QReS: Quantizing Recurrent Language Models Is Easier Than You Think

Anonymous authors
Paper under double-blind review

Abstract

Linear recurrences are becoming an increasingly common building block in large language models, replacing the growing KV cache of softmax attention with a fixed-size state that enables efficient decoding. However, the quantization stack has largely been developed around softmax Transformers, whose numerical properties and inference state differ from those of recurrent models, motivating quantization schemes tailored to recurrent structure. We study how quantizing linear recurrences differs from quantizing softmax attention along two axes. First, extra-state quantization concerns weights and activations outside the recurrent state, whose distributions are shaped indirectly by the sequence mixer during training. In controlled 1.3B models, we find that decayed recurrences develop much weaker residual outlier channels than softmax attention and are therefore easier to quantize without incoherence processing. Second, intra-state quantization concerns the recurrent state itself during decode, where repeated updates and recurrent decay create a different numerical problem from KV cache quantization. We use these insights to develop QReS, a pipeline that combines value-axis scale grouping, decay-based mixed-precision allocation and stochastic rounding for recurrent state quantization. Across Qwen-3.5 9B, Nemotron-H 8B, and Kimi-Linear 48B-A3B, QReS supports robust 8-bit state quantization and lower effective precision on shorter decode horizons. We implement QReS in vLLM and demonstrate end-to-end decode speedups.

1 Introduction

Softmax attention has shaped not only the architecture of large language models, but also much of the deployment stack built around them. This is particularly true for post-training quantization (PTQ). A large body of prior work investigates quantization of the weights and activations of softmax Transformers (Frantar et al., 2022; Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2025), while a complementary line studies quantization of the key-value cache that grows during autoregressive generation (Liu et al., 2024; Hooper et al., 2024; Duanmu et al., 2024). Concurrently, the space of sequence mixers used in foundation models is expanding. Linear attention and state-space models replace softmax attention with linear recurrences (Katharopoulos et al., 2020; Dao & Gu, 2024; Yang et al., 2023; 2025; Tumma et al., 2026), and recent large-scale models interleave recurrent and softmax layers within the same architecture (Blakeman et al., 2025; Team et al., 2025; Qwen Team, 2026). As these hybrid models become increasingly common, the inference stack developed for softmax Transformers must be adapted accordingly.

Compared with softmax attention, however, the behavior of recurrent language models under quantization remains poorly understood. Existing work has begun to adapt quantization to recurrent architectures (Chiang et al., 2025b;a; Xu et al., 2025; Kim et al., 2026; Chandiramani et al., 2026), but has focused on particular architectures and quantization targets. In this work, we instead ask the more general question: *how does replacing softmax attention with a linear recurrent sequence mixer change the quantization problem?*

We argue that this change matters along two distinct axes. First, the choice of sequence mixer changes the training dynamics of the model and therefore the distributions of the weights and activations learned throughout the network. We refer to quantization of these tensors as *extra-state weight/activation (W/A) quantization*. Second, softmax attention and recurrent mixers maintain fun-

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.

Table 1: State-update rules for recurrent models in the mixer-zoo design space. Colors identify decay, write gating, and delta correction.

model	update
Linear attention (Katharopoulos et al., 2020)	$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top$
Mamba-2 (Dao & Gu, 2024)	$\mathbf{S}_t = \alpha_t\, \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top$
Gated Linear Attention (GLA) (Yang et al., 2023)	$\mathbf{S}_t = \mathbf{S}_{t-1}\,\mathrm{diag}(\boldsymbol{\alpha}_t) + \mathbf{v}_t\mathbf{k}_t^\top$
DeltaNet (Schlag et al., 2021; Yang et al., 2024)	$\mathbf{S}_t = \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top\right) + \beta_t\,\mathbf{v}_t\mathbf{k}_t^\top$
Gated DeltaNet (Yang et al., 2025)	$\mathbf{S}_t = \alpha_t\,\mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top\right) + \beta_t\,\mathbf{v}_t\mathbf{k}_t^\top$
Kimi Delta Attention (KDA) (Team et al., 2025)	$\mathbf{S}_t = \mathbf{S}_{t-1}\,\mathrm{diag}(\boldsymbol{\alpha}_t)\left(\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top\right) + \beta_t\,\mathbf{v}_t\mathbf{k}_t^\top$

damentally different objects during inference. Softmax attention stores a growing cache of keys and values, whereas a recurrent mixer repeatedly updates a fixed-size hidden state. We refer to quantization of this object as *(intra-)state quantization*. Together, these two settings separate the implicit effect of the sequence mixer on the rest of the model from the explicit effect of quantizing the mixer state itself. Our main technical contributions can be summarized as follows:

- For extra-state W/A quantization, we show that decayed linear recurrences develop substantially weaker residual outlier channels than softmax attention, making them more robust to W/A quantization before incoherence processing. Incoherence processing largely closes this gap.
- For intra-state quantization, we develop QReS, a quantization pipeline that combines value-axis grouping, stochastic rounding and decay-based mixed precision, with theoretical and empirical justification for each component.
- We validate these findings in controlled 1.3B-parameter experiments and show that QReS transfers to the large-scale hybrids Qwen-3.5 9B, Nemotron-H 8B, and Kimi-Linear 48B-A3B.
- We implement QReS in vLLM, demonstrating end-to-end decode speedups at scale.

2 Background and Preliminaries

2.1 Sequence Mixers

Softmax attention. Given queries $\mathbf{Q} \in \mathbb{R}^{T,d_k}$, keys $\mathbf{K} \in \mathbb{R}^{T,d_k}$, and values $\mathbf{V} \in \mathbb{R}^{T,d_v}$, causal softmax attention (Vaswani et al., 2017) computes

$$\mathbf{o}_t = \mathbf{V}_{1:t}^\top \,\mathrm{softmax}\left(\mathbf{K}_{1:t}\mathbf{q}_t/\sqrt{d_k}\right).$$

During autoregressive generation, the model stores previously computed keys and values in a key-value (KV) cache and reuses them at subsequent decode steps. As a result, the sequence mixer state grows linearly with sequence length.

Linear attention. Linear attention replaces the nonlinear softmax with an unnormalized linear kernel (Katharopoulos et al., 2020; Schlag et al., 2021). In its simplest form, associativity allows keys and values to be summarized into a fixed-size matrix-valued state $\mathbf{S}_t \in \mathbb{R}^{d_v,d_k}$,

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t.$$

Unlike the KV cache, the recurrent state has constant size and is updated in place after every token. Modern recurrent sequence mixers refine this basic recurrence through mechanisms such as forgetting, gated writes, and delta-rule corrections (Dao & Gu, 2024; Yang et al., 2023; 2025; Team et al., 2025). To isolate these architectural choices, we construct a *mixer zoo* in which the sequence mixer varies while the surrounding layer design and training recipe are fixed. In particular, all models share the backbone around the mixer, including the query, key, and value projections, causal convolutions, normalization, RoPE (Su et al., 2024), and output projection, apart from the explicitly varied query/key normalization (which we denote by $\|\mathbf{QK}\|$). The recurrent variants in the zoo are parameterized by the unified update

$$\mathbf{S}_t = \mathbf{S}_{t-1}\,\mathrm{diag}(\boldsymbol{\alpha}_t)\left(\mathbf{I} - \delta\,\beta_t\,\mathbf{k}_t\mathbf{k}_t^\top\right) + \beta_t\,\mathbf{v}_t\mathbf{k}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t \tag{1}$$

where $\|\mathbf{k}_t\|_2 = \|\mathbf{q}_t\|_2 = 1$, $\beta_t \in (0, 2)$, and the vector $\boldsymbol{\alpha}_t$ has entries $\alpha_{t,i} \in (0, 1]$. In our notation, the indicator $\delta \in \{0, 1\}$ enables delta correction. The update therefore composes three mechanisms:

- **Decay** is a multiplicative forget gate applied to the state before each write. A scalar gate defines one decay rate per attention head, as in Mamba-2 (Dao & Gu, 2024) and Gated DeltaNet (Yang et al., 2025), while a diagonal gate defines a separate rate per key channel, as in Gated Linear Attention (Yang et al., 2023) and Kimi Delta Attention (Team et al., 2025).
- **Write gating** controls the strength of the outer-product update $\mathbf{v}_t\mathbf{k}_t^\top$.
- **Delta correction**, introduced in DeltaNet (Schlag et al., 2021; Yang et al., 2024), applies a rank-one correction along the current key through $(\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^\top)$ before writing the new association.

Table 1 classifies linear attention models using this taxonomy. Our mixer zoo additionally contains softmax attention with and without query/key RMSNorm (Zhang & Sennrich, 2019).

2.2 Quantizing Linear Attention

Figure 1 distinguishes two quantization targets. *Extra-state quantization* concerns the weights and activations (W/A) outside the mixer state, while *intra-state quantization* concerns the state maintained during decode. The distinction between quantizing softmax attention and linear recurrences is implicit in the former, through the representations learned during training, and explicit in the latter, through the object that is stored and updated during inference.

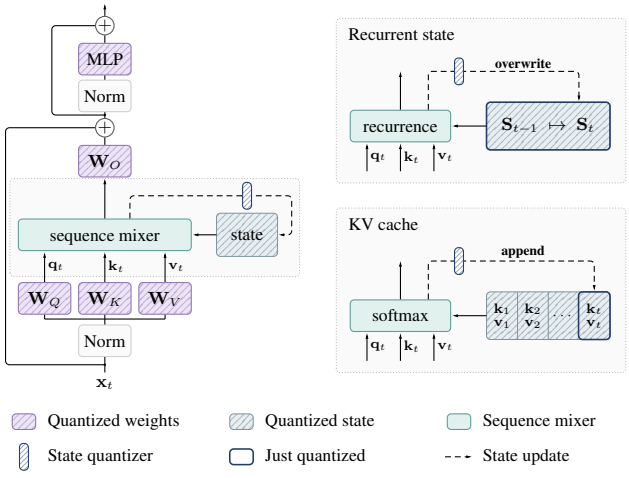


Figure 1: We consider two quantization targets. Extra-state quantization acts on weights and activations outside the mixer state, while intra-state quantization acts on the state used during inference. A recurrent state is repeatedly updated and requantized in place, whereas entries in a KV cache are quantized once and subsequently reused.

