce *state-write* traffic associated with state updates, ReplaySSM (Liou & Dao, 2026) and KVBuffer (Zou & Zhong, 2026) buffer the much smaller keys and values and apply their accumulated updates to the full state once every several decode steps, thereby amortizing state writes. However, *state-read* traffic remains unchanged: computing each output vector requires multiplying the new query by the full state. Therefore, a full state-read occurs at every decode step even though the state remains fixed until the buffered updates are applied. As shown in Fig. 1, linear attention remains a major bottleneck even after applying ReplaySSM due to this repeated state-read overhead.

We identify a key opportunity to avoid repeated full-state reads while accurately reconstructing state-read outputs. Although the query changes at every decode step, the state-read output can be approximated using fixed query basis vectors with reconstruction adapted to the current state and query. We observe that even a small number of query basis vectors is sufficient to reconstruct outputs with low error, as shown in Fig. 2.

SketchSSM exploits this opportunity by using a single full-state read at each update step to multiply the state by all fixed query basis vectors and precompute their outputs. We store these output vectors

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.

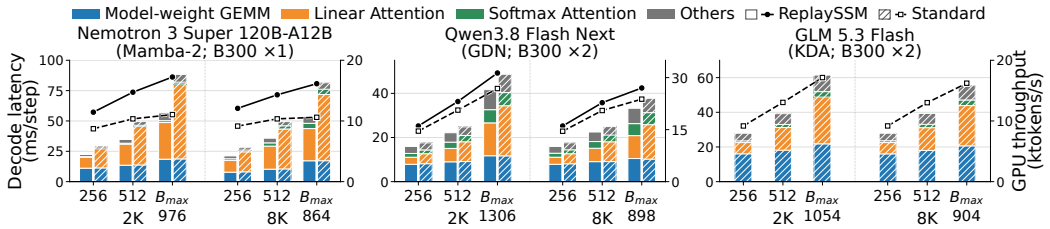

Fig. 1: Decode latency breakdown and decode throughput at batch sizes 256, 512, and $B_{\max}$ for 2K and 8K contexts. At $B_{\max}$, linear attention accounts for 41–70% of decode latency with Standard, and 31–53% with ReplaySSM. $B_{\max}$ denotes the largest tested batch size that fits within GPU memory for each model and context length. Throughput generally increases as batch size grows toward $B_{\max}$. GLM uses only Standard because KDA lacks an official ReplaySSM implementation.

in a compact matrix called a *sketch*. At each decode step between updates, SketchSSM approximates the new query's output as a linear combination of sketch vectors with query-dependent coefficients, without reading the full state. With G fixed query basis vectors and a K-dimensional query, the sketch is only G/K the size of the full state, where $G \ll K$ (Equation 7). Reading this compact sketch instead of the full state substantially reduces state-read traffic, as shown in Fig. 3. By approximating only reads while preserving exact full-state updates, SketchSSM avoids propagating state compression errors through updates.

Across four Mamba-2-, GDN-, and KDA-based models, SketchSSM reduces state-access traffic by approximately 10× while largely preserving average accuracy across four benchmarks. In vLLM on one NVIDIA B300 GPU, SketchSSM achieves linear-attention kernel speedups of up to 7.78×, 5.22×, and 5.20× over the standard full-state baseline for Mamba-2, GDN, and KDA, respectively, and up to 2.64× higher decode throughput on Nemotron 3 Super 120B-A12B (Section 5.2). Our primary contributions can be summarized as follows:

- We propose a training-free design that preserves the full state for updates while approximating only state-read outputs, reducing read traffic without propagating state-compression errors(Section 3.3).
- We design a sketch of precomputed output vectors that can be constructed with a single full-state read. We improve accuracy by allocating sketch ranks differently across state heads (Section 4).
- We demonstrate substantially better accuracy–traffic tradeoffs than state pruning and quantization, and accelerate linear attention across Mamba-2, GDN, and KDA with optimized kernels (Section 5).

2 Related Work

Recurrent Linear Attention and Hybrid Models. Linear attention computes recurrent updates and reads over a fixed-size matrix state, keeping storage independent of context length (Katharopoulos et al., 2020; Yang et al., 2024a). Mamba-2 connects structured SSMs and linear attention through state-space duality, while GDN and KDA extend this recurrence with delta-rule erasure and more expressive gating (Dao & Gu, 2024; Yang et al., 2025; Zhang et al., 2025; Schlag et al., 2021). Recent hybrid-attention models interleave these recurrent layers with fewer softmax or sparse-attention layers, limiting context-growing KV caches while retaining token-level retrieval (Lieber et al., 2024; NVIDIA et al., 2026; Qwen Team, 2026a; GLM-5-Team et al., 2026).

Efficient Recurrent-state Inference. The performance bottleneck of linear attention is state access: reading and updating state at every decode step. State quantization (Chiang et al., 2025; Tianqi et al., 2025; Zhang et al., 2026) and pruning (Menezes & Kyrillidis, 2026; Nazari & Rusch, 2026) reduce traffic by shrinking the state, but limit information capacity (Arora et al., 2024; Qin et al., 2024; Pan et al., 2026) and propagate compression errors through updates. Recently, ReplaySSM (Liou & Dao, 2026) and KVBuffer (Zou & Zhong, 2026) reduce write traffic by buffering keys and values and applying accumulated updates at each flush, leaving read traffic unchanged. SketchSSM complements buffered updates by reducing the remaining read traffic with a sketch.

Matrix Sketching. Matrix sketching constructs compact representations for approximate matrix computations (Halko et al., 2011; Woodruff, 2014), including randomized projections, streaming sketches, and learned sketches (Liberty, 2013; Indyk et al., 2019). In LLM inference, low-rank methods use activation statistics to approximate weights (Yuan et al., 2025; Wang et al., 2025), or perform low-rank attention to reduce KV-cache storage (Saxena et al., 2024). SketchSSM builds on the range-finding framework of Halko et al. (2011), as detailed in Appendix A.2.1. It sketches the state with an offline-calibrated query basis and reconstructs outputs with coefficients minimizing output error for the current state and query.

3 MOTIVATION

3.1 STATE-ACCESS BOTTLENECK IN HYBRID-ATTENTION DECODING

Recent hybrid-attention models substantially reduce decode latency while retaining high accuracy (NVIDIA et al., 2026; Qwen Team, 2026b; GLM-5-Team et al., 2026). Most layers replace context-growing KV caches with fixed-size recurrent states, allowing larger decode batches. Fig. 1 profiles this large-batch regime at 2K- and 8K-token inputs on Nemotron 3 Super (Mamba-2, one B300), Qwen3.8 Flash Next (GDN, two B300s), and GLM 5.3 Flash (KDA, two B300s), all using NVFP4 weights. It separates each decode step into model-weight GEMMs, linear attention, softmax attention, and other costs. Softmax attention takes only a small fraction of decode latency because these models replace most softmax attention layers with linear attention and optimize the remaining ones. Nemotron 3 Super uses an 8-bit KV cache, Qwen3.8 Flash Next uses Qwen Sparse Attention (QSA), and GLM 5.3 Flash combines Multi-head Latent Attention (MLA) with DeepSeek Sparse Attention (DSA); QSA and DSA bound per-step KV reads during long-context serving.

Linear attention emerges as the dominant decode bottleneck, as each layer reads and updates an independent recurrent state for every request at each decode step. With Standard execution, which updates the full recurrent state at every decode step, linear attention accounts for 41–70% of decode latency at $B_{\max}$ across the three models. ReplaySSM buffers keys and values over a window of decode steps to amortize state-write traffic and lower linear attention latency, but leaves state-read traffic unchanged. Further reducing this bottleneck therefore requires reducing state-read traffic.

3.2 WHY EVERY DECODE STEP READS THE FULL STATE

Reformulation: Decomposing the State Read. To examine this state-read traffic, consider a W-step buffering window starting from state $S_0 \in \mathbb{R}^{K\times V}$, with K key channels and V value channels. Let D_t denote the decay matrix and $D_{a:b} := D_b \cdots D_a$ its accumulated product. The state recurrence and output decomposition used by ReplaySSM (Liou & Dao, 2026) and KVBuffer (Zou & Zhong, 2026) can be written as

$$S_t = D_t S_{t-1} + k_t w_t^\top = D_{1:t} S_0 + \sum_{s=1}^{t} \big(D_{s+1:t} k_s\big) w_s^\top \quad \text{(update)} \quad (1)$$

$$o_t = S_t^\top q_t = {o'}_t^{\text{state}} + {o'}_t^{\text{buffer}}, \quad {o'}_t^{\text{state}} := S_0^\top D_{1:t}^\top q_t, \quad {o'}_t^{\text{buffer}} := \sum_{s=1}^{t} \big\langle D_{s+1:t} k_s, q_t \big\rangle w_s \quad \text{(read)} \quad (2)$$

Here, $q_t, k_t \in \mathbb{R}^K$ are the query and key, $v_t, w_t \in \mathbb{R}^V$ are the value and write vector, and β_t is the write gate. The state term ${o'}_t^{\text{state}}$ is computed directly from S_0. While ${o'}_t^{\text{buffer}}$ is computed from the buffer, its write vectors $w_t = \beta_t(v_t - S_{t-1}^\top D_t^\top k_t)$ require reading S_0 in delta-rule models. To isolate the effect of reading S_0, we collect all state dependence into a single output term. To this end, we absorb the delta-rule erase into the transition: $M_t = (I - \beta_t k_t k_t^\top) D_t$ for GDN and KDA, and $M_t = D_t$ for Mamba-2. We define $M_{a:b} := M_b \cdots M_a$ and the effective query $\tilde{q}_t := M_{1:t}^\top q_t$. An adapted WY representation (Yang et al., 2024b) evaluates $\tilde{q}_t$ through parallel accumulation over buffered tokens (Appendix C.1). The update and read can then be written as

$$S_t = M_t S_{t-1} + \beta_t k_t v_t^\top = M_{1:t} S_0 + \sum_{s=1}^{t} \beta_s \big(M_{s+1:t} k_s\big) v_s^\top \quad \text{(update)} \quad (3)$$

$$o_t = S_t^\top q_t = o_t^{\text{state}} + o_t^{\text{buffer}}, \quad o_t^{\text{state}} := S_0^\top \tilde{q}_t, \quad o_t^{\text{buffer}} := \sum_{s=1}^{t} \beta_s \big\langle M_{s+1:t} k_s, q_t \big\rangle v_s \quad \text{(read)} \quad (4)$$

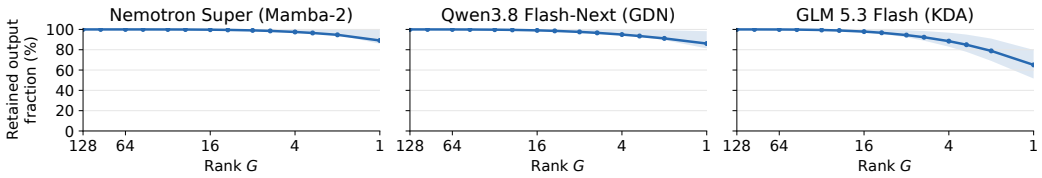

Fig. 2: Retained state-read output fraction ρ_G versus rank G on Nemotron Super (Mamba-2), Qwen3.8 Flash-Next (GDN), and GLM 5.3 Flash (KDA), using a query basis fixed offline and state-dependent coefficients. Lines show the mean across heads; shaded regions denote the interquartile range. At $G = 4$, the mean retained output fraction is 88.4–97.5% across the three models.

All dependence on S_0 is now confined to o_t^{state}, while o_t^{buffer} is computed exactly from raw buffered inputs. Thus, computing $o_t^{\text{state}} = S_0^\top \tilde{q}_t$ is the source of full-state read traffic at each decode step.

Source of State Read: Changing Queries over the Same State. Although S_0 remains fixed throughout the window, the effective query $\tilde{q}_t = M_{1:t}^\top q_t$ changes at each decode step as both the query q_t and the accumulated transition $M_{1:t}$ change. Directly evaluating $o_t^{\text{state}} = S_0^\top \tilde{q}_t$ thus requires reading all $K \times V$ state entries for each query. The challenge is to answer these changing queries without repeatedly reading the same full state.

3.3 Replacing Full-State Reads with a Compact Sketch

Observation: A Small, Fixed Query Basis for Accurate State Reads. We observe that a small, fixed query basis can accurately reconstruct state-read outputs for changing queries, using state- and query-dependent coefficients. Specifically, an effective query $\tilde{q}_t \in \mathbb{R}^K$ can be approximated as Ωc_t in a G-dimensional subspace, where $\Omega \in \mathbb{R}^{K \times G}$ contains G query basis vectors and $c_t \in \mathbb{R}^G$ gives the coefficients. Here, the goal is to preserve the state-read output o_t^{state}, rather than the effective query itself. We therefore measure the approximation quality through the output reconstruction $\hat{o}_t^{\text{state}} = S_0^\top \Omega c_t$ and define its error as

$$\mathcal{E}_t^{\text{state}} := \left\| o_t^{\text{state}} - \hat{o}_t^{\text{state}} \right\|_2^2 = \left\| S_0^\top \tilde{q}_t - S_0^\top \Omega c_t \right\|_2^2, \qquad S_0^\top \tilde{q}_t \in \mathbb{R}^V. \tag{5}$$

Expanding the squared norm, we can rewrite the output error as a query approximation error weighted by the state Gram matrix $S_0 S_0^\top$:

$$\mathcal{E}_t^{\text{state}} = (\tilde{q}_t - \Omega c_t)^\top S_0 S_0^\top (\tilde{q}_t - \Omega c_t) = \left\| (S_0 S_0^\top)^{1/2} \tilde{q}_t - (S_0 S_0^\top)^{1/2} \Omega c_t \right\|_2^2, \qquad (S_0 S_0^\top)^{1/2} \tilde{q}_t \in \mathbb{R}^K. \tag{6}$$

Thus, minimizing output reconstruction error is equivalent to minimizing the state-weighted query approximation error. This means that larger query errors can be tolerated along directions that the state amplifies less. Our key observation is that the state-weighted query $(S_0 S_0^\top)^{1/2} \tilde{q}_t$ can be well approximated within the low-dimensional subspace spanned by $(S_0 S_0^\top)^{1/2} \Omega$, using state- and query-dependent coefficients c_t. Section 4 details how we select the fixed query basis Ω and compute coefficients c_t to minimize output reconstruction error. Fig. 2 supports this observation using the retained output fraction $\rho_G := 1 - \mathbb{E}_{\mathcal{D}}[\mathcal{E}_t^{\text{state}}] / \mathbb{E}_{\mathcal{D}}[\|o_t^{\text{state}}\|_2^2]$, computed per head and summarized by the mean across heads. Using just $G = 4$ basis vectors in the $K = 128$-dimensional query space retains 88.4–97.5% of output energy on held-out samples, averaged across heads for each of the three model families.

Key Idea: Precomputing Output Vectors as a Sketch. This observation allows us to compute o_t^{state} accurately while avoiding repeated reads of the full state for each new query. With the query approximation $\tilde{q}_t \approx \Omega c_t$, we can express the output as

$$o_t^{\text{state}} = S_0^\top \tilde{q}_t \approx S_0^\top \Omega c_t = (S_0^\top \Omega) c_t = U c_t, \qquad U := S_0^\top \Omega \in \mathbb{R}^{V \times G}. \tag{7}$$

The approximated state-read output is then $\hat{o}_t^{\text{state}} = U c_t$. With fixed Ω, a single read of the full state S_0 at the flush step suffices to precompute the outputs for all G query basis vectors in Ω before the

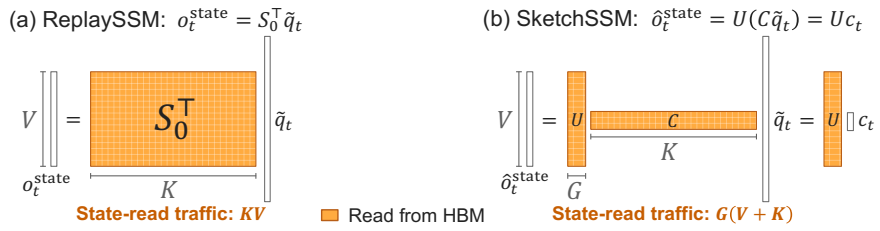

Fig. 3: (a) ReplaySSM reads full state to compute o_t^{state}, incurring KV elements of state-read traffic. (b) SketchSSM reconstructs $\hat{o}_t^{\text{state}}$ from rank-G factors, reducing traffic to $G(K+V)$ elements.

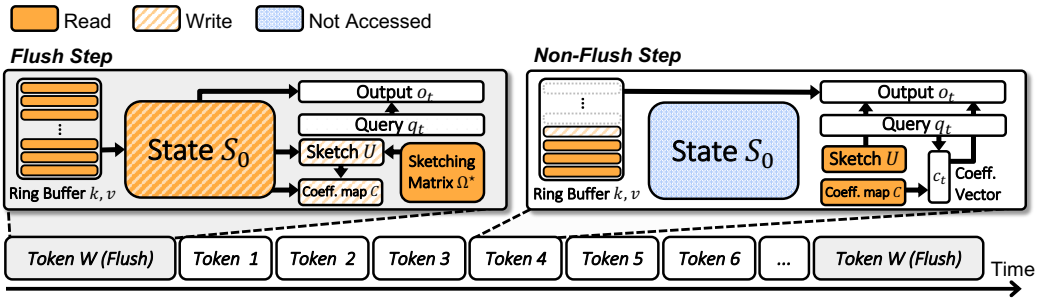

Fig. 4: SketchSSM over a window. At a flush step, SketchSSM updates full state S_0 and refreshes sketch U. At non-flush steps, it multiplies U by query-dependent coefficients c_t to reconstruct output. Note that S_0 is not accessed between state updates. Algorithm 1 summarizes the procedure.

actual queries arrive. Each column $u_g = S_0^\top \omega_g \in \mathbb{R}^V$ stores the output for one query basis vector. We call this matrix of G precomputed output vectors the *sketch*. Using sketch U, each decode step between state updates no longer needs to read the full state. Instead, it reconstructs the output as a linear combination of output vectors in sketch using the query-dependent coefficients $c_t = C\tilde{q}_t$, where $C \in \mathbb{R}^{G\times K}$ is a coefficient map precomputed from the current state S_0 per window. Thus, as illustrated in Fig. 3, repeated reads of the full state S_0 are replaced by reads of the much smaller sketch U and coefficient map C, reducing state-read traffic. This design decouples read approximation from updates: the sketch serves only reads, while the full state is updated exactly from the buffered inputs. Section 4 next describes how we compute Ω, U, and c_t.

4 SketchSSM

SketchSSM finds an accurate yet inference-efficient solution to Equation 6. To avoid the computational overhead of repeatedly computing the query basis, we calibrate the sketching matrix Ω offline and fix it throughout inference. In contrast, we refresh the sketch U at each window boundary to reflect the current state, precomputing the outputs for these fixed query basis vectors. Finally, at each decode step between state updates, we compute query-dependent coefficients c_t that linearly combine the sketch vectors to accurately approximate the current query's output. We first detail these three computations, then describe how we allocate sketch ranks across heads for better accuracy. Algorithm 1 (Appendix B) summarizes the complete procedure.

4.1 Constructing the Sketch

Offline: Sketching Matrix (Ω). The columns of $(S_0 S_0^\top)^{1/2}\Omega$ span the low-dimensional subspace used to approximate the state-weighted query $(S_0 S_0^\top)^{1/2}\tilde{q}_t$. The optimal Ω therefore depends on S_0, which varies across requests and decode steps. However, optimizing Ω for each S_0 is computationally expensive. We therefore fix Ω throughout inference and calibrate it offline for each state head using

the average state geometry over the calibration distribution $\mathcal{D}$, $E_0 := \mathbb{E}_{\mathcal{D}}[S_0 S_0^\top]$. This means that, at runtime, we still approximate $(S_0 S_0^\top)^{1/2}\tilde{q}_t$ with $(S_0 S_0^\top)^{1/2}\Omega c_t$, but use a shared Ω selected from calibration samples using E_0. The corresponding offline loss is

$$\mathcal{L}_G(\Omega) := \mathbb{E}_{\mathcal{D}}\left[\min_{c_t}\left\|E_0^{1/2}\tilde{q}_t - E_0^{1/2}\Omega c_t\right\|_2^2\right]. \tag{8}$$

It is solved by uncentered PCA of the state-weighted query $z_t := E_0^{1/2}\tilde{q}_t$. With $C_z := \mathbb{E}_{\mathcal{D}}[z_t z_t^\top]$, the solution is $\Omega^\star = E_0^{-1/2}P_G \in \mathbb{R}^{K\times G}$, where P_G contains the leading G eigenvectors of C_z. Appendix A.1 gives the full derivation, and Appendix F.1 describes the detailed calibration procedure.

Per-window: Sketch (U). At the flush step of each window, the buffered updates are applied to the full state, producing S_0 for the next window. As described in Section 3.3, we precompute the output vectors that S_0 produces for the query basis vectors in Ω, forming the sketch $U = S_0^\top\Omega$. Constructing the sketch requires a single full-state read, so we fuse this computation into the flush kernel that updates S_0. The kernel constructs U before writing the updated state back to memory, eliminating additional full-state traffic.

Per-step: Coefficient Vector (c_t). At each decode step between flushes, SketchSSM combines the precomputed output vectors in U to approximate o_t^{state} as $\hat{o}_t^{\text{state}} = Uc_t$. We choose the coefficients to minimize output error for the current state and query:

$$c_t^\star := \arg\min_{c\in\mathbb{R}^G}\left\|S_0^\top\tilde{q}_t - Uc\right\|_2^2 = C\tilde{q}_t, \qquad C := \left(U^\top U\right)^\dagger U^\top S_0^\top. \tag{9}$$

Appendix A.2.1 gives the derivation. The coefficient map C depends on the current state, which remains fixed within a window. We therefore compute C once at each flush and obtain $c_t = C\tilde{q}_t$ for each subsequent query.

A cheaper alternative is to precompute a fixed map using the average state metric E_0:

$$c_t^{\text{offline}} := \arg\min_{c\in\mathbb{R}^G}\left\|E_0^{1/2}\tilde{q}_t - E_0^{1/2}\Omega c\right\|_2^2 = C_{\text{offline}}\tilde{q}_t, \qquad C_{\text{offline}} := (\Omega^\top E_0\Omega)^\dagger\Omega^\top E_0. \tag{10}$$

This avoids recomputing the map at each flush, but does not adapt to the current state and substantially reduces accuracy in our ablation (Table 1). We therefore retain the state-dependent map and reduce its computation cost using a low-rank-plus-diagonal approximation detailed in Appendix A.2.2. Table 2 shows that this approximation accelerates the complete flush step by $4.41\times$ relative to exact coefficient-map computation on Nemotron 3 Super at $\bar{G} = 5$ and batch size 256. With this optimization, Fig. 7(b) shows that sketch and coefficient-map construction adds only 11.5–17.0% to flush-step latency relative to the same CUDA flush implementation without these operations.

4.2 Allocating Sketch Ranks Across Layers and Heads

SketchSSM uses offline calibration to allocate a sketch-rank budget, which determines sketch size, non-uniformly across state heads. It assigns larger ranks to heads whose low-rank sketches leave larger o_t^{state} reconstruction errors and whose o_t^{state} has a stronger influence on the loss. For each head and candidate rank G, let $\delta^o_{t,l,h}(G) \in \mathbb{R}^V$ be the output reconstruction error from using a rank-G sketch instead of the full state, and let $g_{t,l,h} := \nabla_{o^{\text{state}}_{t,l,h}}\ell \in \mathbb{R}^V$ be the loss gradient, which measures how sensitive the model loss is to changes in $o^{\text{state}}_{t,l,h}$. We define

$$J_{l,h}(G) := \mathbb{E}_t\left[\left(g_{t,l,h}^\top\delta^o_{t,l,h}(G)\right)^2\right]. \tag{11}$$

Our goal is to minimize the sum of these scores across state heads under a fixed mean sketch-rank budget, $\bar{G}$. We solve this problem as a multiple-choice knapsack problem using a Lagrangian relaxation, as detailed in Appendix E. An offline calibration run computes all per-head candidate scores, which we reuse to generate allocations for different mean sketch-rank budgets $\bar{G}$. Table 1 shows our allocation improves accuracy over uniform allocation by 16.17 percentage points on average.

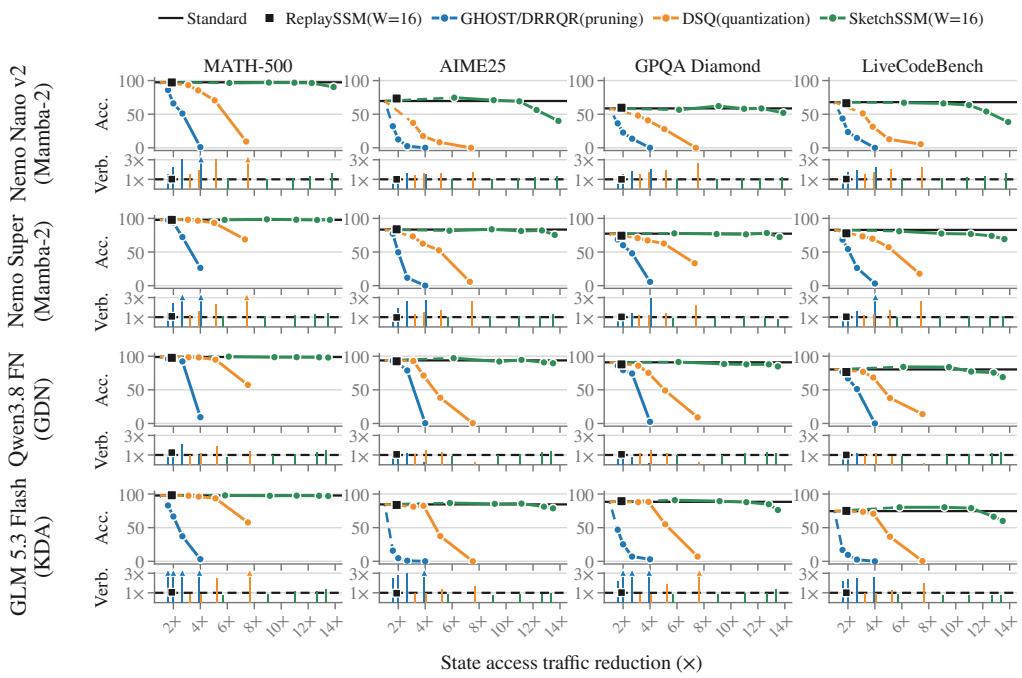

Fig. 5: Accuracy and verbosity versus state access traffic reduction relative to Standard as detailed in Appendix D. ReplaySSM and SketchSSM use $W = 16$. DSQ uses $\{10, 8, 6, 4\}$-bit states; GHOST (Nano/Super/GLM) and DRRQR (Qwen) prune $\{37.5, 50, 62.5, 75\}\%$ of the state. SketchSSM uses $\bar{G} \in \{21, 10, 6, 4, 2\}$, $\{20, 9, 5, 3, 2\}$, $\{26, 11, 7, 4, 3\}$, and $\{28, 12, 7, 4, 3\}$ for Nano, Super, Qwen, and GLM, respectively. SketchSSM largely preserves average accuracy across the four benchmarks at reductions up to approximately $10\times$, whereas pruning and quantization degrade accuracy at smaller reductions. Verbosity is the mean generation length normalized to Standard. Raw accuracy results are provided in Appendix F.2.