Extra-state quantization. Softmax and recurrent models expose the same class of weights and activations to a W/A quantizer, so the same general PTQ machinery can be applied to both (Frantar et al., 2022; Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2025). However, changing the sequence mixer can change the distributions of these tensors learned during training. Prior work on Mamba models has consequently developed architecture-specific treatments for their activation statistics (Chiang et al., 2025b;a; Xu et al., 2025). We instead study this dependence directly: using a common quantization framework and controlled architecture, Section 3 asks how the sequence mixer itself changes W/A quantization.

Intra-state quantization. Here the distinction between sequence mixers is explicit. KV cache quantization methods are designed around keys and values that are appended once and then reused. They exploit this structure through choices such as separate scaling and grouping for keys and values, outlier treatment, channel reordering, and a recent token window kept at higher precision (Liu et al., 2024; Hooper et al., 2024; Duanmu et al., 2024). A linear recurrence instead stores a fixed-size matrix $\mathbf{S}_t$ that is updated after every decoded token. Quantizing this state therefore requires repeatedly dequantizing, updating, and requantizing the same object, allowing quantization error to propagate through subsequent recurrent updates. The state also has two channel axes and may attenuate previously introduced error through recurrent decay.

Previous work on recurrent state quantization during decoding has focused primarily on Mamba models. Q-Mamba (Tianqi et al., 2025) develops scaling methods that account for state and channel statistics in quantized Mamba states, while Quamba2 (Chiang et al., 2025a) stores SSM states in INT8 during generation. Nemotron 3 Super (Chandiramani et al., 2026) studies reduced-precision

state storage during decode and uses stochastic rounding to control accumulated error. SSDi8 (Kim et al., 2026) considers a related but different target, quantizing the chunked Mamba-2 SSD computation used during prefill to maintain a low-precision execution path.

Our goal is different from these model-specific quantization recipes. We study state quantization as a problem across a broad family of linear recurrences, and isolate three properties that govern it: the geometry of scale grouping, error accumulation under repeated requantization, and the attenuation of previous error through recurrent decay. Section 4 develops theoretical and empirical characterizations of each to construct QReS, a recurrent state quantization framework that transfers to large hybrid models.

3 Extra-State Weight and Activation Quantization

Experimental setup. In this section, we investigate how the choice of sequence mixer affects W/A quantization using a 1.3B-parameter mixer zoo (training recipe in Appendix B.1). We apply the same quantization stack to every model: GPTQ (Frantar et al., 2022) with per-output-channel weight scales and dynamic activation scales (Liu et al., 2025) (details in Appendix B.2). Following rotation-based PTQ methods (Ashkboos et al., 2024; Liu et al., 2025), we evaluate three orthogonal rotations to reduce activation outliers. $\mathbf{R}_1$ rotates the residual stream, $\mathbf{R}_4$ rotates the input to the MLP down projection, and $\mathbf{R}_o$ the input to the mixer's output projection. We report relative WikiText-2 perplexity changes, with absolute perplexities in Appendix D.1 (Merity et al., 2016).

Table 2: Relative perplexity change (%) on WikiText-2 in the 1.3B mixer zoo. The first two rows use softmax attention, $\|\mathbf{QK}\|$ denotes QK-norm. Subsequent rows are recurrent mixers. Rotation columns use W4A4.

Architectural choices				No rotation		With rotation (W4A4)		
Decay	$\|\mathbf{QK}\|$	β	Δ	W4A8	W4A4	$\mathbf{R}_1$	$\mathbf{R}_4$	$\mathbf{R}_1\mathbf{R}_4\mathbf{R}_o$
–	–	–	–	5.8	98.4	36.7	47.9	10.6
–	✓	–	–	5.9	92.3	37.8	46.5	11.1
–	✓	–	–	6.8	111.4	48.1	54.0	12.5
–	✓	✓	–	6.9	82.4	47.2	35.9	12.4
–	✓	✓	✓	6.1	68.2	46.5	28.5	12.7
Scal	✓	–	–	6.0	56.6	40.3	24.7	11.9
Scal	✓	✓	–	5.5	53.5	41.3	21.3	11.3
Scal	✓	✓	✓	6.0	55.4	43.2	21.5	11.8
Diag	✓	–	–	4.8	45.0	32.6	18.3	9.3
Diag	✓	✓	–	5.4	49.2	35.7	20.7	10.6
Diag	✓	✓	✓	5.8	47.8	37.0	19.5	11.2

Decayed recurrences are robust. Without rotations, the robustness advantage of decayed linear recurrences over softmax attention emerges at 4-bit activations (Table 2). Within matched recurrent families, decay consistently improves robustness, with diagonal decay performing best. Applying the full rotation stack reduces W4A4 degradation to a similar level across the zoo, suggesting that the advantage of decayed recurrences arises primarily from the representation they learn. The individual rotations further localize this difference: $\mathbf{R}_1$, which acts on the residual stream, is more effective for softmax models. In decayed recurrences, $\mathbf{R}_4$ becomes relatively more important. This motivates the following residual stream analysis:

Residual stream outliers. Figure 2 shows that softmax attention develops a dominant residual channel whose magnitude is approximately constant across tokens (and layers, Figure 8b). Decayed recurrences exhibit weaker outliers. Across the zoo, residual stream kurtosis closely tracks unrotated W4A4 degradation (Figure 8a), showing that models with stronger residual outliers are harder to quantize. We next test how strongly the models rely on this channel (Table 3). Removing it before the query and key projections sharply increases perplexity in the softmax models, while removing it for the value projection is less detrimental.

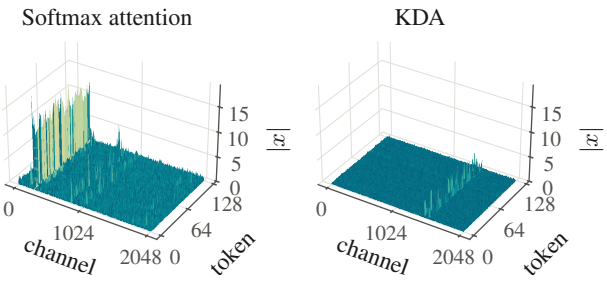


Figure 2: Residual stream of a representative layer.

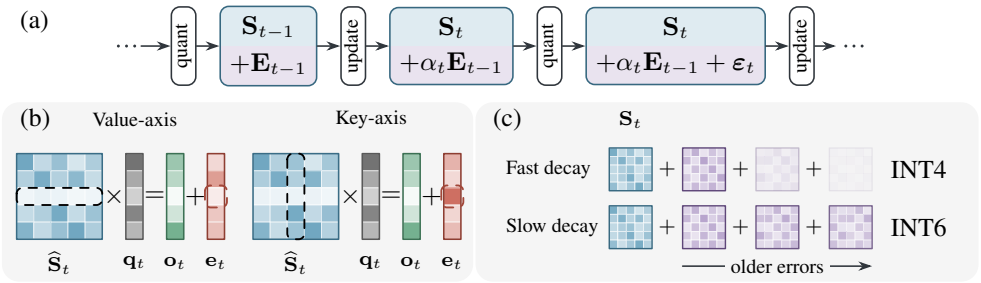


Figure 3: Recurrent-state quantization. (a) Fresh quantization error adds to decayed inherited error. (b) Value- and key-axis grouping impact quantization readout error $\mathbf{e}_t$. Black outlines mark scale groups. (c) Faster decay suppresses older errors more strongly, allowing for lower precision.

Replacing the channel at the mixer input by its mean over tokens is also far less harmful than removing it entirely. This approximately constant component thus provides a useful bias to the queries and keys. Decayed recurrent models are substantially less sensitive to the same intervention. Prior work connects large activation channels to attention sinks and the ability of softmax attention to concentrate probability mass on particular tokens (Sun et al., 2024; 2026; Chen et al., 2026). We hypothesize that decayed recurrences have less need to develop the same residual structure because recurrent decay already provides a mechanism for focusing on a small subset of tokens. We leave further investigation to future work.

Table 3: Removing or replacing the dominant channel by its mean for QK, V, or the whole layer. Reported as relative WikiText-2 perplexity changes (%). The first two rows use softmax attention.

Architectural choices				removed			mean
Decay	$\|\mathbf{QK}\|$	β	Δ	QK	V	QKV	QKV
–	–	–	–	1691.3	7.6	2765.2	7.9
–	✓	–	–	419.3	9.8	389.3	2.7
Diag	✓	✓	✓	2.2	1.1	9.9	1.7

4 QRES: STATE QUANTIZATION

We next study quantization of the recurrent state stored between decode steps, keeping the state update in full precision. Figure 3 illustrates the numerical structure of this problem. We evaluate state quantization on five 1.3B mixer-zoo models and the three hybrids using teacher-forced decode over eight sequences of 2048 WikiText-2 tokens. Our core metric $\Delta\ell$ is the median token-level increase in cross-entropy relative to the FP32 state. Appendix F.1 shows that $\Delta\ell$ correlates with generation accuracy, supporting its use as a proxy loss. Throughout this section, we refer to zoo variants by the architectures they model.