5 Evaluation

Evaluation Setup. We evaluate four models spanning three linear-attention methods: Nemotron Nano 9B v2 and Nemotron 3 Super (Mamba-2), Qwen3.8 Flash-Next (GDN), and GLM 5.3 Flash (KDA), all with NVFP4 weights. We evaluate accuracy on MATH-500 (Lightman et al., 2023), AIME25 Zhang & Math-AI (2025), GPQA Diamond (Rein et al., 2024), and LiveCodeBench Jain et al. (2024) against state quantization, state pruning, and full-state baselines with Standard execution and ReplaySSM (Liou & Dao, 2026). ReplaySSM represents buffered-update baselines such as KVBuffer (Zou & Zhong, 2026). For state quantization, we use Decoupled Scale Quantization(DSQ) from Q-Mamba (Tianqi et al., 2025), which dynamically computes separate scales along the two dimensions of the state. For state pruning, we use GHOST (Menezes & Kyrillidis, 2026) for Mamba-2- and KDA-based models and DRRQR (Nazari & Rusch, 2026) for GDN-based models. We integrate our optimized SketchSSM kernels into vLLM Kwon et al. (2023) and conduct all evaluations on vLLM. For speed evaluation, we measure linear attention kernel latency on Mamba-2, GDN, and KDA, and measure end-to-end decode throughput on Nemotron 3 Super using a single NVIDIA B300 GPU. Detailed evaluation settings are provided in Appendix F.1.

5.1 Accuracy

Accuracy Comparison with Baselines. As shown in Fig. 5, SketchSSM largely preserves average accuracy across four decode benchmarks for all four models at state-access traffic reductions of up to approximately $10\times$ relative to Standard. Larger reductions incur accuracy losses, illustrating

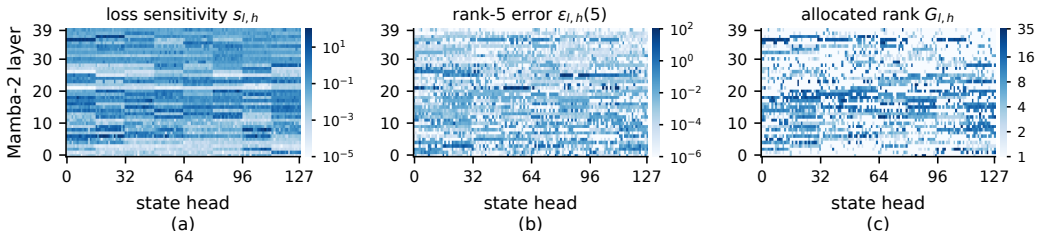

Fig. 6: Illustrative head-level rank allocation on Nemotron 3 Super 120B-A12B at $\bar{G} = 5$ (11.12× state-access traffic reduction). (a) Loss sensitivity. (b) Rank-five state-read error. (c) Allocated rank (range 1–35; 12 heads retain the dense state, shown at the maximum color intensity).

Table 1: Ablation of internal design choices on Nemotron Nano v2 9B, including sketch-basis calibration, state-dependent coefficient, sketch rank allocation, at $\bar{G} = 6$

Ablation	Method	MATH-500	AIME25	GPQA-D	LCB	Average
Full method	SketchSSM	96.80	69.17	58.08	63.49	71.88
Sketch basis Ω	Random orthogonal	81.80 (−15.00)	17.50 (−51.67)	40.91 (−17.17)	38.41 (−25.08)	44.66 (−27.22)
Coefficient vector c_t	Offline Coefficient Map	89.40 (−7.40)	35.42 (−33.75)	48.99 (−9.09)	17.14 (−46.35)	47.74 (−24.14)
Sketch-rank allocation	Uniform allocation	90.60 (−6.20)	39.58 (−29.59)	52.02 (−6.06)	40.63 (−22.86)	55.71 (−16.17)
	Error-only score	96.20 (−0.60)	57.50 (−11.67)	57.07 (−1.01)	49.21 (−14.28)	65.00 (−6.88)
	Gradient-only score	90.60 (−6.20)	36.25 (−32.92)	44.95 (−13.13)	40.63 (−22.86)	53.11 (−18.77)

the tradeoff between accuracy and state-access traffic. In contrast, at approximately 4× reductions, 8-bit state quantization and 75% state pruning reduce average accuracy by 1.9–29.6 and 73.1–87.9 percentage points, respectively, relative to the full-state baseline. Appendix D defines the state-access traffic metric, including both reads and writes, and Appendix F.2 provides detailed results of accuracy evaluation. We additionally evaluate recall on four RULER retrieval tasks. SketchSSM largely preserves retrieval accuracy, whereas state pruning causes larger losses as detailed in Appendix F.3.

Accuracy Ablation. Table 1 shows the effects of basis calibration, state-dependent coefficients, and head-level rank allocation at the same mean sketch rank. First, calibrating Ω for state-weighted queries improves accuracy by 15.0–51.7 percentage points over a random orthogonal basis. Second, state-dependent coefficient maps improve accuracy by 7.4–46.4 points over an offline map. Finally, allocating ranks using both output error and loss sensitivity improves accuracy by 6.1–29.6 points over uniform allocation and yields higher average accuracy than using either criterion alone. Fig. 6 illustrates their variation across heads, motivating head-level allocation. Appendix G shows that preserving full-state updates while approximating reads substantially improves accuracy by preventing propagation of state compression errors, with sketches further improving the accuracy–traffic tradeoff over other read representations.

5.2 Speedups

Kernel Speedups. We evaluate Mamba-2, GDN, and KDA kernels on one NVIDIA B300 with $W = 16$ and batch sizes 128, 256, and 512 (Fig. 7). SketchSSM reduces non-flush latency by replacing full-state reads with sketch reads. Its fused flush kernel constructs the sketch and coefficient map while applying state updates, limiting the additional construction overhead. On Nemotron Super at $\bar{G} = 5$, this construction adds 11.5–17.0% latency over the same flush kernel without it. Including all steps in the window, total linear attention speedups over Standard reach 7.78×, 5.22×, and 5.20× for Nemotron Super, Qwen3.8 Flash-Next, and GLM 5.3 Flash, respectively.

End-to-End Decode Speedups. With $\bar{G} = 5$, SketchSSM improves maximum decode throughput by 2.64× at 2K and 2.41× at 8K over the Standard full-state baseline. Fig. 8(a) shows throughput across batch sizes, up to each method's capacity limit. These gains come from reducing recurrent computation, which accounts for much of the baseline's decode time. While ReplaySSM improves maximum throughput by 1.59× at 2K and 1.55× at 8K by amortizing state updates, SketchSSM

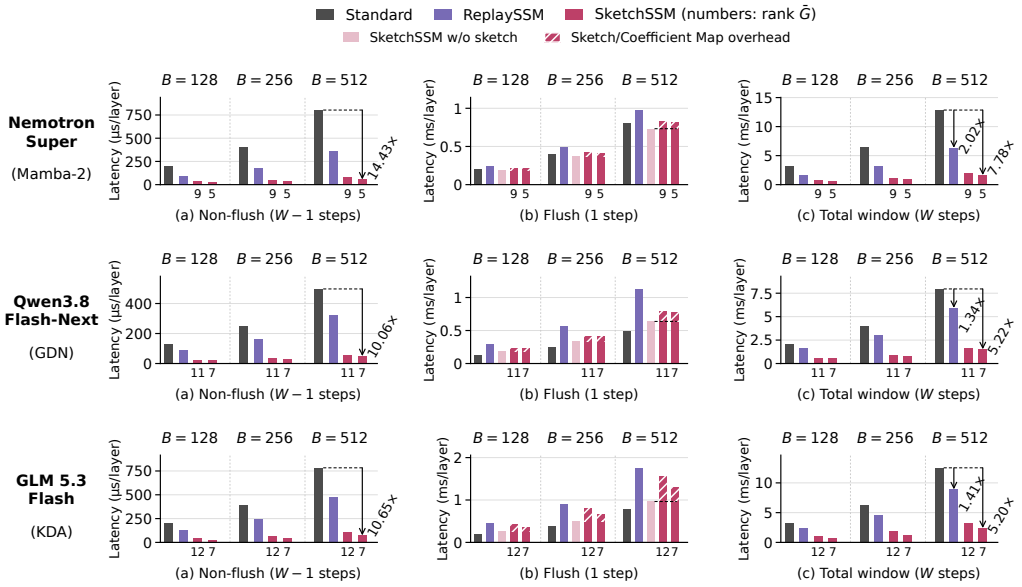

Fig. 7: Linear-attention latency on one NVIDIA B300 with $W = 16$ and batch sizes 128, 256, and 512. Rows show Nemotron 3 Super($\bar{G} \in \{5, 9\}$), Qwen3.8 Flash-Next ($\bar{G} \in \{7, 11\}$), and GLM 5.3 Flash ($\bar{G} \in \{7, 12\}$). (a) Non-flush-step latency, incurred at each of the $W - 1$ steps between flush steps. (b) Flush-step latency, incurred once per window. Hatching indicates sketch and coefficient-map construction overhead at flush step. (c) Total linear-attention latency over W steps.

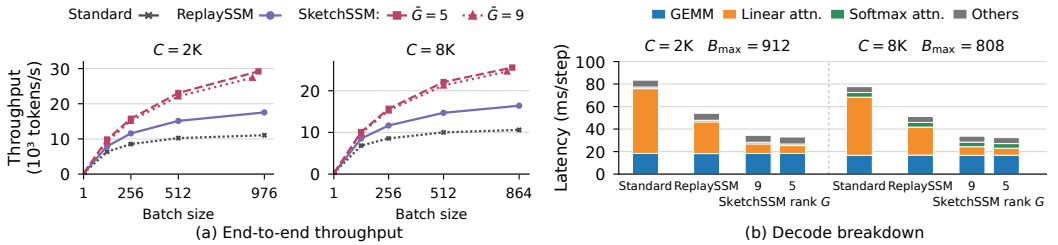

Fig. 8: Decoding performance for Nemotron 3 Super on one NVIDIA B300, comparing Standard, ReplaySSM, and SketchSSM at $\bar{G} \in \{5, 9\}$. (a) decode throughput at 2K and 8K up to each method's maximum batch, averaged over two 128-step runs using six steady windows. (b) Decode latency breakdown at common maximum batches of 912 at $C = 2K$ and 808 at $C = 8K$.

further improves throughput by 1.67× and 1.56×, respectively, by reducing state-read traffic with a compact sketch. With $\bar{G} = 5$, linear attention's share of decode time falls from 69.0% to 21.3% at 2K and from 66.5% to 19.4% at 8K (Fig. 8(b)), so linear attention no longer dominates decode time.

6 Conclusion

We present SketchSSM, which preserves exact full-state updates while using compact sketches for output reconstruction. We construct the sketch at runtime from the current state using an offline-calibrated sketching matrix. Sketch ranks are also allocated offline based on each state head's output error and its impact on model loss. Across four hybrid-attention models, SketchSSM reduces state-access traffic by approximately $10\times$ while largely preserving accuracy and recall. On one NVIDIA B300, it achieves 5.20–$7.78\times$ linear-attention speedups across Mamba-2, GDN, and KDA, yielding up to $2.64\times$ higher maximum throughput over the full-state baseline.

AI USE STATEMENT

We used generative AI tools to assist with identifying a broad range of prior work relevant to our research, drafting portions of the manuscript, and improving its wording, clarity, and organization. The authors take full responsibility for the final manuscript, including all scientific claims, experimental results, and AI-assisted content.

ETHICS STATEMENT

We acknowledge the ICLR Code of Ethics. This work focuses on improving the inference efficiency of existing language models using established benchmarks. We do not identify ethical concerns specific to the proposed method beyond those associated with the underlying models and their applications.

REPRODUCIBILITY STATEMENT

Section 4 describes our method, and Appendix B summarizes the algorithm. Appendices A.1 and A.2.1 provide the mathematical derivations. Calibration and evaluation settings are described in Appendix F.1, and the state-access traffic metric is defined in Appendix D. Detailed accuracy and recall results are provided in Appendices F.2 and F.3.

REFERENCES

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1763–1840. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/arora24a.html.

Hung-Yueh Chiang, Chi-Chih Chang, Natalia Frumkin, Kai-Chiang Wu, Mohamed S. Abdelfattah, and Diana Marculescu. Quamba2: A robust and scalable post-training quantization framework for selective state space models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10411–10427. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chiang25a.html.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

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.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. doi: 10.1137/090771806. URL https://doi.org/10.1137/090771806.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1625abb8e458a79765c62009235e9d5b-Paper.pdf.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

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

Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pp. 581–588, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321747. doi: 10.1145/2487575.2487623. URL https://doi.org/10.1145/2487575.2487623.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Ze-Wei Liou and Tri Dao. ReplaySSM: Cache SSM inputs, not state. Dao AI Lab Blog, June 2026. URL https://dao-lab.ai/blog/2026/replayssm/. Accessed: 2026-09-07.

Michael Menezes and Anastasios Kyrillidis. GHOST: Grouped hidden-state output-aware selection and truncation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

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

NVIDIA, :, Aakshita Chandiramani, Aaron Blakeman, Abdullahi Olaoye, Abhibha Gupta, Abhilash Somasamudramath, Abhinav Khattar, Adeola Adesoba, Adi Renduchintala, Adil Asif, Aditya Agrawal, Aditya Vavre, Ahmad Kiswani, Aishwarya Padmakumar, Ajay Hotchandani, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Gronskiy, Alex Kondratenko, Alex Neefus, Alex Steiner, Alex Yang, Alexander Bukharin, Alexander Young, Ali Hatamizadeh, Ali Taghibakhshi, Alina Galiautdinova, Alisa Liu, Alok Kumar, Ameya Sunil Mahabaleshwarkar, Amir Klein, Amit Zuker, Amnon Geifman, Anahita Bhiwandiwalla, Ananth Subramaniam, Andrew Tao, Anjaney Shrivastava, Anjulie Agrusa, Ankur Srivastava, Ankur Verma, Ann Guan, Anna Shors, Annamalai Chockalingam, Anubhav Mandarwal, Aparnaa Ramani,

Arham Mehta, Arti Jain, Arun Venkatesan, Asha Anoosheh, Ashwath Aithal, Ashwin Poojary, Asif Ahamed, Asit Mishra, Asli Sabanci Demiroz, Asma Kuriparambil Thekkumpate, Atefeh Sohrabizadeh, Avinash Kaur, Ayush Dattagupta, Barath Subramaniam Anandan, Bardiya Sadeghi, Barnaby Simkin, Ben Lanir, Benedikt Schifferer, Benjamin Chislett, Besmira Nushi, Bilal Kartal, Bill Thiede, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Branislav Kisacanin, Brian Yu, Bryan Catanzaro, Buvaneswari Mani, Carlo del Mundo, Chankyu Lee, Chanran Kim, Chantal Hwang, Chao Ni, Charles Wang, Charlie Truong, Cheng-Ping Hsieh, Chenhan Yu, Chenjie Luo, Cherie Wang, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Chris Holguin, Chris Wing, Christian Munley, Christopher Parisien, Chuck Desai, Chunyang Sheng, Collin Neale, Cyril Meurillon, Dakshi Kumar, Dan Gil, Dan Su, Dane Corneil, Daniel Afrimi, Daniel Burkhardt Eliuth Triana, Daniel Egert, Daniel Fatade, Daniel Lo, Daniel Rohrer, Daniel Serebrenik, Daniil Sorokin, Daria Gitman, Daria Levy, Darko Stosic, David Edelsohn, David Messina, David Mosallanezhad, David Tamok, Deena Donia, Deepak Narayanan, Devin O'Kelly, Dheeraj Peri, Dhruv Nathawani, Di Wu, Dima Rekesh, Dina Yared, Divyanshu Kakwani, Dmitry Konyagin Brandon Tuttle, Dong Ahn, Dongfu Jiang, Dorrin Poorkay, Douglas O'Flaherty, Duncan Riach, Dusan Stosic, Dustin Van Stee, Edgar Minasyan, Edward Lin, Eileen Peters Long, Elad Segal, Elena Lantz, Elena Lewis, Ellie Evans, Elliott Ning, Eric Chung, Eric Harper, Eric Pham-Hung, Eric W. Tramel, Erick Galinkin, Erik Pounds, Esti Etrog, Evan Briones, Evan Wu, Evelina Bakhturina, Evgeny Tsykunov, Ewa Dobrowolska, Farshad Saberi Movahed, Farzan Memarian, Fay Wang, Fei Jia, Felipe Soares, Felipe Vieira Frujeri, Feng Chen, Fengguang Lin, Ferenc Galko, Fortuna Zhang, Frankie Siino, Frida Hou, Gantavya Bhatt, Gargi Prasad, Geethapriya Venkataramani, Geetika Gupta, George Armstrong, Gerald Shen, Giulio Borghesi, Gordana Neskovic, Gorkem Batmaz, Grace Lam, Grace Wu, Greg Pauloski, Greyson Davis, Grigor Nalbandyan, Guoming Zhang, Guy Farber, Guyue Huang, Haifeng Qian, Haran Kumar Shiv Kumar, Harry Kim, Harsh Sharma, Hayate Iso, Hayley Ross, Herbert Hum, Herman Sahota, Hexin Wang, Himanshu Soni, Hiren Upadhyay, Huy Nguyen, Iain Cunningham, Ido Galil, Ido Shahaf, Igino Padovani, Igor Gitman, Igor Shovkun, Ikroop Dhillon, Ilya Loshchilov, Ingrid Kelly, Itamar Schen, Itay Levy, Ivan Moshkov, Izik Golan, Izzy Putterman, Jain Tu, Jan Baczek, Jan Kautz, Jane Polak Scowcroft, Janica Rosenberg, Jared Casper, Jarrod Pflum, Jason Grant, Jason Sewall, Jatin Mitra, Jeffrey Glick, Jenny Chen, Jesse Oliver, Jiacheng Xu, Jiafan Zhu, Jialin Song, Jian Zhang, Jiaqi Zeng, Jie Lou, Jill Milton, Jim Chow, Jimmy Zhang, Jinhang Choi, Jining Huang, Jocelyn Huang, Joel Caruso, Joey Conway, Joey Guman, Johan Jatko, John Kamalu, Johnny Greco, Jonathan Cohen, Jonathan Raiman, Joseph Jennings, Joyjit Daw, Juan Yu, Julio Tapia, Junkeun Yi, Jupinder Parmar, Jyothi Achar, Kari Briski, Kartik Mattoo, Katherine Cheung, Katherine Luna, Keith Wyss, Kevin Shih, Kezhi Kong, Khanh Nguyen, Khushi Bhardwaj, Kirill Buryak, Kirthi Shankar Sivamani, Konstantinos Krommydas, Kris Murphy, Krishna C. Puvvada, Krzysztof Pawelec, Kumar Anik, Laikh Tewari, Laya Sleiman, Leo Du, Leon Derczynski, Li Ding, Lilach Ilan, Lingjie Wu, Lizzie Wei, Luis Vega, Lun Su, Maarten Van Segbroeck, Maer Rodrigues de Melo, Magaret Zhang, Mahan Fathi, Makesh Narsimhan Sreedhar, Makesh Sreedhar, Makesh Tarun Chandran, Manuel Reyes Gomez, Maor Ashkenazi, Marc Cuevas, Marc Romeijn, Margaret Zhang, Mark Cai, Mark Gabel, Markus Kliegl, Martyna Patelka, Maryam Moosaei, Matthew Varacalli, Matvei Novikov, Mauricio Ferrato, Mehrzad Samadi, Melissa Corpuz, Meng Xin, Mengdi Wang, Mengru Wang, Meredith Price, Micah Schaffer, Michael Andersch, Michael Boone, Michael Evans, Michael Z Wang, Miguel Martinez, Mikail Khona, Mike Chrzanowski, Mike Hollinger, Mingyuan Ma, Minseok Lee, Mohammad Dabbah, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Nader Khalil, Najeeb Nabwani, Nancy Agarwal, Nanthini Balasubramaniam, Narimane Hennouni, Narsi Kodukula, Natalie Hereth, Nathaniel Pinckney, Nave Assaf, Negar Habibi, Nestor Qin, Neta Zmora, Netanel Haber, Nick Reamaroon, Nickson Quak, Nidhi Bhatia, Nikhil Jukar, Nikki Pope, Nikolai Ludwig, Nima Tajbakhsh, Nir Ailon, Nirmal Juluru, Nirmalya De, Nowel Pitt, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Oluwatobi Olabiyi, Omer Ullman Argov, Omri Almog, Omri Puny, Oren Tropp, Otavio Padovani, Ouye Xie, Parth Chadha, Pasha Shamis, Paul Gibbons, Pavlo Molchanov, Peter Belcak, Peter Jin, Pinky Xu, Piotr Januszewski, Pooya Jannaty, Prachi Shevate, Pradeep Thalasta, Pranav Prashant Thombre, Prasoon Varshney, Prerana Gambhir, Pritam Gundecha, Przemek Tredak, Qing Miao, Qiyu Wan, Quan Tran Minh, Rabeeh Karimi Mahabadi, Rachel Oberman, Rachit Garg, Rahul Kandu, Raina Zhong, Ran El-Yaniv, Ran Zilberstein, Rasoul Shafipour, Renee Yao, Renjie Pi, Richard Mazzarese, Richard Wang, Rick Izzo, Ridhima Singla, Rima Shahbazyan, Rishabh Garg, Ritika Borkar, Ritu Gala, Riyad Islam, Robert Clark, Robert Hesse, Roger Waleffe, Rohit Varma Kalidindi, Rohit Watve, Roi Koren, Ron Fan, Ruchika Kharwar, Ruisi Cai, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Ryan Timbrook, Ryota Egashira, Sadegh Mahdavi, Sagar Singh Ashutosh

Joshi, Sahil Modi, Samuel Kriman, Sandeep Pombra, Sanjay Kariyappa, Sanjeev Satheesh, Santiago Pombo, Saori Kaji, Satish Pasumarthi, Saurav Mishra, Saurav Muralidharan, Scott Hara, Sean Narenthiran, Sebastian Rogawski, Seonjin Na, Seonmyeong Bak, Sepehr Sameni, Seth Poulos, Shahar Mor, Shantanu Acharya, Shaona Ghosh Adam Lord, Sharath Turuvekere Sreenivas, Shaun Kotek, Shaya Gharghabi, Shelby Thomas, Sheng-Chieh Lin, Shibani Likhite, Shiqing Fan, Shiyang Chen, Shreya Gopal, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuo Zhang, Shuoyang Ding, Shyam Renjith, Shyamala Prayaga, Siddhartha Jain, Simeng Sun, Sirisha Rella, Sirshak Das, Smita Ithape, Sneha Harishchandra S, Somshubra Majumdar, Soumye Singhal, Sri Harsha Singudasu, Sriharsha Niverty, Stas Sergienko, Stefana Gloginic, Stefania Alborghetti, Stephen Ge, Stephen McCullough, Sugam Dipak Devare, Suguna Varshini Velury, Sukrit Rao, Sumeet Kumar Barua, Sunny Gai, Suseella Panguluri, Sushil Koundinyan, Swathi Patnam, Sweta Priyadarshi, Swetha Bhendigeri, Syeda Nahida Akter, Sylendran Arunagiri, Tailling Yuan, Talor Abramovich, Tan Bui, Tan Yu, Terry Kong, Thanh Do, Thomas Gburek, Thorgane Marques, Tiffany Moore, Tijmen Blankevoort, Tim Moon, Timothy Ma, Tiyasa Mitra, Tomasz Grzegorzek, Tomer Asida, Tomer Bar Natan, Tomer Keren, Tomer Ronen, Traian Rebedea, Trenton Starkey, Tugrul Konuk, Twinkle Vashishth, Tyler Condensa, Udi Karpas, Ushnish De, Vahid Noorozi, Vahid Noroozi, Vanshil Atul Shah, Veena Vaidyanathan, Venkat Srinivasan, Venmugil Elango, Victor Cui, Vijay Korthikanti, Vikas Mehta, Virginia Adams, Virginia Wu, Vitaly Kurin, Vitaly Lavrukhin, Vladimir Anisimov, Wan Seo, Wanli Jiang, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wei-Ming Chen, Wendy Quan, Wenliang Dai, Wenwen Gao, Will Jennings, William Zhang, Xiaowei Ren, Xiaowen Xin, Xin Li, Yang Yu, Yangyi Chen, Yaniv Galron, Yashaswi Karnati, Yejin Choi, Yev Meyer, Yi-Fu Wu, Yian Zhang, Ying Lin, Yonatan Geifman, Yonggan Fu, Yoshi Suhara, Youngeun Kwon, Yuan Zhang, Yuki Huang, Zach Moshe, Zhilin Wang, Zhiyu Cheng, Zhongbo Zhu, Zhuolin Yang, Zihan Liu, Zijia Chen, Zijie Yan, and Zuhair Ahmed. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026. URL https://arxiv.org/abs/2604.12374.