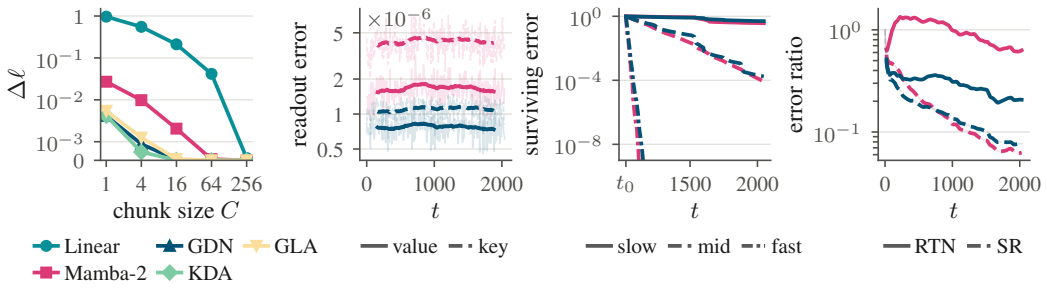
We organize the analysis around three properties that distinguish recurrent state quantization from KV cache quantization. First, the state has two channel axes, whereas a KV cache has one channel axis and one time axis. Section 4.1 addresses how this affects scale grouping (Figure 3(b)). Second, linear recurrences typically employ a decay factor that suppresses quantization error over time (Figure 3(c)). Section 4.2 uses this structure to allocate precision according to decay strength. Third, each KV cache entry is written and quantized once, whereas the recurrent state is dequantized, updated, and requantized after every decode step (Figure 3(a)). Section 4.3 studies how the rounding rule affects this accumulation. The proofs of all mathematical statements are in Appendix C.

For the analysis below, we consider the general recurrence in Equation 1. Let $\mathbf{E}_t$ denote the total state error accumulated up to time t and $\boldsymbol{\varepsilon}_t$ the fresh error introduced when the state is quantized at step t. The accumulated error then contains a propagated component and fresh quantization error,

$$\mathbf{E}_t = \mathbf{E}_{t-1}\mathrm{diag}(\boldsymbol{\alpha}_t)(\mathbf{I} - \delta\,\beta_t\mathbf{k}_t\mathbf{k}_t^\top) + \boldsymbol{\varepsilon}_t. \tag{2}$$

4.1 SCALE GROUPING

We compare two choices for scale grouping of the matrix state $\mathbf{S}_t \in \mathbb{R}^{d_v,d_k}$: value-axis grouping, which assigns one scale to each value row, and key-axis grouping, which assigns one scale to each key column. Both reduce degradation relative to using a single scale for the entire state (Table 11).



(a) chunk size (b) grouping (c) decay (d) rounding

Figure 4: (a) $\Delta\ell$ under INT6 end-of-chunk state quantization with chunk size C. (b) Median readout error $\|\mathbf{E}_t\mathbf{q}_t\|_2^2$ of INT6 quantization under value- and key-axis grouping. (c) Surviving fraction $\|\mathbf{E}_t\|_F^2/\|\boldsymbol{\varepsilon}_{t_0}\|_F^2$ of a single INT4 error injected at $t_0 = 1024$. Slow, mid, and fast denote heads at the 98th, 85th, and 60th percentile of the mean decay. (d) Accumulated-to-injected error ratio $\|\mathbf{E}_t\|_F^2/\sum_{\tau=1}^{t}\|\boldsymbol{\varepsilon}_\tau\|_F^2$ under INT6 quantization with RTN and SR.

Consider the readout error introduced by a single quantization step. For simplicity, assume no inherited error. At b bits, let $Q := 2^{b-1} - 1$ and let group G have quantization scale $s_G := \|\mathbf{S}_G\|_\infty/Q$. We approximate errors as independent and uniformly distributed on $[-s_G/2, s_G/2]$.

Proposition 1 (Per-channel readout error)**.** *Define the readout error* $\mathbf{e} := \mathbf{E}\mathbf{q}$ *and the row and column ranges* $M_i := \max_j |\mathbf{S}_{ij}|$ *and* $C_j := \max_i |\mathbf{S}_{ij}|$*. The expected per-channel squared readout errors for value- and key-axis grouping are, respectively,*

$$\mathbb{E}_{\mathbf{E}}\left[(e_i^{\text{val}})^2\right] = M_i^2/12Q^2, \qquad \mathbb{E}_{\mathbf{E}}\left[(e_i^{\text{key}})^2\right] = \sum_j q_j^2 C_j^2/12Q^2. \tag{3}$$

We use this result to develop a heuristic for choosing the grouping axis. Figure 4(b) shows the corresponding readout error over time. Intuitively, since the state is read out per row, value-axis grouping is the safer default. Indeed, the worst-case error under value-axis grouping is no larger than that under key-axis grouping:

$$\sup_{\|\mathbf{q}\|_2=1} \mathbb{E}_{\mathbf{E}}\left[\|\mathbf{e}^{\text{val}}\|_2^2\right] \le \sup_{\|\mathbf{q}\|_2=1} \mathbb{E}_{\mathbf{E}}\left[\|\mathbf{e}^{\text{key}}\|_2^2\right]. \tag{4}$$

Thus, under the one-step approximation above, value-axis grouping is preferred over key-axis grouping in the worst case. Indeed, it turns out to be the safer default also empirically:

Value grouping as the default. Table 4 compares the two grouping axes. Value-axis grouping is preferred for every model at INT8 and for all models except Linear at INT6. At INT4, errors increase substantially, indicating that the bit budget is insufficient in this regime. This also exposes a limitation of the local approximation above, which does not consider error accumulation from repeated requantization (a phenomenon that is exacerbated in low-bit regimes). We discuss and mitigate this effect in Section 4.3. Based on these results, QReS adopts value-axis grouping by default.

Table 4: Scale grouping under RTN with symmetric grids. Entries are median cross-entropy increases $\Delta\ell$. Bold marks the better axis.

	INT8		INT6		INT4	
model	value	key	value	key	value	key
Linear	**0.125**	0.332	1.230	**0.967**	3.647	**2.479**
Mamba-2	**0.003**	0.011	**0.027**	0.052	0.521	**0.400**
GLA	**0.000**	0.005	**0.005**	0.034	**0.269**	0.309
GDN	**0.000**	0.005	**0.004**	0.025	**0.202**	0.313
KDA	**0.000**	0.007	**0.004**	0.033	0.276	**0.266**
Qwen-3.5	**0.001**	0.010	**0.004**	0.157	**0.825**	1.037
Nemotron-H	**0.001**	0.019	**0.010**	0.183	**0.763**	1.227
Kimi-Linear	**0.001**	0.001	**0.005**	0.009	0.262	**0.056**

Ablations. On Qwen-3.5 and Nemotron-H, uniformly spaced INT8 with value-axis grouping yields lower $\Delta\ell$ than FP8 E4M3 with the same grouping under both rounding rules (Appendix E.1). Hadamard rotations used in KV cache quantization (Ashkboos et al., 2024; Liu et al., 2025) have a mixed effect across models, precisions and rounding rules. We therefore omit them from QReS (see Appendix E.2 for more details). We consider their study an interesting avenue of future work.

4.2 MIXED PRECISION

Decay attenuates inherited error. Unlike static entries in a KV cache, a recurrent state can suppress previously introduced quantization error through decay. The following proposition bounds the accumulated state error using the previously introduced per-step quantization error:

Proposition 2 (Decaying error)**.** *Assume the decay is bounded by* $\|\boldsymbol{\alpha}_t\|_\infty \le \alpha < 1$ *for all* $t \ge 1$*. Then*

$$\|\mathbf{E}_t\|_F \le \sum_{\tau=1}^{t} \alpha^{t-\tau} \|\boldsymbol{\varepsilon}_\tau\|_F.$$

The bound shows that each injected error $\boldsymbol{\varepsilon}_\tau$ decays exponentially over time. Figure 4(c) tracks a single injected error and shows faster suppression in faster-decaying heads.

Table 5: Decay-based mixed precision. INT5 promotes the slowest-decaying half of heads (or channels for GLA, KDA and Kimi). Entries are $\Delta\ell$ under RTN.

model	INT4	INT5	INT6
Mamba-2	0.400	0.033	0.027
GLA	0.269	0.009	0.005
GDN	0.202	0.005	0.004
KDA	0.266	0.005	0.004
Qwen-3.5	0.825	0.006	0.004
Nemotron-H	0.763	0.045	0.010
Kimi-Linear	0.056	0.006	0.005

Decay guides precision allocation. We use the above observation that decay attenuates injected error to assign more precision to components that retain error for longer. We rank heads (or channels for per-channel decayed models) by their average decay during calibration and promote the slowest-decaying half from the best INT4 configuration to the best INT6 configuration (Table 4). Table 5 shows that the resulting INT5 effective precision recovers most of the improvement from INT4 to INT6.

4.3 ROUNDING

The scale grouping analysis in Section 4.1 considers the error introduced by a single quantization step. During decode, however, the state is requantized after every token, so new rounding error interacts with error inherited from earlier steps. Nemotron 3 Super (Chandiramani et al., 2026) uses stochastic rounding (SR) for reduced-precision Mamba state storage. We characterize the effect of SR generally across the recurrence family in Equation 1. Unlike round-to-nearest (RTN), SR is unbiased in expectation (Gupta et al., 2015; Croci et al., 2022). The following proposition formalizes how this changes the accumulation of state error across repeated quantization.

Proposition 3 (Repeated quantization)**.** *The SR and RTN errors satisfy*

$$\mathbb{E}\left[\|\mathbf{E}_t^{\mathrm{SR}}\|_F^2\right] \le \sum_{\tau=1}^{t} \mathbb{E}\left[\|\boldsymbol{\varepsilon}_\tau^{\mathrm{SR}}\|_F^2\right], \quad \|\mathbf{E}_t^{\mathrm{RTN}}\|_F^2 \le t \sum_{\tau=1}^{t} \|\boldsymbol{\varepsilon}_\tau^{\mathrm{RTN}}\|_F^2.$$

If the per-step quantization errors are uniformly bounded, the worst-case squared-error bounds grow linearly in expectation under SR and quadratically under RTN. Both rates are attainable.

Figure 4(d) shows the same effect empirically: relative to the cumulative error injected by quantization, less error survives under SR than survives under RTN.