Yuqi Pan, Yongqi An, Zheng Li, Yuhong Chou, Rui-Jie Zhu, Xiaohui Wang, Mingxuan Wang, Jinqiao Wang, and Guoqi Li. Scaling linear attention capacity with sparse state expansion. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=R6DrJ4tnGV.

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.

Qwen Team. On the design of Qwen3.8-Next architecture: Evaluation, efficiency, and training stability. Technical report, Alibaba Group, August 2026a.

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026b. URL https://qwen.ai/blog?id=qwen3.8-flash-next.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.899. URL https://aclanthology.org/2024.findings-emnlp.899/.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proc. Int. Conf. on Machine Learning (ICML)*, Virtual only, July 2021.

Chen Tianqi, Yuanteng Chen, Peisong Wang, Weixiang Xu, Zeyu Zhu, and Jian Cheng. Q-mamba: Towards more efficient mamba models via post-training quantization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10594–10610, Vienna, Austria, July 2025. Association

for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.551. URL https://aclanthology.org/2025.findings-acl.551/.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=LNYIUouhdt.

David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 10 2014. ISSN 1551-305X. doi: 10.1561/0400000060. URL https://doi.org/10.1561/0400000060.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/yang24ab.html.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 115491–115522. Curran Associates, Inc., 2024b. doi: 10.52202/079017-3668. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d13a3eae72366e61dfdc7eea82eeb685-Paper-Conference.pdf.

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.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models, 2025. URL https://arxiv.org/abs/2312.05821.

Tao Zhang, Jianchao Tan, Pingwei Sun, Yanqi Yu, Zixu Jiang, Yuchen Xie, Xunliang Cai, and Ziqian Zeng. Damp: Decay-aware mixed-precision recurrent-state quantization, 2026. URL https://arxiv.org/abs/2608.27513.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, Y. T. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025.

Longwei Zou and Lin Zhong. Kvbuffer: Io-aware serving for linear attention, 2026. URL https://arxiv.org/abs/2605.19049.

A Derivation of SketchSSM

SketchSSM solves two related approximation problems. First, it calibrates the shared sketching matrix Ω from many state–query samples. Second, for the current state and query, it computes the coefficient vector c_t used to combine the columns of the state sketch. We derive these two steps separately and then explain their efficient runtime implementation.

A.1 Derivation of the Optimal Offline Sketching Matrix (Ω)

We derive the PCA solution used in Equation 8. Recall that $E_0 \in \mathbb{R}^{K\times K}$ is the calibration-averaged state geometry, $\tilde{q}_t \in \mathbb{R}^K$, and $\Omega \in \mathbb{R}^{K\times G}$. The goal is to choose the G-dimensional column space of Ω that minimizes the offline reconstruction loss in Equation 8

Step 1: Convert the weighted error to an ordinary Euclidean error. Define

$$z_t := E_0^{1/2}\tilde{q}_t \in \mathbb{R}^K, \qquad B := E_0^{1/2}\Omega \in \mathbb{R}^{K\times G}. \tag{12}$$

These are only changes of variables. In particular,

$$\left\| E_0^{1/2}(\tilde{q}_t - \Omega c_t) \right\|_2^2 = \|z_t - Bc_t\|_2^2. \tag{13}$$

Thus, for a fixed B, the inner problem asks for the linear combination of its G columns that is closest to z_t.

Step 2: Solve for the best coefficient at one sample. Expanding the squared distance gives

$$\begin{aligned}\|z_t - Bc\|_2^2 &= (z_t - Bc)^\top (z_t - Bc) \\ &= z_t^\top z_t - 2c^\top B^\top z_t + c^\top B^\top Bc.\end{aligned} \tag{14}$$

Differentiating with respect to c and setting the derivative to zero produces the normal equation

$$B^\top Bc = B^\top z_t. \tag{15}$$

If B has full column rank, $B^\top B$ is invertible, and hence

$$c_{t,\mathrm{off}}^\star(B) = (B^\top B)^{-1}B^\top z_t. \tag{16}$$

The reconstructed vector is $Bc_{t,\mathrm{off}}^\star = \Pi_B z_t$, where

$$\Pi_B := B(B^\top B)^{-1}B^\top \tag{17}$$

is the orthogonal projector onto the column space of B. Therefore, the minimum sample error is $\|(I - \Pi_B)z_t\|_2^2$, the energy left outside that column space. If B is rank deficient, the same statements hold with the Moore–Penrose pseudoinverse.

Step 3: Average the projection error over calibration samples. Let $C_z := \mathbb{E}_{\mathcal{D}}[z_t z_t^\top] \in \mathbb{R}^{K\times K}$. Because Π_B is symmetric and idempotent, $(I - \Pi_B)^\top(I - \Pi_B) = I - \Pi_B$. We also use the scalar identity $x^\top Ax = \mathrm{tr}(Axx^\top)$. Applying these two facts one line at a time gives

$$\begin{aligned}\mathcal{L}_G(\Omega) &= \mathbb{E}_{\mathcal{D}}\left[\|(I - \Pi_B)z_t\|_2^2\right] \\ &= \mathbb{E}_{\mathcal{D}}\left[z_t^\top(I - \Pi_B)z_t\right] \\ &= \mathbb{E}_{\mathcal{D}}\left[\mathrm{tr}((I - \Pi_B)z_t z_t^\top)\right] \\ &= \mathrm{tr}((I - \Pi_B)C_z) \\ &= \mathrm{tr}(C_z) - \mathrm{tr}(\Pi_B C_z).\end{aligned} \tag{18}$$

Step 4: Choose the best G-dimensional subspace. The total energy $\mathrm{tr}(C_z)$ does not depend on B. Minimizing Equation 18 is therefore equivalent to maximizing the captured energy $\mathrm{tr}(\Pi_B C_z)$. Write the eigendecomposition as

$$C_z = P\,\mathrm{Diag}(\lambda_1, \ldots, \lambda_K)P^\top, \qquad \lambda_1 \geq \cdots \geq \lambda_K \geq 0. \tag{19}$$

The PCA variational principle states that a rank-G projector captures the most energy by selecting the eigenvectors with the G largest eigenvalues. To see this directly, let $Q = [q_1, \ldots, q_G]$ be any orthonormal basis for the chosen subspace, so $\Pi_B = QQ^\top$. Then

$$\mathrm{tr}(\Pi_B C_z) = \sum_{g=1}^{G} q_g^\top C_z q_g = \sum_{i=1}^{K} \lambda_i \underbrace{\sum_{g=1}^{G} (p_i^\top q_g)^2}_{w_i}. \tag{20}$$

The weights satisfy $0 \le w_i \le 1$ and $\sum_i w_i = G$. Because the eigenvalues are sorted from largest to smallest, the weighted sum is maximized by assigning unit weight to the first G eigenvectors. Let $P_G = [p_1, \ldots, p_G]$ contain those eigenvectors. We can therefore choose $B^\star = P_G$, for which $\Pi_{B^\star} = P_G P_G^\top$.

Step 5: Map the PCA basis back to Ω. From the definition $B = E_0^{1/2}\Omega$, the choice $B^\star = P_G$ gives

$$\Omega^\star = E_0^{-1/2} P_G, \qquad c_{t,\mathrm{off}}^\star = P_G^\top z_t, \qquad \min_{\Omega} \mathcal{L}_G(\Omega) = \sum_{i=G+1}^{K} \lambda_i. \tag{21}$$

The final equality has a direct interpretation: PCA retains the first G eigen-directions, so the minimum residual is the energy in the remaining $K - G$ directions. No centering is performed, which is why the main text calls this procedure uncentered PCA. The coefficient $c_{t,\mathrm{off}}^\star$ is used only inside this offline calibration objective; inference uses the current-state coefficient derived in Appendix A.2.1. If E_0 is singular, $E_0^{-1/2}$ denotes the Moore–Penrose pseudoinverse on the support of E_0; directions in its null space do not contribute to the output error.

A.2 Derivation of the Coefficient Vector (c_t)

A.2.1 Derivation of the Optimal Coefficient Vector

We next derive the per-step coefficient used with the current state sketch. Within one window, $S_0 \in \mathbb{R}^{K\times V}$ and $U \in \mathbb{R}^{V\times G}$ are fixed, whereas $\tilde{q}_t \in \mathbb{R}^K$ changes at every step. The exact output $o_t^{\mathrm{state}} = S_0^\top \tilde{q}_t \in \mathbb{R}^V$ is approximated by Uc, a linear combination of the G columns of U.

Step 1: Write and expand the least-squares problem. For the current query, we solve

$$c_t^\star := \arg\min_{c\in\mathbb{R}^G} \|o_t^{\mathrm{state}} - Uc\|_2^2,$$
$$\|o_t^{\mathrm{state}} - Uc\|_2^2 = {o_t^{\mathrm{state}}}^\top o_t^{\mathrm{state}} - 2c^\top U^\top o_t^{\mathrm{state}} + c^\top U^\top U c. \tag{22}$$

Step 2: Solve the normal equation. Differentiating the second line with respect to c gives

$$2U^\top(Uc - o_t^{\mathrm{state}}) = 0 \quad \Longrightarrow \quad U^\top U c = U^\top o_t^{\mathrm{state}}. \tag{23}$$

Substituting $o_t^{\mathrm{state}} = S_0^\top \tilde{q}_t$ and choosing the minimum-norm solution yields

$$\begin{aligned} c_t^\star &= (U^\top U)^\dagger U^\top S_0^\top \tilde{q}_t \\ &= C\tilde{q}_t, \qquad C := (U^\top U)^\dagger U^\top S_0^\top \in \mathbb{R}^{G\times K}. \end{aligned} \tag{24}$$

Here, $\dagger$ denotes the Moore–Penrose pseudoinverse. The first factor $(U^\top U)^\dagger \in \mathbb{R}^{G\times G}$ corrects for correlations among the sketch basis vectors, and $U^\top S_0^\top \in \mathbb{R}^{G\times K}$ measures how the full state maps a query into those basis vectors.

With the exact least-squares coefficient map, the reconstructed output is

$$Uc_t^\star = U(U^\top U)^\dagger U^\top S_0^\top \tilde{q}_t = \Pi_U o_t^{\mathrm{state}}, \tag{25}$$

where Π_U is the orthogonal projector onto $\mathrm{range}(U) = \mathrm{range}(S_0^\top \Omega)$. This has the range-projection form of Halko et al. (2011), with the calibrated Ω serving as the test matrix.

Step 3: Reuse the query-independent factors. Both S_0 and U remain fixed within a window, so C is also fixed. SketchSSM constructs C once at the flush step. Each non-flush step then needs only the matrix–vector product $c_t^\star = C\tilde{q}_t$. The remaining question is how to evaluate this product without serially forming the effective query; Appendix C.2 describes that separate execution optimization.

A.2.2 Low-Rank-Plus-Diagonal Approximation for the Coefficient Map

Constructing the exact coefficient map $C = (U^\top U)^\dagger U^\top S_0^\top$ at every flush requires forming and factoring $U^\top U$ and forming $U^\top S_0^\top$, which costs $O(KVG + VG^2 + G^3)$ arithmetic per head. This section derives the approximation used to reduce that flush-step cost.

Step 1: Express the solve in basis-aligned coordinates. For the derivation, we express the coefficient solve in coordinates aligned with the calibrated query subspace. This change of coordinates does not require storing or updating the recurrent state in the rotated basis. We use an orthonormal basis Ω for this subspace and complete it to an orthogonal matrix $R^\top$, where $R^\top R = I_K$. Define

$$S_0' := RS_0 \in \mathbb{R}^{K\times V}, \qquad x_t := R\tilde{q}_t \in \mathbb{R}^K, \qquad J := [e_1 \ \cdots \ e_G] \in \mathbb{R}^{K\times G}. \tag{26}$$

Here, J selects the first G basis-aligned coordinates, so $\Omega = R^\top J$ and $U = S_0^\top R^\top J = {S_0'}^\top J$. This coordinate change preserves the output reconstruction error exactly.

Define the normalized state metric

$$\mu := \|S_0'\|_F^2/K, \qquad H := S_0'{S_0'}^\top/\mu \in \mathbb{R}^{K\times K}, \tag{27}$$

where we set $\mu = 1$ for a zero state. The normalized output error is

$$\|{S_0'}^\top(x_t - Jc)\|_2^2/\mu = (x_t - Jc)^\top H(x_t - Jc). \tag{28}$$

We add ridge regularization $\lambda\|c\|_2^2$ with $\lambda = 0.1$, defined for coefficients in this orthonormal basis. Expanding Equation 28, differentiating the regularized objective with respect to c, and setting the result to zero gives

$$\begin{aligned} 0 &= 2J^\top H(Jc - x_t) + 2\lambda c, \\ (J^\top HJ + \lambda I_G)c &= J^\top Hx_t, \\ c_{t,\lambda} &= (J^\top HJ + \lambda I_G)^{-1}J^\top Hx_t. \end{aligned} \tag{29}$$

The exact regularized solve therefore depends on the dense state metric H. Constructing the required blocks of H at every flush is the expensive operation that we approximate below.

Step 2: Approximate the state metric. Choose a small configurable pivot count P and set $p := \min(G, P)$. Let $Q \in \mathbb{R}^{V\times p}$ be an orthonormal basis for the first p columns of the current sketch $U = {S_0'}^\top J$. The projector $QQ^\top$ keeps the part of the value space spanned by these pivot sketch vectors. Define

$$L := \frac{S_0'Q}{\sqrt{\mu}} \in \mathbb{R}^{K\times p}. \tag{30}$$

Its product is

$$LL^\top = S_0'QQ^\top{S_0'}^\top/\mu, \tag{31}$$

so $LL^\top$ retains the correlations in H that pass through the selected p-dimensional output subspace. A low-rank matrix alone can underestimate the energy of individual state coordinates. We preserve the remaining diagonal energy with

$$d_i := \max\{H_{ii} - \|L_{i,:}\|_2^2, 0\}, \qquad \widehat{H} := LL^\top + \mathrm{Diag}(d). \tag{32}$$

Since $(LL^\top)_{ii} = \|L_{i,:}\|_2^2$, this choice makes $\widehat{H}_{ii} = H_{ii}$ whenever the residual is nonnegative; the maximum with zero protects against numerical roundoff. Thus, $\widehat{H}$ keeps a small set of important cross-coordinate correlations and the per-coordinate energy left outside that set.

Step 3: Substitute the approximation into the solve. Let

$$L_G := J^\top L \in \mathbb{R}^{G\times p}, \qquad D_0 := \mathrm{Diag}(J^\top d), \qquad D_G := D_0 + \lambda I_G. \tag{33}$$

Substituting $\widehat{H}$ for H in the two sides of the normal equation gives

$$J^\top \widehat{H} J + \lambda I_G = L_G L_G^\top + D_G, \tag{34}$$

$$J^\top \widehat{H} x_t = L_G(L^\top x_t) + J^\top \mathrm{Diag}(d) x_t. \tag{35}$$

These equalities follow by distributing $J^\top$ and J over $\widehat{H} = LL^\top + \mathrm{Diag}(d)$. They show explicitly that the approximation is applied to both the Gram matrix and the query-dependent right-hand side.

Step 4: Replace the $G \times G$ inverse with a $p \times p$ inverse. The approximate Gram matrix has the diagonal-plus-low-rank form $D_G + L_G L_G^\top$. The Woodbury identity gives

$$(D_G + L_G L_G^\top)^{-1} = D_G^{-1} - D_G^{-1} L_G \left(I_p + L_G^\top D_G^{-1} L_G\right)^{-1} L_G^\top D_G^{-1}. \tag{36}$$

Only

$$T_P := I_p + L_G^\top D_G^{-1} L_G \in \mathbb{R}^{p\times p} \tag{37}$$

requires a dense factorization. Because $p \le P$ is small, this replaces a dense $G \times G$ solve with diagonal operations in G and a dense $p \times p$ solve.

Step 5: Identify what is and is not approximated. The flush prepares the factors above in $O(KVp + Vp^2 + Gp^2 + p^3)$ arithmetic, without forming the exact $K \times G$ cross-product or factoring its $G \times G$ Gram matrix. The recurrent state and the sketch remain exact; only the state metric used to compute the coefficient vector is replaced by $\widehat{H}$.

Flush-step latency. Table 2 compares the exact regularized coefficient map with the $P = 4$ approximation, using the same sketch bases and rank allocations. Both paths fuse coefficient construction into the state update, without an additional full-state read. By reducing the cross-product and Gram-system computation described above, $P = 4$ reduces complete flush-step latency by 4.02–$4.83\times$ across Mamba-2, GDN, and KDA.

Table 2: Complete flush-step latency on one NVIDIA B300 at batch size 256 and $W = 16$, with FP32 states and BF16 sketch/coefficient map. Kernel measurements use synthetic activations and average the per-layer median latency over all recurrent layers (seven timing rounds). Exact uses the full Gram system with $\lambda = 0.1$; speedup is Exact latency divided by $P = 4$ latency.

Model	$\bar{G}$	Exact (μs)	$P = 4$ (μs)	Speedup
Nemotron 3 Super	5	1,854.55	420.20	$4.41\times$
Qwen3.8 Flash-Next	7	2,120.23	439.17	$4.83\times$
GLM 5.3 Flash	7	2,754.81	685.84	$4.02\times$

B SketchSSM Algorithm

Algorithm 1 summarizes SketchSSM in Section 4 for one state head. The sketching matrix Ω is calibrated offline and fixed throughout inference. After initialization, each window contains $W-1$ non-flush steps that read U and C, followed by one flush step that reads S_0 and applies the buffered updates from $\mathcal{B}$ exactly. The flush step computes the output from the updated full state and constructs U, C before writing the state back to HBM.

Algorithm 1 SketchSSM calibration and inference.

Require: Calibration distribution $\mathcal{D}$, mean rank budget $\bar{G}$, window size W

Offline

Calibrate $\Omega^\star$ using Equation 8
Allocate $G_{l,h}$ to each state head under $\bar{G}$ using Equation 49
Fix Ω to the first $G_{l,h}$ columns of $\Omega^\star$ for each head

Inference

Initialize the full state S_0 from prefill
Construct the initial U, C from S_0; initialize buffer $\mathcal{B} \leftarrow \varnothing$
while decoding continues **do**
    Non-flush steps: $W-1$ **times per window**
    for $t = 1, \ldots, W-1$ **do**
        Append the current update inputs to $\mathcal{B}$
        Compute $\tilde{q}_t$ and o_t^{buffer} exactly from $\mathcal{B}$ using Equation 4
        Read U, C from HBM
        $o_t \leftarrow UC\tilde{q}_t + o_t^{\text{buffer}}$
        Emit o_t; terminate if decoding is complete
    end for
    Flush step: once per window, $t = W$
    Append the current update inputs to $\mathcal{B}$
    Read the full state S_0 from HBM
    Update S_0 exactly using all buffered inputs from $\mathcal{B}$
    $o_t \leftarrow S_0^\top q_t$
    Construct the sketch $U \leftarrow S_0^\top \Omega$
    Construct the coefficient map C from S_0 via Equation 9 and Appendix A.2.2
    Write S_0, U, C to HBM
    Clear $\mathcal{B}$; emit o_t
end while

C WY-BASED EFFECTIVE-QUERY AND COEFFICIENT COMPUTATION

We adapt the WY representation used in chunkwise delta-rule computation (Yang et al., 2024b) to the decay-and-erase transitions in Equation 3. We first derive an exact effective-query evaluation without a sketch or coefficient map, then describe its implementation in coefficient space.

C.1 PARALLEL EFFECTIVE-QUERY EVALUATION

Within a window indexed by $t = 1, \dots, W$, the effective query is

$$\tilde{q}_t = M_{1:t}^\top q_t, \qquad M_{1:t} = M_t \cdots M_1. \tag{38}$$

Applying each transposed transition to the query separately creates a serial dependency chain. A WY-style factorization replaces this chain with a sum of contributions from buffered tokens.

Factor the transition product. For GDN and KDA,

$$M_s = (I - \beta_s k_s k_s^\top) D_s = D_s - \beta_s k_s (k_s^\top D_s), \tag{39}$$

where D_s is diagonal, including scalar decay as a special case. Let $d_{[t]} = \mathrm{diag}(D_t \cdots D_1)$ and

$$\ell_s(t) := (D_t \cdots D_{s+1}) k_s, \tag{40}$$

with an empty matrix product equal to the identity. The ordered product has the exact form

$$M_{1:t} = \mathrm{Diag}(d_{[t]}) - \sum_{s=1}^{t} \ell_s(t) \pi_s^\top, \tag{41}$$

where the erase factors satisfy

$$\pi_s = \beta_s \left(d_{[s]} \odot k_s - \sum_{j=1}^{s-1} \langle \ell_j(s), k_s \rangle \pi_j \right) \in \mathbb{R}^K. \tag{42}$$

To verify this form, the base case is $\pi_1 = \beta_1 (d_{[1]} \odot k_1)$. At step s, multiplication by D_s updates each existing left factor to $\ell_j(s)$. Multiplication by $I - \beta_s k_s k_s^\top$ then adds $-k_s \pi_s^\top$, with π_s given above. Thus the recurrence retains all cross terms between erase operations.

Apply the product to the query. Transposing Equation 41 gives

$$\tilde{q}_t = d_{[t]} \odot q_t - \sum_{s=1}^{t} \pi_s \langle \ell_s(t), q_t \rangle. \tag{43}$$

Once the factors have been maintained as tokens arrive, their contributions to the current query can be accumulated in parallel over buffer slots. This evaluation is exact and requires no read of S_0; it does not remove the causal dependence between arriving tokens. For Mamba-2, the erase terms are absent and the effective query reduces to $d_{[t]} \odot q_t$.

C.2 COEFFICIENT COMPUTATION WITH PROJECTED FACTORS

We need $C\tilde{q}_t$, not $\tilde{q}_t$ itself. Applying C to Equation 43 and defining $f_s := C\pi_s \in \mathbb{R}^G$ gives

$$c_t^\star = C(d_{[t]} \odot q_t) - \sum_{s=1}^{t} f_s \langle \ell_s(t), q_t \rangle. \tag{44}$$

Applying C to the recurrence for π_s likewise gives

$$f_s = \beta_s \left[C(d_{[s]} \odot k_s) - \sum_{j=1}^{s-1} f_j \langle \ell_j(s), k_s \rangle \right]. \tag{45}$$

For a fixed coefficient map, these identities compute $C\tilde{q}_t$ exactly; projecting the factors introduces no additional approximation. Thus, once slot s is appended, later steps need only its G-dimensional f_s, rather than its K-dimensional π_s.

Reusing buffered keys. Because the D_s matrices are diagonal, $\ell_s(t) = (d_{[t]} \oslash d_{[s]}) \odot k_s$. Hence, for any $y \in \mathbb{R}^K$,

$$\langle \ell_s(t), y \rangle = \langle k_s \oslash d_{[s]}, d_{[t]} \odot y \rangle. \tag{46}$$

The kernel can compute the required scalar from the stored ring key and the cumulative diagonal, so no additional K-vector is needed per slot. Both sums in Equations 44 and 45 are parallel reductions over ring slots. In addition to the buffered inputs, this calculation uses the coefficient map $C \in \mathbb{R}^{G \times K}$ plus one $f_s \in \mathbb{R}^G$ per slot for erasing transitions. Mamba-2 has no erase, so the f_s vectors and the two sums disappear. None of these transformations changes the raw ring transitions used for the exact full-state update at a flush.

D State Memory Traffic Reduction

Table 3 compares amortized state-memory traffic, including state-read and state-write accesses, in bytes per state head per decode step. Standard reads and writes the full FP32 $K \times V$ state at every step, giving a reference cost of $8KV$ bytes. Let $D := 4KV$ denote the full-state size, and let C_{rate} be the effective state compression ratio, including auxiliary metadata such as quantization scales.

State Write. Standard writes D bytes per step. Quantization and pruning reduce this cost to D/C_{rate}, as they update the compressed state at every step. ReplaySSM and SketchSSM instead buffer updates and write the full FP32 state once every W steps. Their state-write traffic is zero at non-flush steps and D bytes at flush steps, averaging D/W bytes per step.

State Read. Standard and ReplaySSM read D bytes per step, while quantization and pruning read D/C_{rate} bytes. SketchSSM stores the sketch U and coefficient map C in BF16, so reading both at a non-flush step costs $2G(V + K)$ bytes. For GDN and KDA, the BF16 projected erase vectors (Appendix C.2) add $2GW$ bytes. The total non-flush read cost is therefore $\mathcal{T}_G := 2G(V + K + eW)$, with $e = 0$ for Mamba-2 and $e = 1$ for GDN and KDA. Each flush reads the full FP32 state to apply the buffered updates and refresh these representations. Thus, SketchSSM's average read cost is $B_G := ((W - 1)\mathcal{T}_G + D)/W$.

Table 3: Average state-memory traffic per state head per decode step, including reads and writes, and reduction relative to Standard. $D = 4KV$, $B_G = ((W-1)\mathcal{T}_G+D)/W$, and $\mathcal{T}_G = 2G(V+K+eW)$. Full states use FP32; sketches, coefficient maps, and projected erase vectors use BF16.

Method	Read	Write	Total	Reduction
Standard	D	D	$2D$	1
ReplaySSM	D	D/W	$D + D/W$	$\frac{2W}{W+1}$
Quant./Pruning	D/C_{rate}	D/C_{rate}	$2D/C_{\text{rate}}$	C_{rate}
SketchSSM	B_G	D/W	$B_G + D/W$	$\frac{2D}{B_G+D/W}$

For a sketched head, the state-memory traffic reduction is

$$R_G = \frac{8KV}{(W - 1)\mathcal{T}_G/W + 8KV/W}. \tag{47}$$

For head-dependent ranks, we sum the byte costs across heads before taking the ratio. Dense-fallback heads read D bytes per step and write D/W bytes on average. At $W = 16$, ReplaySSM reduces this traffic by $32/17 \approx 1.88\times$. Quantization and pruning retain their compression ratios because both reads and writes shrink proportionally. This metric excludes much smaller ring-buffer traffic, writes to sketches and coefficient metadata; these costs are reflected in the measured speedups.

E OFFLINE SKETCH-RANK ALLOCATION

For each state head, we use the first G columns of the offline basis from Section 4.1 to form a rank-G sketch. We keep the basis fixed and compare candidate ranks by their state-read reconstruction errors and the loss sensitivity of those errors.