Rounding and grouping interact strongly. Table 6 shows that the benefit of SR depends on how the state is grouped. For Qwen-3.5 and Nemotron-H, SR improves value-axis grouping but sharply degrades key-axis grouping. Kimi-Linear instead improves under both axes at INT6. Thus, rounding cannot be selected independently of the state geometry studied in Section 4.1. In QReS, we pair SR with value-axis grouping for all three hybrids. Understanding the partial failure of SR under key grouping remains an interesting direction for future work.

Table 6: Median cross-entropy increase $\Delta\ell$ for RTN and SR with symmetric grids. Columns compare value- and key-axis grouping. Bold marks the better rounding rule.

		INT8		INT6	
model	rounding	value	key	value	key
Qwen-3.5	RTN	0.001	**0.010**	0.004	**0.157**
	SR	**0.000**	0.877	**0.002**	7.493
Nemotron-H	RTN	0.001	**0.019**	0.010	**0.183**
	SR	**0.000**	6.513	**0.003**	7.574
Kimi-Linear	RTN	0.001	0.001	0.005	0.009
	SR	**0.000**	**0.000**	**0.001**	**0.001**

Table 7: End-to-end accuracy (%). W/A/S gives weight, activation, and state precision in bits. GPQA and AIME are avg@32.

	Qwen-3.5					Nemotron-H					Kimi-Linear				
W/A/S	GSM8K	MMLU	GPQA	AIME	RULER	GSM8K	MMLU	GPQA	AIME	RULER	GSM8K	MMLU	GPQA	AIME	RULER
16/16/32	95.9	82.6	80.1	82.3	94.6	92.3	70.6	54.1	51.2	88.0	89.1	70.3	63.5	63.6	93.7
16/16/8	96.4	82.1	79.4	79.5	94.7	92.0	69.8	53.1	49.2	88.3	89.2	69.3	62.1	60.8	93.8
8/16/32	96.5	82.5	80.1	83.1	94.6	92.9	70.4	54.0	51.7	88.4	89.8	70.4	63.9	64.3	93.5
8/16/8	96.0	81.8	79.2	80.7	94.5	92.5	70.2	52.9	48.0	88.2	89.6	69.8	63.4	62.6	94.0
8/8/32	96.8	82.4	80.3	82.1	94.6	93.3	70.3	54.8	50.9	88.4	88.9	70.1	63.3	62.3	93.7
8/8/8	96.5	82.8	79.1	80.4	94.7	92.6	69.8	52.9	48.2	88.5	88.4	70.3	62.9	62.9	93.9
4/8/32	97.1	80.7	75.0	73.5	94.9	90.9	67.8	49.0	43.3	85.4	86.4	65.9	53.6	56.9	93.4
4/8/8	96.1	80.4	73.8	70.1	94.8	91.8	66.9	49.1	38.8	85.1	88.9	65.8	54.7	55.2	93.3

5 End-to-End Evaluation

In this section, we combine extra-state W/A quantization (Section 3) with intra-state quantization (Section 4) on the large post-trained hybrid models. Appendix B.3 gives the complete protocol.

Accuracy. We evaluate 8-shot GSM8K (Cobbe et al., 2021), MMLU-Pro (Wang et al., 2024), GPQA (Rein et al., 2023), and AIME (combining the 2024 and 2025 competitions for 60 questions), together with RULER (Hsieh et al., 2024) at a 32k context. Generation is capped at 8k tokens for GSM8K and 64k for MMLU-Pro, GPQA, and AIME.

Following Section 4, states use value-axis grouping and stochastic rounding, with symmetric absmax scales for Qwen-3.5 and asymmetric minmax scales for Nemotron-H and Kimi-Linear (Table 17). Weights use GPTQ (Frantar et al., 2022) with groups of 128 input channels (Appendix B.2). We apply W/A quantization with llm-compressor (AI & vLLM Project, 2024) and serve the resulting models with vLLM (Kwon et al., 2023).

Table 7 compares INT8 and FP32 states across W/A precisions. INT8 states preserve most of the accuracy at each W/A setting, while larger losses generally arise in configurations with 4-bit weights. The additional loss from state quantization does not increase consistently as W/A precision decreases, supporting the combination of extra-state and intra-state quantization.

Next, we examine decode length, which helps interpret the variation across benchmarks. Median response lengths follow AIME > GPQA > MMLU-Pro > GSM8K > RULER (Table 19). GSM8K and MMLU-Pro generally show small changes under state quantization. RULER likewise remains stable when state precision is reduced, despite its 32k-token input, because generation is short. GPQA and AIME tend to show somewhat larger losses, consistent with greater sensitivity over longer decode trajectories. We believe that incorporating quantization during training (e.g. via quantization-aware training or low-precision training) could reduce the remaining degradation, but leave further exploration of this to future work.

We next use decay-based allocation to reduce state precision below INT8, promoting linear attention heads of Qwen-3.5 and Nemotron-H between adjacent precisions based on decay strength. Figure 5 shows that both models support lower effective precision on GSM8K and GPQA. On GSM8K, this permits 5 effective bits for Qwen-3.5 and 5.5 for Nemotron-H with less than a 5% relative drop in accuracy from the FP32-state baseline.

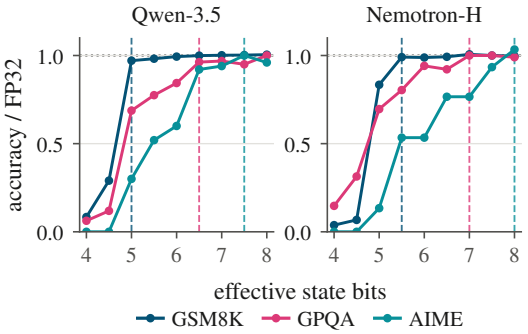
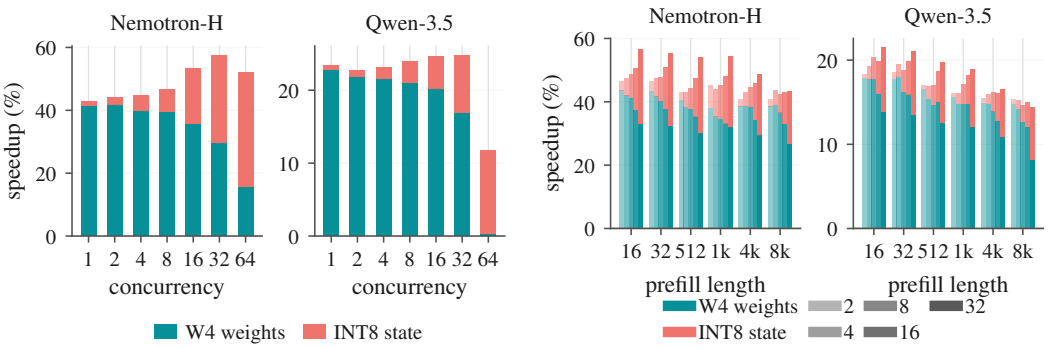


Figure 5: Decay-ranked promotion. Dotted lines mark the minimum bit budget within 5% of FP32 accuracy. GPQA and AIME are pass@1.

Efficiency. We next move to an efficiency analysis of QReS and evaluate how state quantization affects decode speed, both in individual sequence mixer kernels and in end-to-end vLLM serving.



(a) Concurrency sweep.

(b) Prefill-length sweep by concurrency.

Figure 6: Time-per-output-token (TPOT) speedup relative to BF16 weights with an FP32 state. W4 weights form the lower segment and an INT8 state the upper segment. Subfigure (a) varies concurrency at a two-token prefill. Subfigure (b) varies prefill length at several concurrency levels. Appendix G.2 reports contributions separately as well as absolute TPOT.

Figure 7 shows that INT8 state storage accelerates all three sequence mixer kernels as the decode batch size increases. At a batch size of 8, Mamba-2 more than doubles throughput, while GDN and KDA also improve. Appendix G.1 describes the measurement and reports the same experiment on an RTX 6000.

To evaluate the end-to-end efficiency of QReS, we measure fixed-batch decoding of Nemotron-H and Qwen-3.5 with vLLM on one H100, using asynchronous scheduling. We report time per output token (TPOT) after all prefills finish. Appendix G.2 gives the complete measurement protocol. Our main results investigate the 4/16/8 setting. Appendix G reports additional weight and state precisions.

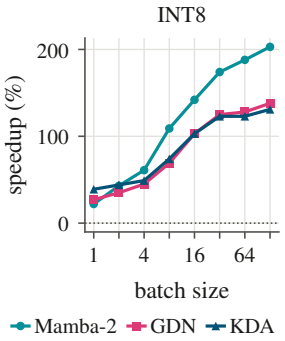


Figure 7: H100 speedup of vLLM mixer kernels using INT8 versus FP32 states.

Figure 6a shows that weight quantization contributes most at low request concurrency, whereas the contribution of INT8 state quantization generally increases as more requests are served concurrently. This is because weight traffic is shared across active sequences, while each sequence carries its own recurrent state. Thus, combining weight and state quantization provides complementary speedups across serving loads. Appendix G.2 reports the contributions separately. Finally, since Qwen and Nemotron are hybrid models, longer prefill grows the KV cache and reduces the relative contribution of the recurrent state to total memory traffic (Figure 6b).

6 Conclusion

In this work, we study PTQ of linear recurrent language models, separating weight/activation quantization from state quantization during decode. First, we focus on extra-state quantization, and in controlled experiments find that decayed recurrences develop weaker residual outliers than softmax attention, making their activations easier to quantize. Next, we characterize the effects of scale grouping, decay, and repeated rounding on state error, yielding QReS, which combines value-axis grouping, stochastic rounding, and decay-based precision allocation. Empirically, QReS supports 8-bit states on Qwen-3.5, Nemotron-H, and Kimi-Linear with modest accuracy loss, and even lower precision at shorter decode horizons. We implement QReS in vLLM and show that combining state and weight quantization improves decode throughput. Future work could explore quantization-aware or low-precision training as well as GPU-native microscaling formats to further improve the performance and efficiency of linear recurrences.