Head-level calibration. For head $j = (l, h)$, let $G_{\max}$ be the largest candidate rank and $U_{j,G_{\max}} \in \mathbb{R}^{V \times G_{\max}}$ its state sketch in one calibration window. A thin QR factorization $U_{j,G_{\max}} = Q_j R_j$ evaluates all prefix residuals without solving a separate least-squares problem for every rank:

$$\left\|\delta^o_{t,j}(G)\right\|_2^2 = \left\|o^{\text{state}}_{t,j}\right\|_2^2 - \left\|Q^\top_{j,1:G} o^{\text{state}}_{t,j}\right\|_2^2. \tag{48}$$

On the same token, we record $g_{t,j} = \nabla_{o^{\text{state}}_{t,j}} \ell$ and accumulate the paired score $J_j(G)$ from Equation (11). This measures whether the discarded output is aligned with a loss-sensitive direction, rather than multiplying independently averaged error and gradient statistics. We also retain $\varepsilon_j(G) = \mathbb{E}_t[\|\delta^o_{t,j}(G)\|_2^2]$ for the gradient-free ablation. Basis fitting and allocation use disjoint calibration examples.

The score is a separable first-order surrogate: it preserves the sample-wise gradient–residual interaction within each head but omits cross-head terms in $(\sum_j g^\top_{t,j}\delta^o_{t,j})^2$. Its advantage is that the resulting per-head curves can be optimized offline under an exact traffic budget.

Budgeted optimization. Let $\tau_{\text{sk}} := 2(V + K + eW)$ be the read traffic in bytes contributed by one unit of sketch rank with BF16 storage. A full-state read costs $4KV$ bytes because the state remains in FP32. The largest admissible rank strictly cheaper than a full-state read is

$$G^\star := \min\left\{K, V, \left\lceil \frac{4KV}{\tau_{\text{sk}}} \right\rceil - 1\right\}.$$

Each head chooses either a prefix rank in $\{1, \ldots, G^\star\}$ or a full-state read:

$$\begin{gathered}\mathcal{A} := \{1, \ldots, G^\star, \text{full}\}, \qquad \tau(G) := G\tau_{\text{sk}}, \qquad \tau(\text{full}) := 4KV, \qquad J_j(\text{full}) := 0, \\ \min_{\{a_j \in \mathcal{A}\}} \sum_j J_j(a_j) \quad \text{s.t.} \quad \sum_j \tau(a_j) \le B, \qquad B := N\bar{G}\tau_{\text{sk}},\end{gathered} \tag{49}$$

where N is the number of state heads and $\bar{G}$ is the mean sketch-rank budget. The flush read cost is identical for every action and therefore does not affect this optimization.

The measured rank curves need not have diminishing returns, so a sequential greedy rule is not guaranteed to minimize Equation (49). We instead introduce a traffic price $\mu \ge 0$. For any price,

$$D(\mu) := \sum_j \min_{a \in \mathcal{A}} \{J_j(a) + \mu\tau(a)\} - \mu B \tag{50}$$

is a lower bound on the budgeted optimum. The price makes the choices independent across heads: each head selects its lowest penalized score, and bisection adjusts μ until their total traffic reaches the budget. We retain the best feasible table encountered and spend any remaining discrete slack through improving single-head substitutions.

If the returned feasible table has objective P, then $(P - D(\mu))/P$ certifies its relative gap from the optimum of the measured separable objective. This optimization runs only during offline calibration; inference stores and applies the resulting fixed rank table.

F Evaluation Details

F.1 Detailed Evaluation Setup

Offline calibration. We use the training split of WikiText-2 (`wikitext-2-raw-v1`) (Merity et al., 2016), tokenized with each model's tokenizer. Basis calibration uses 65,536 input tokens per model. We collect window-boundary states and effective queries with $W = 16$ and estimate the state-weighted query statistics described in Appendix A.1 to construct the sketching matrix. For rank allocation, we use 8,192 post-warmup token positions per model. We compute gradients of the mean next-token cross-entropy loss and pair each head's output gradient with its reconstruction residual to score candidate ranks (Appendix E). These scores use ideal least-squares reconstruction, while inference uses the coefficient approximation in Appendix A.2.2. Calibration leaves model weights unchanged, and the resulting sketching matrices and rank allocations remain fixed during inference.

Accuracy evaluation setup. SketchSSM settings are mean sketch-rank budgets $\bar{G}$. The coefficient approximation uses $P = 4$ pivots for all four models. SketchSSM maintains the full recurrent state in FP32 and stores both the sketch U and the coefficient map C in BF16. ReplaySSM and SketchSSM use a window size of $W = 16$. We evaluate the Mamba-2 models using vLLM 0.27.0, and Qwen3.8 Flash-Next (GDN) and GLM 5.3 Flash (KDA) using vLLM 0.28.1. For the ReplaySSM baseline, we use the official implementation integrated into vLLM for Mamba-2-based models, the implementation from the authors' public repository[1] for the GDN model, and our own implementation integrated into vLLM for the KDA model, for which no official implementation is available. We report full-state results for both Standard execution (without ReplaySSM) and ReplaySSM. We evaluate MATH-500 (500 problems), AIME25 (30), GPQA Diamond (198), and LiveCodeBench (315, split `v5_2407_2412`). Generation caps are 32,768 tokens for Nemotron Nano and 65,536 for Nemotron Super, Qwen 3.8 Flash-Next, and GLM 5.3 Flash.

Speed Evaluation Setup. We measure kernel latency on one NVIDIA B300 with $W = 16$ and batch sizes 128, 256, and 512 (Fig. 7). The sketch-rank budgets are $\bar{G} \in \{5, 9\}$ for Nemotron Super, $\{7, 11\}$ for Qwen3.8 Flash-Next, and $\{7, 12\}$ for GLM 5.3 Flash. Super and Qwen, GLM profiles average all 40, 36 and 34 recurrent layers, respectively, over six steady windows. Because no official ReplaySSM implementation is available for KDA, we implement the KDA ReplaySSM baseline ourselves. Flush overhead is measured against the same SketchSSM flush kernel with sketch and coefficient-map construction disabled. For Nemotron Super, we also measure decode throughput at 2K and 8K input lengths, up to each method's maximum batch size (Fig. 8). Results average two 128-step runs using six steady windows. The model-forward latency breakdown uses common maximum batches of 912 at 2K and 808 at 8K.

F.2 Accuracy Results on Decode Benchmarks

Tables 4–7 report the data plotted in Fig. 5. Accuracy is in percent, and verbosity is the mean number of generated tokens normalized by each model's baseline. The traffic column includes flush traffic and DSQ's dynamic scale-factor loads.

Verbosity. We measure normalized verbosity as the mean generated-token count relative to Standard for each benchmark, averaged across the four benchmarks. At rank budgets $\bar{G} = 10, 9, 11, 4$ for Nemotron Nano, Nemotron Super, Qwen3.8 Flash-Next, and GLM 5.3 Flash, respectively, state-access traffic reductions range from 8.93–12.81×, with normalized verbosity of 1.05, 1.00, 0.97, and 0.97. At lower rank budgets $\bar{G} = 2, 2, 3, 3$, respectively, traffic reductions reach 13.48–13.88×, with normalized verbosity of 1.42, 1.11, 1.31, and 1.10. These results illustrate a tradeoff between state-access traffic reduction and generated response length.

[1] `https://github.com/Johnny-Liou/ReplaySSM`

Table 4: Nemotron Nano v2 9B: numerical results underlying Fig. 5. ReplaySSM is additionally reported. Accuracies are in percent; token counts are mean generated tokens. Traffic is state-read plus state-write traffic reduction relative to Standard. Average is the unweighted mean accuracy across the four benchmarks.

			MATH-500		AIME25		GPQA-D		LiveCodeBench		Average
Method	Setting	Traffic	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.
Standard	full-state	1.00×	97.40	4,998	69.60	19,131	58.60	12,462	67.90	14,056	73.38
ReplaySSM	full-state	1.88×	97.20	5,118	73.33	19,190	59.60	12,458	66.35	14,459	74.12
SketchSSM	$\bar{G} = 21$	6.15×	96.40	5,400	74.58	19,180	56.57	12,655	66.98	14,385	73.63
SketchSSM	$\bar{G} = 10$	9.08×	97.00	5,212	70.83	19,287	62.12	12,379	66.03	15,980	74.00
SketchSSM	$\bar{G} = 6$	10.98×	96.80	5,701	69.17	20,165	58.08	13,274	63.49	16,996	71.88
SketchSSM	$\bar{G} = 4$	12.26×	96.40	6,447	56.25	22,235	58.59	14,054	53.97	18,933	66.30
SketchSSM	$\bar{G} = 2$	13.88×	90.40	8,229	40.00	24,394	52.02	14,900	38.41	22,201	55.21
GHOST	37.5%	1.60×	86.00	7,525	32.10	22,666	36.40	13,890	43.50	19,061	49.50
GHOST	50%	2.00×	66.00	11,156	12.50	26,059	22.70	18,283	23.50	24,053	31.18
GHOST	62.5%	2.67×	51.00	14,995	2.50	30,924	13.60	22,710	14.60	26,609	20.43
GHOST	75%	4.00×	1.00	27,447	0.00	28,530	0.00	25,613	0.00	31,098	0.25
DSQ	10 bit	3.10×	93.20	7,349	37.10	26,878	48.00	18,206	51.10	21,227	57.35
DSQ	8 bit	3.84×	85.40	9,585	17.50	30,167	40.90	21,377	31.40	24,748	43.80
DSQ	6 bit	5.06×	70.60	15,208	8.30	31,599	27.80	24,186	12.70	28,807	29.85
DSQ	4 bit	7.40×	9.40	29,462	0.00	32,102	0.00	32,185	5.40	30,798	3.70

Table 5: Nemotron 3 Super 120B-A12B: numerical results underlying Fig. 5. ReplaySSM is additionally reported. Accuracies are in percent; token counts are mean generated tokens. Traffic is state-read plus state-write traffic reduction relative to Standard. Average is the unweighted mean accuracy across the four benchmarks.

			MATH-500		AIME25		GPQA-D		LiveCodeBench		Average
Method	Setting	Traffic	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.
Standard	full-state	1.00×	97.80	3,231	83.33	23,174	77.27	19,222	82.86	19,544	85.32
ReplaySSM	full-state	1.88×	97.80	3,495	83.80	22,531	74.20	19,795	77.80	20,332	83.40
SketchSSM	$\bar{G} = 20$	5.80×	97.80	3,564	81.67	22,701	77.78	17,669	80.95	19,722	84.55
SketchSSM	$\bar{G} = 9$	8.93×	98.60	3,566	83.75	24,221	76.77	17,544	77.46	18,292	84.14
SketchSSM	$\bar{G} = 5$	11.12×	98.00	3,899	81.25	25,364	76.26	18,173	76.83	20,492	83.08
SketchSSM	$\bar{G} = 3$	12.66×	97.60	4,004	82.08	24,945	78.28	16,411	73.97	19,745	82.98
SketchSSM	$\bar{G} = 2$	13.61×	97.80	4,324	75.42	28,693	72.22	15,574	69.21	20,954	78.66
GHOST	37.5%	1.60×	97.40	4,866	77.10	28,365	68.70	16,654	68.30	19,991	77.88
GHOST	50%	2.00×	93.80	7,904	49.60	43,232	60.10	17,834	54.30	23,895	64.45
GHOST	62.5%	2.67×	72.20	21,121	11.70	60,528	48.00	26,047	26.30	31,228	39.55
GHOST	75%	4.00×	26.40	45,937	0.00	63,443	5.60	56,059	3.20	59,983	8.80
DSQ	10 bit	3.08×	98.00	3,794	73.30	30,457	70.70	21,955	73.30	23,033	78.83
DSQ	8 bit	3.82×	96.60	5,075	62.50	33,506	67.20	22,668	69.80	24,807	74.03
DSQ	6 bit	5.02×	93.40	7,431	52.50	39,523	62.60	25,577	57.10	32,716	66.40
DSQ	4 bit	7.31×	68.80	21,032	5.80	60,756	33.30	42,752	17.80	51,194	31.42

Table 6: Qwen3.8 Flash Next: numerical results underlying Fig. 5. ReplaySSM is additionally reported. Accuracies are in percent; token counts are mean generated tokens. Traffic is state-read plus state-write traffic reduction relative to Standard. Average is the unweighted mean accuracy across the four benchmarks.

			MATH-500		AIME25		GPQA-D		LiveCodeBench		Average
Method	Setting	Traffic	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.
Standard	full-state	1.00×	98.80	1,564	93.75	19,376	90.91	14,666	80.32	22,936	90.94
ReplaySSM	full-state	1.88×	97.60	1,946	92.50	20,547	87.88	15,638	76.51	23,023	88.62
SketchSSM	$\bar{G} = 26$	6.11×	99.40	1,275	97.08	17,344	91.41	13,624	84.13	19,898	93.01
SketchSSM	$\bar{G} = 11$	9.50×	98.60	1,573	92.08	19,161	88.38	13,797	83.81	21,313	90.72
SketchSSM	$\bar{G} = 7$	11.14×	98.60	1,663	94.58	20,541	87.88	16,681	77.14	24,571	89.55
SketchSSM	$\bar{G} = 4$	12.81×	98.40	2,037	90.83	24,319	87.88	17,988	75.56	27,322	88.17
SketchSSM	$\bar{G} = 3$	13.48×	98.00	2,244	89.58	25,945	84.85	17,752	68.89	28,704	85.33
DRRQR	37.5%	1.60×	96.00	1,744	92.92	20,167	85.86	13,603	76.51	21,005	87.82
DRRQR	50%	2.00×	95.00	2,008	89.58	20,705	79.29	11,735	66.98	24,840	82.72
DRRQR	62.5%	2.67×	92.20	3,276	78.75	22,805	74.24	10,918	51.11	28,072	74.08
DRRQR	75%	4.00×	9.40	1,548	0.42	5,926	2.53	1,879	0.00	1,969	3.09
DSQ	10 bit	3.12×	98.40	1,477	92.92	21,182	85.86	16,598	76.83	22,605	88.50
DSQ	8 bit	3.88×	98.20	1,900	71.25	29,344	75.25	21,650	68.25	26,888	78.24
DSQ	6 bit	5.12×	94.80	2,952	38.33	24,680	48.99	17,251	37.78	19,475	54.98
DSQ	4 bit	7.53×	57.40	2,162	0.42	4,872	9.09	3,928	13.97	3,756	20.22

Table 7: GLM 5.3 Flash: numerical results underlying Fig. 5. ReplaySSM is additionally reported. Accuracies are in percent; token counts are mean generated tokens. Traffic is state-read plus state-write traffic reduction relative to Standard. Average is the unweighted mean accuracy across the four benchmarks.

			MATH-500		AIME25		GPQA-D		LiveCodeBench		Average
Method	Setting	Traffic	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.
Standard	full-state	1.00×	97.80	1,087	84.58	21,248	88.38	11,582	74.60	24,657	86.34
ReplaySSM	full-state	1.88×	98.20	1,142	83.75	20,417	89.39	11,848	74.92	24,881	86.57
SketchSSM	$\bar{G} = 28$	5.83×	98.00	835	86.67	18,142	90.91	10,273	80.32	21,264	88.97
SketchSSM	$\bar{G} = 12$	9.16×	97.40	917	85.42	16,290	89.39	9,694	80.32	19,312	88.13
SketchSSM	$\bar{G} = 7$	11.14×	97.60	1,271	85.83	18,126	87.88	10,925	79.05	19,701	87.59
SketchSSM	$\bar{G} = 4$	12.81×	97.40	1,273	81.25	19,444	84.85	11,715	66.35	18,966	82.46
SketchSSM	$\bar{G} = 3$	13.48×	97.00	1,466	78.75	19,604	76.26	15,471	60.00	19,344	78.00
GHOST	37.5%	1.60×	83.00	8,623	15.83	53,798	46.97	25,635	16.83	54,240	40.66
GHOST	50%	2.00×	66.60	16,894	4.58	58,929	25.25	40,181	9.52	59,656	26.49
GHOST	62.5%	2.67×	37.20	35,219	0.83	63,749	7.07	54,124	2.54	62,403	11.91
GHOST	75%	4.00×	3.20	62,161	0.00	64,948	3.03	61,297	0.00	64,747	1.56
DSQ	10 bit	3.12×	97.40	921	81.25	19,951	87.88	10,294	73.65	24,183	85.04
DSQ	8 bit	3.88×	96.20	998	82.50	20,791	88.38	10,625	70.79	24,455	84.47
DSQ	6 bit	5.12×	93.60	2,744	37.50	29,547	55.05	21,322	36.51	30,678	55.66
DSQ	4 bit	7.53×	57.80	15,839	0.00	36,383	7.07	45,971	0.32	48,541	16.30

F.3 RECALL EVALUATION

Evaluation setup. We evaluate long-context recall on four retrieval tasks from RULER (Hsieh et al., 2024): Single UUID (`niah_single_3`), Multi-key (`niah_multikey_2`), Multi-value (`niah_multivalue`), and Multi-query (`niah_multiquery`). These tasks cover retrieving a UUID, identifying a value among distractor key–value pairs, retrieving four values for one key, and answering four queries, respectively. We use 500 examples per task at a 16K context budget, counted with each model's tokenizer, and reuse identical inputs across methods within each model. Generation is greedy with a cap of 128 tokens. We follow RULER's answer-coverage metric: the fraction of reference answers found in the generated text by case-insensitive substring matching, averaged over examples and reported as a percentage. Average is the unweighted mean of the four task scores. We wrap the task in each model's chat template and configure direct-answer generation. We retain RULER's default answer prefix, except for Qwen3.8 Flash-Next, for which we observed premature termination with this prefix. We therefore omit it consistently across all evaluated methods for this model, while retaining its native non-thinking template. For GLM 5.3 Flash, we use a custom direct-answer template that removes the reasoning-effort instruction and closes the initial thinking block. We compare Standard execution, ReplaySSM with $W = 16$, SketchSSM with $W = 16$ and $P = 4$, and state pruning at 50%, 62.5%, and 75% sparsity. SketchSSM uses fixed offline-calibrated bases and rank allocations, without calibration on the recall examples; its full state is FP32 and its sketch and coefficient map are stored in BF16.

Results. On Nemotron Nano, SketchSSM achieves average recall scores of 97.11–97.63% across the reported ranks, compared with 98.29% for Standard execution and 98.25% for ReplaySSM. The largest task-level decrease occurs on Multi-value at $\bar{G} = 6$, from 97.45% for Standard execution to 91.00%. On Nemotron Super, SketchSSM matches both full-state methods at 100% on all four tasks for each reported rank. Qwen3.8 Flash-Next also scores 100% on all four tasks for Standard execution, ReplaySSM, and SketchSSM at all three reported ranks. On GLM 5.3 Flash, SketchSSM achieves average scores of 99.49–99.71%, compared with 99.40% for Standard execution and 99.46% for ReplaySSM. State pruning causes larger losses: at 75% sparsity, average scores are 76.70% for Nemotron Nano, 53.03% for Nemotron Super, 43.05% for Qwen3.8 Flash-Next, and 46.78% for GLM 5.3 Flash. The task-level results below expose differences that an average alone would obscure.

Table 8: Recall scores (%) on four RULER retrieval tasks at 16K context for Nemotron Nano. Traffic is the state-read plus state-write traffic reduction relative to Standard, using the same accounting as the accuracy tables. Average is the unweighted mean of the four task scores.

Method	Setting	Traffic	Single UUID	Multi-key	Multi-value	Multi-query	Average
Standard	full-state	1.00×	100.00	99.80	97.45	95.90	98.29
ReplaySSM	full-state	1.88×	100.00	99.80	97.20	96.00	98.25
SketchSSM	$\bar{G} = 21$	6.15×	100.00	99.80	94.60	96.10	97.63
SketchSSM	$\bar{G} = 10$	9.08×	100.00	99.80	93.20	96.90	97.48
SketchSSM	$\bar{G} = 6$	10.98×	100.00	99.80	91.00	97.65	97.11
GHOST	50%	2.00×	100.00	85.60	93.45	95.55	93.65
GHOST	62.5%	2.67×	99.80	39.40	96.10	94.30	82.40
GHOST	75%	4.00×	99.40	22.60	89.95	94.85	76.70

Table 9: Recall scores (%) on four RULER retrieval tasks at 16K context for Nemotron Super. Traffic is the state-read plus state-write traffic reduction relative to Standard, using the same accounting as the accuracy tables. Average is the unweighted mean of the four task scores.

Method	Setting	Traffic	Single UUID	Multi-key	Multi-value	Multi-query	Average
Standard	full-state	1.00×	100.00	100.00	100.00	100.00	100.00
ReplaySSM	full-state	1.88×	100.00	100.00	100.00	100.00	100.00
SketchSSM	$\bar{G} = 20$	5.80×	100.00	100.00	100.00	100.00	100.00
SketchSSM	$\bar{G} = 9$	8.93×	100.00	100.00	100.00	100.00	100.00
SketchSSM	$\bar{G} = 5$	11.12×	100.00	100.00	100.00	100.00	100.00
GHOST	50%	2.00×	100.00	90.20	99.60	99.65	97.36
GHOST	62.5%	2.67×	99.40	44.80	93.65	95.20	83.26
GHOST	75%	4.00×	96.60	1.20	57.90	56.40	53.03

Table 10: Recall scores (%) on four RULER retrieval tasks at 16K context for Qwen3.8 Flash-Next. Traffic is the state-read plus state-write traffic reduction relative to Standard, using the same accounting as the accuracy tables. Average is the unweighted mean of the four task scores.

Method	Setting	Traffic	Single UUID	Multi-key	Multi-value	Multi-query	Average
Standard	full-state	1.00×	100.00	100.00	100.00	100.00	100.00
ReplaySSM	full-state	1.88×	100.00	100.00	100.00	100.00	100.00
SketchSSM	$\bar{G} = 26$	6.11×	100.00	100.00	100.00	100.00	100.00
SketchSSM	$\bar{G} = 11$	9.50×	100.00	100.00	100.00	100.00	100.00
SketchSSM	$\bar{G} = 7$	11.14×	100.00	100.00	100.00	100.00	100.00
DRRQR	50%	2.00×	99.80	100.00	99.75	100.00	99.89
DRRQR	62.5%	2.67×	99.40	98.60	97.10	99.25	98.59
DRRQR	75%	4.00×	65.20	21.60	33.90	51.50	43.05

Table 11: Recall scores (%) on four RULER retrieval tasks at 16K context for GLM 5.3 Flash. Traffic is the state-read plus state-write traffic reduction relative to Standard, using the same accounting as the accuracy tables. Average is the unweighted mean of the four task scores.

Method	Setting	Traffic	Single UUID	Multi-key	Multi-value	Multi-query	Average
Standard	full-state	1.00×	99.80	100.00	97.80	100.00	99.40
ReplaySSM	full-state	1.88×	99.80	100.00	98.05	100.00	99.46
SketchSSM	$\bar{G} = 28$	5.83×	99.60	100.00	98.40	99.95	99.49
SketchSSM	$\bar{G} = 12$	9.16×	99.80	100.00	99.10	99.95	99.71
SketchSSM	$\bar{G} = 7$	11.14×	100.00	100.00	98.85	100.00	99.71
GHOST	50%	2.00×	100.00	98.80	82.10	100.00	95.23
GHOST	62.5%	2.67×	99.80	95.00	73.20	96.55	91.14
GHOST	75%	4.00×	61.20	47.20	47.30	31.40	46.78

G Ablation of Preserving Full-State Updates with Approximate Reads

We evaluate the effect of preserving the full recurrent state for updates while using a compact representation only for reads. To this end, we construct ablation variants within our proposed design that differ in their read representation: a quantized state copy, a low-rank state approximation, or a sketch. The low-rank baseline builds on classical truncated SVD (Eckart & Young, 1936), using a fixed basis obtained offline from the leading eigenvectors of the average state Gram matrix, with direct query projection for coefficients. Unlike the sketch, this variant uses fixed rather than state-dependent coefficients.

As discussed in Section 2, compressing the recurrent state itself introduces errors that propagate through subsequent state updates. Our design instead retains the full state for these updates and applies approximation only to the read path. Because updates occur only at flush steps, their full-state traffic is amortized over multiple decode steps.

Each variant constructs its compact read representation from the updated state at a flush step and reads it between flushes. All variants retain the full state for subsequent updates; they are ablations of the read representation within our design.

Table 12 reports results for quantized read-only copies, low-rank state representations, and sketches, with state quantization included as a reference. Retaining the full state for updates substantially improves accuracy over quantizing the recurrent state itself. Within this design, using a sketch further improves average accuracy over reading a quantized state copy at comparable state-access traffic reductions.

Table 12: Accuracy ablation for maintaining full states and approximating reads on Nemotron Nano v2 9B ($W = 16$). Ours decouples full-state updates from approximate reads and varies only the read representation. Traffic is the state-read plus state-write traffic reduction relative to Standard; sketch reads store the sketch rows and coefficient map in BF16. Low-rank traffic includes BF16 reads of both the compressed state and fixed projection map, using the same per-head ranks as the sketch.

Methods	Update	Read	Traffic	MATH-500	AIME25	GPQA-D	LCB	Average
Baseline	full state	full state	$1.00\times$	97.40	69.60	58.60	67.90	73.38
Baseline	6-bit state	6-bit state	$5.06\times$	70.60	8.30	27.80	12.70	29.85
Ours	full state	4-bit state copy	$7.95\times$	97.40	68.33	55.05	63.49	71.07
Ours	full state	low-rank state, $\bar{G} = 10$	$9.08\times$	94.40	54.58	54.04	42.54	61.39
Ours (SketchSSM)	full state	sketch, $\bar{G} = 10$	$9.08\times$	97.00	70.83	62.12	66.03	74.00
Baseline	4-bit state	4-bit state	$7.40\times$	9.40	0.00	0.00	5.40	3.70
Ours	full state	2-bit state copy	$10.36\times$	88.20	30.00	39.39	26.98	46.14
Ours	full state	low-rank state, $\bar{G} = 6$	$10.98\times$	91.80	50.42	52.53	37.78	58.13
Ours (SketchSSM)	full state	sketch, $\bar{G} = 6$	$10.98\times$	96.80	69.17	58.08	63.49	71.88