AI USE STATEMENT

In this work, we use generative AI tools to formulate mathematical claims, provide critical ingredients for proving them, to design or give feedback on our methodology and experiments, to create or edit code, implement methods, to propose or refine hypotheses, to support qualitative and thematic data analysis, interpret results, and to do a literature review. Additionally, we used generative AI tools for LaTeX formatting, to draft parts of the appendix, and for brainstorming. We have not used generative AI tools for generating synthetic datasets, to help develop theoretical models or conceptual frameworks, to assist with translation, or to clean or reformat datasets. We have reviewed all AI-assisted work, checked the code as well as the reformatted LaTeX tables. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Red Hat AI and vLLM Project. LLM Compressor, 8 2024. URL https://github.com/vllm-project/llm-compressor.

Alexander Amini, Anna Banaszak, Harold Benoit, Arthur Böök, Tarek Dakhran, Song Duong, Alfred Eng, Fernando Fernandes, Marc Härkönen, Anne Harrington, Ramin Hasani, Saniya Karwa, Yuri Khrustalev, Maxime Labonne, Mathias Lechner, Valentine Lechner, Simon Lee, Zetian Li, Noel Loo, Jacob Marks, Edoardo Mosca, Samuel J. Paech, Paul Pak, Rom N. Parnichkun, Alex Quach, Ryan Rogers, Daniela Rus, Nayan Saxena, Bettina Schlager, Tim Seyde, Jimmy T. H. Smith, Aditya Tadimeti, and Neehal Tumma. Lfm2 technical report, 2025. URL https://arxiv.org/abs/2511.23404.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

Makram Chahine, Philipp Nazari, Daniela Rus, and T Konstantin Rusch. The curious case of in-training compression of state space models. In *International Conference on Learning Representations*, volume 2026, pp. 111759–111798, 2026.

Aakshita Chandiramani, Aaron Blakeman, Abdullahi Olaoye, Abhibha Gupta, Abhilash Somasamudramath, Abhinav Khattar, Adeola Adesoba, Adi Renduchintala, Adil Asif, Aditya Agrawal, et al. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2604.12374*, 2026.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36:4396–4429, 2023.

Yihong Chen, Zhouchen Lin, and Quanming Yao. Attention sinks induce gradient sinks: Massive activations as gradient regulators in transformers. *arXiv preprint arXiv:2603.17771*, 2026.

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. *arXiv preprint arXiv:2503.22879*, 2025a.

Hung-Yueh Chiang, Chi-Chih Chang, Natalia Frumkin, Kai-Chiang Wu, and Diana Marculescu. Quamba: A post-training quantization recipe for selective state space models. In *International Conference on Learning Representations*, volume 2025, pp. 101328–101354, 2025b.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1800–1807. Ieee, 2017.

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

Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3): 211631, 2022.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. Skvq: Sliding-window key and value cache quantization for large language models. *arXiv preprint arXiv:2405.06219*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

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? *arXiv preprint arXiv:2404.06654*, 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.

Hyunwoo Kim, Byoungchan Ko, Minseok Kang, Minwoo Kim, Dongjin Lee, Jaehoon Lee, Sungroh Yoon, and Dahuin Jung. Ssdi8: Accurate and efficient 8-bit quantization for state space duality. In *International Conference on Learning Representations*, volume 2026, pp. 97014–97042, 2026.

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.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Philipp Nazari and T Konstantin Rusch. The key to state reduction in linear attention: A rank-based perspective. *arXiv preprint arXiv:2602.04852*, 2026.

Rom N Parnichkun, Neehal Tumma, Armin W Thomas, Alessandro Moro, Qi An, Taiji Suzuki, Atsushi Yamashita, Michael Poli, and Stefano Massaroli. Quantifying memory utilization with effective state-size. *arXiv preprint arXiv:2504.19561*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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. *arXiv preprint arXiv:2311.12022*, 2023.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Searching for efficient transformers for language modeling. *Advances in neural information processing systems*, 34:6010–6022, 2021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Shangwen Sun, Alfredo Canziani, Yann LeCun, and Jiachen Zhu. The spike, the sparse and the sink: Anatomy of massive activations and attention sinks. *arXiv preprint arXiv:2603.05498*, 2026.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

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 *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10594–10610, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024.

Neehal Tumma, Mathias Lechner, Noel Loo, Ramin Hasani, and Daniela Rus. Leveraging low-rank and sparse recurrent connectivity for robust closed-loop control, 2023. URL https://arxiv.org/abs/2310.03915.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned deltanet: Curvature-aware sequence modeling for linear recurrences. *arXiv preprint arXiv:2604.21100*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Zukang Xu, Yuxuan Yue, Xing Hu, Dawei Yang, Zhihang Yuan, Zixu Jiang, Zhixuan Chen, JiangyongYu, XUCHEN, and Sifan Zhou. Mambaquant: Quantizing the mamba family with variance aligned rotation methods. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=KI45uDnmzv.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

A Extended Related Work

Linear recurrent and hybrid language models. A growing line of work replaces softmax attention with recurrent sequence mixers whose state size and per-token decode cost do not grow with context length. Linear attention uses associativity to express attention as a recurrent update over a fixed-size matrix state (Katharopoulos et al., 2020; Schlag et al., 2021). Selective state-space models such as Mamba and Mamba-2 introduce input-dependent recurrent dynamics and hardware-efficient implementations (Gu & Dao, 2024; Dao & Gu, 2024).

More recent linear recurrences increase the expressivity of the state update. Gated Linear Attention introduces data-dependent forgetting (Yang et al., 2023). DeltaNet uses the delta rule to update stored associations (Schlag et al., 2021; Yang et al., 2024), while Gated DeltaNet combines delta updates with forgetting (Yang et al., 2025). Preconditioned DeltaNet instead introduces curvature-aware preconditioning into DeltaNet-style updates (Tumma et al., 2026). Kimi Delta Attention extends this family with a diagonal decay term and is deployed in Kimi-Linear, which interleaves recurrent and attention layers (Team et al., 2025). Large-scale hybrid models such as Nemotron-H and Qwen-3.5 likewise combine recurrent and softmax attention layers (Blakeman et al., 2025; Qwen Team, 2026). LFM2 is a hardware-aware hybrid architecture that combines gated short convolutions with grouped-query attention for efficient edge inference (Amini et al., 2025).

Post-training quantization of softmax Transformers. Post-training quantization (PTQ) of softmax Transformers has been studied extensively. Weight-only methods such as GPTQ use layer-wise reconstruction to reduce low-bit weight quantization error (Frantar et al., 2022). Joint weight and activation quantization must additionally contend with heavy-tailed activation distributions. SmoothQuant transfers activation scale into the weights (Xiao et al., 2023), while incoherence-based methods apply a change of basis to spread concentrated energy across channels. QuIP and QuIP# apply randomized incoherence processing for low-bit weight quantization (Chee et al., 2023; Tseng et al., 2024). QuaRot extends rotational invariances to weights, activations, and the KV cache (Ashkboos et al., 2024), while SpinQuant learns rotations from calibration data (Liu et al., 2025).

The activation outliers that motivate these methods have also been studied independently of quantization. Sun et al. (2024) identify massive activations that can act as implicit bias terms. More recent work studies their relationship with attention sinks (Sun et al., 2026; Chen et al., 2026). Our extra-state analysis is complementary: instead of introducing another outlier treatment, we ask whether changing the sequence mixer alters the outlier structure learned during training in the first place.

A second line of Transformer PTQ targets the KV cache. KIVI observes different quantization geometry for keys and values and uses per-channel key quantization and per-token value quantization (Liu et al., 2024). KVQuant similarly exploits asymmetric key/value structure together with outlier-aware quantization (Hooper et al., 2024), while SKVQ combines channel reordering with groupwise quantization and a recent-token window retained at higher precision (Duanmu et al., 2024). Rotation-based methods can also transform keys and values before cache quantization (Ashkboos et al., 2024; Liu et al., 2025). These methods operate on cache entries that are written once and subsequently reused, in contrast to a recurrent state that is modified in place after every generated token.

Quantization of recurrent language models. Several recent methods adapt weight and activation PTQ to Mamba-family architectures. Quamba identifies sensitive feature maps inside the selective scan and large output activation outliers, and develops an 8-bit weight and activation quantization recipe around these statistics (Chiang et al., 2025b). MambaQuant likewise targets Mamba-specific activation distributions using variance-aligned rotations (Xu et al., 2025). Quamba2 extends PTQ to Mamba and Mamba-2 across several weight and activation precisions (Chiang et al., 2025a). These works develop effective quantization recipes for particular recurrent architectures. Our extra-state study instead holds the surrounding architecture and quantization procedure fixed and asks how the sequence mixer itself changes the numerical properties of the learned weights and activations.

Regarding state quantization, Q-Mamba directly quantizes Mamba state caches in addition to weights and activations. Its decoupled scale quantization accounts for outliers along multiple state dimensions and supports low-bit state caches (Tianqi et al., 2025). Nemotron 3 Super studies reduced-precision Mamba state storage during autoregressive decode. It evaluates FP16 storage

with stochastic rounding and block-scaled INT16, and shows that stochastic rounding helps control error accumulation across recurrent updates (Chandiramani et al., 2026). These works establish that low-precision state storage is feasible for Mamba-family models.

SSDi8 considers a related but distinct setting. It maintains an INT8 representation through the internal Mamba-2 SSD computation and quantizes intermediate activations inside the chunked SSD algorithm (Kim et al., 2026). Its primary target is therefore low-precision execution of the SSD computation.

Our intra-state study takes a different perspective from these architecture-specific recipes. We treat recurrent state quantization as a numerical problem across several recurrence families and isolate three sources of error: the geometry of scale grouping, accumulation under repeated requantization, and attenuation through recurrent decay. We then use these properties to construct QReS and evaluate its transfer to large hybrid language models.

Recurrent-state compression beyond quantization. Quantization reduces the precision of each stored state, while an orthogonal line of work improves recurrent efficiency by changing the dimensionality or structure of the recurrence. Chahine et al. (2026) use ideas from balanced truncation and Hankel singular values to reduce state dimension during training, while Parnichkun et al. (2025) investigate effective state size in linear operators. For pretrained linear attention models, Nazari & Rusch (2026) identify low-rank recurrent states and structurally prune query and key dimensions after training. Related work has also studied low-rank and sparse recurrent structure outside language modeling (Tumma et al., 2023).

B Experimental Setup

B.1 Models and training

All mixer zoo checkpoints share the same 1.3B-parameter backbone: 24 layers, hidden size 2048, an MLP ratio of 4, and 16 mixer heads of head dimension 128 per layer, both along the key- and the value axis. All sequence mixer layers share the same scaffold, using short convolutions on queries, keys and values (Chollet, 2017; So et al., 2021; Fu et al., 2022) followed by SiLU and RoPE. The output is generated using a gated RMSNorm before an output projection. All models are trained for 100B tokens with the same recipe and a learning rate of 4×10^{-4}.

B.2 Quantization and calibration

For hybrid models, we use GPTQ for weight quantization (Frantar et al., 2022) through llm-compressor (AI & vLLM Project, 2024). We use groups of 128 input channels, except that W8A8 uses per-output-channel weight scales, and calibrate on 128 sequences of 2048 tokens from the WikiText-2 *train* split. Activations are quantized per token. The rotations described in Section 3 are active during calibration, and foldable rotations are absorbed into the weights before quantization. The language-model head remains in BF16. For Kimi-Linear, every expert is calibrated on all calibration tokens, while the router remains in BF16.

Mixer zoo. We use GPTQ with symmetric per-output-channel weight scales and calibrate on 128 sequences of 2048 tokens from the WikiText-2 *train* split. Dynamic activation scales are computed per token, with groups of 128 channels at mixer output-projection inputs. INT4 grids are asymmetric and INT8 grids are symmetric. Decay and write-gate projections use the same W/A precision as the other block linears, and the rotations use Hadamard matrices. The unquantized baseline uses BF16. Embeddings and the language-model head remain in BF16. We evaluate word perplexity on the full WikiText-2 test split with a maximum sequence length of 2048.

B.3 Evaluation protocols

Our generative evaluations follow a common protocol, with model-specific sampling parameters and reasoning modes. Qwen-3.5 uses temperature 0.6, top-p 0.95, top-k 20, following the "precise" thinking-mode preset in its model card (Qwen Team, 2026). Nemotron-H uses temperature 0.6 and top-p 0.95, as recommended in the model card (Blakeman et al., 2025). Kimi-Linear uses

temperature 1.0 and top-p 0.95. While Qwen-3.5 has reasoning enabled by default, we explicitly activate it for all evals but RULER for Nemotron-H.

Generation budgets. GSM8K decodes up to 8192 new tokens. AIME, GPQA (diamond), and MMLU-Pro decode up to 65536. GSM8K uses the 8-shot chain-of-thought prompt.

Answer-format instructions. Our GPQA evaluations use the following NeMo-Skills multiple-choice instruction for all models: "Answer the following multiple choice question. The last line of your response should be in the following format: 'Answer: A/B/C/D' (e.g. 'Answer: A')." The question and four lettered choices follow this instruction. For AIME, all models use the NeMo-Skills math instruction "Solve the following math problem. Make sure to put the answer (and only answer) inside `\boxed{}`." MMLU-Pro uses the corresponding ten-choice instruction with the format line "Answer: A/B/C/D/E/F/G/H/I/J," options listed in dataset order, and no few-shot examples. We omit the math harness's base-model stop strings (`Q:`, `Question:`, and `Problem:`) because we find they can also occur within a reasoning trace.

Scoring. AIME results pool the 2024 and 2025 competitions, each containing 30 problems. AIME responses are graded with the NeMo-Skills math grader: the integer in the last valid `\boxed{}` expression, compared with the reference by Math-Verify. For GSM8K, we select the last boxed answer, then the last explicit answer cue, and otherwise fall back to Math-Verify's mathematical extraction. Numeric answers are compared exactly, without integer rounding. Other expressions use the NeMo-Skills math grader (Blakeman et al., 2025). For GPQA and MMLU-Pro, a rule modeled on the NeMo-Skills multiple-choice grader extracts the choice from the last valid `Answer: X` line and otherwise uses the last `\boxed{X}` expression, where X is A–D for GPQA and A–J for MMLU-Pro. Our implementation also accepts a parenthesized choice after the answer cue, as in `Answer: (C)`. Responses without a valid choice are marked incorrect.

C PROOFS

Proof of Proposition 1. For fixed $\mathbf{S}$ and $\mathbf{q}$, the readout error in component i is $e_i = \sum_j \mathbf{E}_{ij} q_j$. If an entry belongs to a group with scale s_G, the high-resolution approximation gives $\mathbf{E}_{ij} \sim \mathrm{Unif}[-s_G/2, s_G/2]$, so

$$\mathbb{E}_{\mathbf{E}}[\mathbf{E}_{ij}] = 0, \qquad \mathbb{E}_{\mathbf{E}}[\mathbf{E}_{ij}^2] = \frac{s_G^2}{12}.$$

Independence and zero means imply $\mathbb{E}_{\mathbf{E}}[\mathbf{E}_{ij}\mathbf{E}_{ik}] = 0$ for $j \neq k$. Expanding the square therefore yields

$$\mathbb{E}_{\mathbf{E}}[e_i^2] = \sum_j q_j^2 \mathbb{E}_{\mathbf{E}}[\mathbf{E}_{ij}^2] + 2\sum_{j<k} q_j q_k \mathbb{E}_{\mathbf{E}}[\mathbf{E}_{ij}\mathbf{E}_{ik}] = \sum_j q_j^2 \mathbb{E}_{\mathbf{E}}[\mathbf{E}_{ij}^2]. \tag{5}$$

Under value-axis grouping, every entry in row i has scale $s_i^{\mathrm{val}} = M_i/Q$. Substitution into equation 5 gives

$$\mathbb{E}_{\mathbf{E}}\left[(e_i^{\mathrm{val}})^2\right] = \frac{M_i^2}{12Q^2}\sum_j q_j^2 = \frac{M_i^2}{12Q^2},$$

Linear attention uses unit-norm queries, $\|\mathbf{q}\|_2 = 1$, which gives the final equality. Under key-axis grouping, every entry in column j has scale $s_j^{\mathrm{key}} = C_j/Q$, hence

$$\mathbb{E}_{\mathbf{E}}\left[(e_i^{\mathrm{key}})^2\right] = \frac{1}{12Q^2}\sum_j q_j^2 C_j^2.$$

These are the two expressions in equation 3. □

Proof of Equation 4. Summing equation 3 over output components using $\|\mathbf{e}\|_2^2 = \sum_i e_i^2$ gives

$$\mathbb{E}_{\mathbf{E}}\left[\|\mathbf{e}^{\mathrm{val}}\|_2^2\right] = \frac{1}{12Q^2}\sum_i M_i^2$$

and

$$\mathbb{E}_{\mathbf{E}}\big[\|\mathbf{e}^{\text{key}}\|_2^2\big] = \frac{d_v}{12Q^2}\sum_j q_j^2 C_j^2.$$

Let $C_{\max} \coloneqq \max_j C_j = \max_{i,j} |\mathbf{S}_{ij}|$. For a unit-norm query,

$$\sup_{\|\mathbf{q}\|_2=1} \sum_j q_j^2 C_j^2 = C_{\max}^2.$$

Moreover, $M_i \le C_{\max}$ for every i, so

$$\sum_i M_i^2 \le d_v C_{\max}^2.$$

Substitution into the two expected errors proves equation 4. □

Proof of Proposition 2. We assume zero initial state error, $\mathbf{E}_0 = 0$. Because $\|\mathbf{k}_t\|_2 = 1$, the matrix $\mathbf{I} - \delta\,\beta_t \mathbf{k}_t\mathbf{k}_t^\top$ acts as the identity on the orthogonal complement of $\mathbf{k}_t$ and has eigenvalue $1 - \delta\,\beta_t$ along $\mathbf{k}_t$. The ranges $\delta \in \{0, 1\}$ and $\beta_t \in (0, 2)$ therefore give $\|\mathbf{I} - \delta\,\beta_t \mathbf{k}_t\mathbf{k}_t^\top\|_2 \le 1$. Moreover, $\|\mathrm{diag}(\boldsymbol{\alpha}_t)\|_2 = \|\boldsymbol{\alpha}_t\|_\infty \le \alpha$. Applying the triangle inequality and $\|\mathbf{XY}\|_F \le \|\mathbf{X}\|_F\|\mathbf{Y}\|_2$ to equation 2 yields

$$\|\mathbf{E}_t\|_F \le \alpha\|\mathbf{E}_{t-1}\|_F + \|\boldsymbol{\varepsilon}_t\|_F.$$

Iterating from $\mathbf{E}_0 = 0$ proves the claimed bound. If each injection is bounded by ϵ, summing the geometric series gives

$$\|\mathbf{E}_t\|_F \le \epsilon\sum_{r=1}^{t}\alpha^{t-r} = \epsilon\frac{1-\alpha^t}{1-\alpha} \le \frac{\epsilon}{1-\alpha}.$$

□

Proof of Proposition 3. We assume zero initial state error, $\mathbf{E}_0 = 0$. Let $\mathcal{H}_t$ denote the history immediately before rounding at step t, including the update and quantization scales. Under SR, $\mathbb{E}[\boldsymbol{\varepsilon}_t \mid \mathcal{H}_t] = 0$. The propagated error is $\mathcal{H}_t$-measurable, so the cross term vanishes in expectation. Moreover, $\|\mathrm{diag}(\boldsymbol{\alpha}_t)\|_2 \le 1$ and, as above, $\|\mathbf{I} - \delta\,\beta_t \mathbf{k}_t\mathbf{k}_t^\top\|_2 \le 1$. Squaring equation 2 therefore gives

$$\begin{aligned}\mathbb{E}\big[\|\mathbf{E}_t\|_F^2\big] &= \mathbb{E}\Big[\big\|\mathbf{E}_{t-1}\,\mathrm{diag}(\boldsymbol{\alpha}_t)(\mathbf{I} - \delta\,\beta_t\mathbf{k}_t\mathbf{k}_t^\top)\big\|_F^2\Big] + \mathbb{E}\big[\|\boldsymbol{\varepsilon}_t\|_F^2\big] \\ &\le \mathbb{E}\big[\|\mathbf{E}_{t-1}\|_F^2\big] + \mathbb{E}\big[\|\boldsymbol{\varepsilon}_t\|_F^2\big]\,.\end{aligned}$$

Iterating from $\mathbf{E}_0 = 0$ yields the SR bound. For additive updates without decay or delta correction, the inequality is an equality.

For any rounding rule, the triangle inequality and non-expansiveness of the transition give

$$\|\mathbf{E}_t\|_F \le \|\mathbf{E}_{t-1}\|_F + \|\boldsymbol{\varepsilon}_t\|_F \le \sum_{\tau=1}^{t}\|\boldsymbol{\varepsilon}_\tau\|_F.$$

Cauchy–Schwarz then yields

$$\|\mathbf{E}_t\|_F^2 \le \left(\sum_{\tau=1}^{t}\|\boldsymbol{\varepsilon}_\tau\|_F\right)^2 \le t\sum_{\tau=1}^{t}\|\boldsymbol{\varepsilon}_\tau\|_F^2.$$

Uniform boundedness gives a constant $v < \infty$, independent of t, such that $\mathbb{E}[\|\boldsymbol{\varepsilon}_t^{\text{SR}}\|_F^2] \le v$ and $\|\boldsymbol{\varepsilon}_t^{\text{RTN}}\|_F^2 \le v$. The two bounds therefore imply

$$\mathbb{E}\big[\|\mathbf{E}_t^{\text{SR}}\|_F^2\big] \le tv, \qquad \|\mathbf{E}_t^{\text{RTN}}\|_F^2 \le t^2 v.$$

To show that both rates are attainable with varying writes, take an additive recurrence without decay or delta correction, $\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{U}_t$, where $\mathbf{U}_t$ is the additive write at step t. Set $\mathbf{S}_0 = \widehat{\mathbf{S}}_0 = 0$ and use elementwise rounding on a fixed grid $h\mathbb{Z}$, where $h > 0$. Let the $(1, 1)$ entry u_t of $\mathbf{U}_t$ alternate between $h/4$ and $h/3$, with all other entries zero. RTN discards every write, so $\widehat{\mathbf{S}}_t = 0$. Under SR,

the quantized state remains on the grid, and the fresh error in the $(1,1)$ entry is $-u_t$ with probability $1-u_t/h$ and $h-u_t$ with probability u_t/h. Its expected square is therefore $u_t(h-u_t)$. Using the SR equality for additive updates gives

$$\mathbb{E}\big[\|\mathbf{E}_t^{\mathrm{SR}}\|_F^2\big] = \sum_{\tau=1}^{t} u_\tau(h-u_\tau) = \Theta(th^2),$$

$$\|\mathbf{E}_t^{\mathrm{RTN}}\|_F^2 = \left(\sum_{\tau=1}^{t} u_\tau\right)^2 = \Theta(t^2h^2).$$

All fresh errors in this construction have Frobenius norm below h, so they are uniformly bounded. Since h is fixed, this proves attainment of both growth rates. □

D Extra-State Quantization

D.1 Quantization results and rotation ablations

Table 8 reports the full weight–activation precision ladder in absolute WikiText-2 perplexity, including the unquantized baseline. The main text (Table 2) shows relative changes for unrotated W4A8 and W4A4, together with W4A4 rotation ablations.

Table 8: Complete absolute WikiText-2 perplexity ladder. Baseline is the unquantized BF16 model. 16-bit entries denote BF16.

Architectural choices					No rotation					Full stack $\mathbf{R}_1\mathbf{R}_4\mathbf{R}_o$				
Decay	$\|\mathbf{QK}\|$	β	Δ	Baseline	W8A8	W4A16	W4A8	W16A4	W4A4	W8A8	W4A16	W4A8	W16A4	W4A4
–	–	–	–	15.66	15.71	16.52	16.57	26.47	31.07	15.67	16.34	16.34	16.44	17.33
–	✓	–	–	15.66	15.70	16.53	16.58	25.87	30.11	15.67	16.40	16.40	16.46	17.40
–	✓	–	–	17.93	17.98	19.06	19.15	31.72	37.91	17.93	18.84	18.86	18.95	20.16
–	✓	✓	–	18.05	18.10	19.24	19.30	28.54	32.93	18.06	19.01	19.02	19.12	20.30
–	✓	✓	✓	17.72	17.77	18.72	18.80	26.32	29.80	17.72	18.61	18.61	18.73	19.97
Scal	✓	–	–	16.94	16.99	17.93	17.97	23.68	26.53	16.95	17.78	17.80	17.87	18.96
Scal	✓	✓	–	17.19	17.23	18.09	18.15	23.82	26.40	17.19	18.01	18.02	18.07	19.13
Scal	✓	✓	✓	17.10	17.15	18.09	18.12	23.90	26.58	17.12	17.94	17.94	18.03	19.12
Diag	✓	–	–	15.86	15.88	16.58	16.62	20.95	22.99	15.86	16.50	16.50	16.55	17.33
Diag	✓	✓	–	16.08	16.12	16.89	16.95	21.75	23.99	16.09	16.83	16.82	16.83	17.78
Diag	✓	✓	✓	16.55	16.58	17.46	17.51	22.13	24.46	16.55	17.30	17.29	17.42	18.40

Table 9 ablates the full rotation stack $\mathbf{R}_1\mathbf{R}_4\mathbf{R}_o$ on the 1.3B mixer zoo. Among the individual rotations, $\mathbf{R}_1$ is more effective for the softmax models and for plain linear attention, whereas $\mathbf{R}_4$ is more effective for every recurrent cell that carries a decay mechanism, a write gate, or a delta-rule correction. This shows that decayed linear recurrence PTQ error is not dominated by outliers in the residual stream.

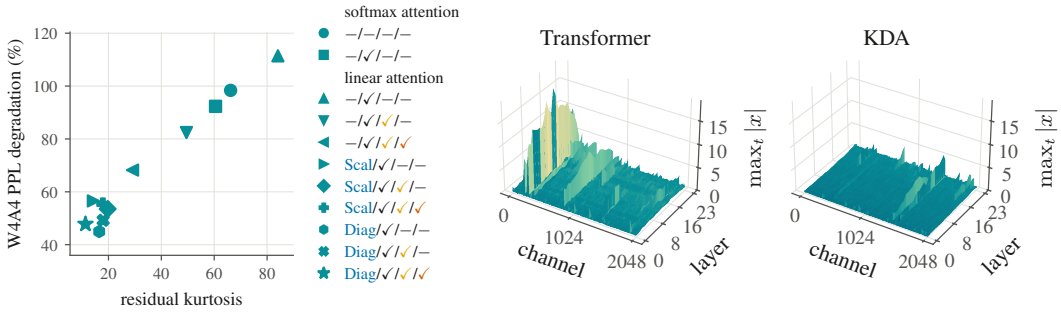
Table 9: W4A4 ablation of all subsets of $\mathbf{R}_1\mathbf{R}_4\mathbf{R}_o$ on the 1.3B mixer zoo. Entries are relative WikiText-2 perplexity increases (%).

Architectural choices				Δ_{PPL} (%)							
Decay	$\|\mathbf{QK}\|$	β	Δ	none	$\mathbf{R}_1$	$\mathbf{R}_4$	$\mathbf{R}_o$	$\mathbf{R}_1\mathbf{R}_4$	$\mathbf{R}_1\mathbf{R}_o$	$\mathbf{R}_4\mathbf{R}_o$	$\mathbf{R}_1\mathbf{R}_4\mathbf{R}_o$
–	–	–	–	98.4	36.7	47.9	90.1	12.2	34.4	44.3	**10.6**
–	✓	–	–	92.3	37.8	46.5	86.0	12.9	35.7	42.8	**11.1**
–	✓	–	–	111.4	48.1	54.0	105.9	14.0	45.5	50.5	**12.5**
–	✓	✓	–	82.4	47.2	35.9	77.9	13.9	44.8	33.2	**12.4**
–	✓	✓	✓	68.2	46.5	28.5	65.0	13.9	43.7	26.3	**12.7**
Scal	✓	–	–	56.6	40.3	24.7	54.8	13.0	38.1	23.3	**11.9**
Scal	✓	✓	–	53.5	41.3	21.3	51.2	12.6	38.8	19.5	**11.3**
Scal	✓	✓	✓	55.4	43.2	21.5	52.8	13.1	40.6	19.8	**11.8**
Diag	✓	–	–	45.0	32.6	18.3	42.8	10.4	31.0	16.7	**9.3**
Diag	✓	✓	–	49.2	35.7	20.7	46.8	11.6	34.4	19.3	**10.6**
Diag	✓	✓	✓	47.8	37.0	19.5	45.6	12.5	34.8	18.1	**11.2**

D.2 Residual outliers and interventions

Figure 8a relates residual-stream kurtosis to unrotated W4A4 degradation across the mixer zoo.

Figure 8b shows that the Transformer builds a consistent outlier channel across layers and channels and complements Figure 2.



(a) Kurtosis versus W4A4 degradation.

(b) Mixer-input magnitude by channel and layer.

Figure 8: Subfigure (a) relates residual-stream kurtosis to unrotated W4A4 perplexity degradation. Legend entries denote decay/$\|\mathbf{QK}\|$/β/Δ, as in Table 2. Subfigure (b) shows the mixer-input magnitude over 128 tokens.

Table 10 extends Table 3 to every cell of the 1.3B mixer zoo. Removing the dominant channel, either at the mixer input or before individual projections, has a much larger effect in the softmax cells than in the recurrent cells.

Table 10: Absolute WikiText-2 perplexity under the dominant-channel interventions of Table 3. *None* is the unmodified BF16 model. Under *removed*, QK/V delete the channel only before the corresponding projections and QKV deletes it at the mixer input. *mean* replaces QKV at the mixer input by its token average.

Architectural choices				none	removed			mean
Decay	$\|\mathbf{QK}\|$	β	Δ		QK	V	QKV	QKV
–	–	–	–	15.66	280.51	16.86	448.70	16.89
–	✓	–	–	15.66	81.30	17.18	76.61	16.08
–	✓	–	–	17.93	21.23	18.20	23.62	19.81
–	✓	✓	–	18.05	18.89	18.21	20.64	18.62
–	✓	✓	✓	17.72	18.42	18.10	21.91	17.96
Scal	✓	–	–	16.94	17.41	17.12	18.21	17.16
Scal	✓	✓	–	17.19	17.76	17.36	19.65	17.38
Scal	✓	✓	✓	17.10	17.52	17.22	18.78	17.22
Diag	✓	–	–	15.86	16.64	15.98	18.81	16.26
Diag	✓	✓	–	16.08	16.85	16.35	19.93	16.27
Diag	✓	✓	✓	16.55	16.91	16.74	18.19	16.83

E State Quantization

E.1 Grouping, mixed precision, and rounding

This subsection reports the additional ablations referenced in Section 4.

Table 11 compares per-axis with per-state scale grouping. Both value- and key-axis grouping outperform per-state grouping by a wide margin.

Table 11: INT6 $\Delta\ell$ under per-state, value-axis, and key-axis scaling.

model	state	value	key
Linear	2.358	1.230	0.967
Mamba-2	0.516	0.027	0.052
GLA	0.377	0.005	0.034
GDN	0.158	0.004	0.025
KDA	0.361	0.004	0.033

Table 12 compares the decay-based INT5 allocation from Section 4.2 with a matched random allocation.

Table 12: Decay-based and random INT5 allocations under the setting of Table 5. Entries are $\Delta\ell$. Both assign INT6 to half of the heads, or key channels for GLA, KDA and Kimi-Linear, and INT4 to the remainder.

model	decay	random
Mamba-2	**0.033**	0.203
GLA	**0.009**	0.073
GDN	**0.005**	0.064
KDA	**0.005**	0.108
Qwen-3.5	**0.006**	0.267
Nemotron-H	**0.045**	0.158
Kimi-Linear	**0.006**	0.028

Integer versus floating-point formats. The integer grids in Section 4 are uniformly spaced. Here, we ablate on using FP8 in E4M3 format instead, giving finer spacing near zero and coarser spacing at large magnitudes. We use the same scale-grouping for both formats.

Table 13 gives the teacher-forced comparison. We find that INT8 with stochastic rounding and value-axis groups stays the best setting. However, E4M3 rescues stochastic rounding on the key axis.

Table 13: INT8 and E4M3 under uniform grouping and symmetric grids. Entries are $\Delta\ell$. INT8 repeats Table 6.

		INT8		E4M3	
model	axis	RTN	SR	RTN	SR
Qwen-3.5	value	0.0005	0.0001	0.0010	0.0002
Qwen-3.5	key	0.0100	0.8767	0.0027	0.0004
Nemotron-H	value	0.0006	0.0001	0.0009	0.0015
Nemotron-H	key	0.0193	6.5132	0.0037	0.0429

E.2 Rotations

KV cache quantization methods such as SpinQuant (Liu et al., 2025) exploit rotational symmetries of the softmax operator to reduce outliers before quantization. While linear attention admits the required symmetries (Nazari & Rusch, 2026), we find that rotations do not always improve state quantization. This is because, while every entry inside a KV cache is just quantized once, the state is dequantized, updated, and requantized repeatedly during decode. A rotation therefore affects both the immediate reconstruction error under quantization and how that error propagates forward.

For value-axis grouping, $\mathbf{R}_k$ rotates the key dimension within each scale group. For key-axis grouping, $\mathbf{R}_v$ rotates the value dimension. Unless stated otherwise, we assume rotations to act within scale groups. Table 14 compares the rotated and unrotated configurations under RTN with symmetric grids.

Table 14: Within-group rotations under RTN with symmetric grids. Entries are $\Delta\ell$. Bold marks the best setting within each precision for each model.

	INT6				INT4			
model	value	$+\mathbf{R}_k$	key	$+\mathbf{R}_v$	value	$+\mathbf{R}_k$	key	$+\mathbf{R}_v$
Linear	1.230	**0.290**	0.967	0.345	3.647	3.469	2.479	**2.177**
Mamba-2	0.027	**0.009**	0.052	0.016	0.521	0.463	0.400	**0.201**
GLA	0.005	**0.000**	0.034	0.008	0.269	1.127	0.309	**0.134**
GDN	0.004	**0.002**	0.025	0.007	**0.202**	0.631	0.313	0.217
KDA	0.004	**0.001**	0.033	0.009	0.276	0.951	0.266	**0.157**
Qwen-3.5	**0.004**	0.008	0.157	0.006	**0.825**	6.796	1.037	2.875
Nemotron-H	0.010	**0.001**	0.183	2.883	0.763	**0.029**	1.227	6.558

Under RTN, within-group rotations generally reduce $\Delta\ell$ at INT6, with two exceptions: $\mathbf{R}_k$ slightly harms Qwen-3.5 and $\mathbf{R}_v$ strongly harms Nemotron-H. At INT4, $\mathbf{R}_v$ helps all key-grouped mixer zoo models but harms both hybrids. Since rotations do not consistently improve performance across models and settings, we omit them from our main framework. Understanding these differences remains a direction for future work.

Table 15 investigates within-group rotations under SR. Rotations generally help key grouping but have mixed effects on value grouping. At INT4, $\mathbf{R}_k$ harms every value-grouped model except Linear and Nemotron-H, whereas $\mathbf{R}_v$ improves every key-grouped model. On the value axis, $\mathbf{R}_k$ harms Qwen-3.5 but improves Nemotron-H at both INT6 and INT4. For both hybrids at these precisions, key grouping with $\mathbf{R}_v$ still performs worse than unrotated value grouping. These results support omitting rotations as a default, while leaving room for model-specific gains. Both rotation tables use

symmetric grids. The QReS configuration for Nemotron-H instead uses an asymmetric grid. These ablations therefore do not establish that every final configuration is optimal without rotations.

Table 15: Grouping and within-group rotations under stochastic rounding with symmetric grids. Entries are $\Delta\ell$. Bold marks the best setting within each precision for each model.

	INT8		INT6				INT4			
model	value	key	value	$+\mathbf{R}_k$	key	$+\mathbf{R}_v$	value	$+\mathbf{R}_k$	key	$+\mathbf{R}_v$
Linear	0.223	**0.135**	1.654	1.116	1.019	**0.531**	4.338	4.331	3.884	**3.226**
Mamba-2	0.001	**0.001**	0.035	0.041	0.011	**0.003**	0.813	1.179	0.342	**0.151**
GLA	**0.000**	0.001	0.002	**0.002**	0.009	0.003	**0.103**	0.473	0.266	0.113
GDN	**0.000**	0.001	0.005	0.011	0.019	**0.003**	**0.391**	1.371	0.650	0.405
KDA	**0.000**	0.000	0.004	**0.003**	0.010	0.004	0.218	0.850	0.328	**0.176**
Qwen-3.5	**0.000**	0.877	**0.002**	0.024	7.493	0.120	**2.632**	5.474	8.818	7.521
Nemotron-H	**0.000**	6.513	0.003	**0.001**	7.574	7.222	0.520	**0.193**	9.453	8.903

F End-to-End Evaluation

This section extends the decode results presented in Section 5.

F.1 Configuration comparisons and promotion sweeps

Table 16 compares value- and key-axis grouping under stochastic rounding.

Table 16: GSM8K accuracy (%) under SR. FP32 is the unquantized-state baseline.

		value			key		
model	FP32	INT8	INT6	INT4	INT8	INT6	INT4
Qwen-3.5	95.9	96.4	95.3	8.0	47.8	1.5	0.7
Nemotron-H	92.3	92.0	91.2	3.5	0.9	1.4	1.1

Table 17 compares RTN vs SR and grid choice at INT8 state on AIME. Stochastic rounding outperforms round-to-nearest for all three models.

Table 17: AIME avg@32 accuracy (%) at INT8 state. Sym and asym denote symmetric absmax and asymmetric minmax grids.

		Nemotron-H	Qwen-3.5	Kimi-Linear	
rounding	grid	value axis	value axis	value axis	key axis
RTN	sym	40.3	54.0	56.1	53.9
RTN	asym	24.5	57.4	45.5	34.7
SR	sym	48.0	79.5	60.9	60.5
SR	asym	49.2	67.0	60.8	60.2
FP32	–	51.2	82.3	63.6	63.6

Proxy-to-generation transfer. For sequence n and token t, let $\ell^{(n,t)}_{\text{quant}}$ and $\ell^{(n,t)}_{\text{FP}}$ denote the token-level cross-entropies obtained with quantized and FP32 states, respectively. We define

$$\Delta\ell := \underset{n,t}{\text{median}}\big(\ell^{(n,t)}_{\text{quant}} - \ell^{(n,t)}_{\text{FP}}\big),$$

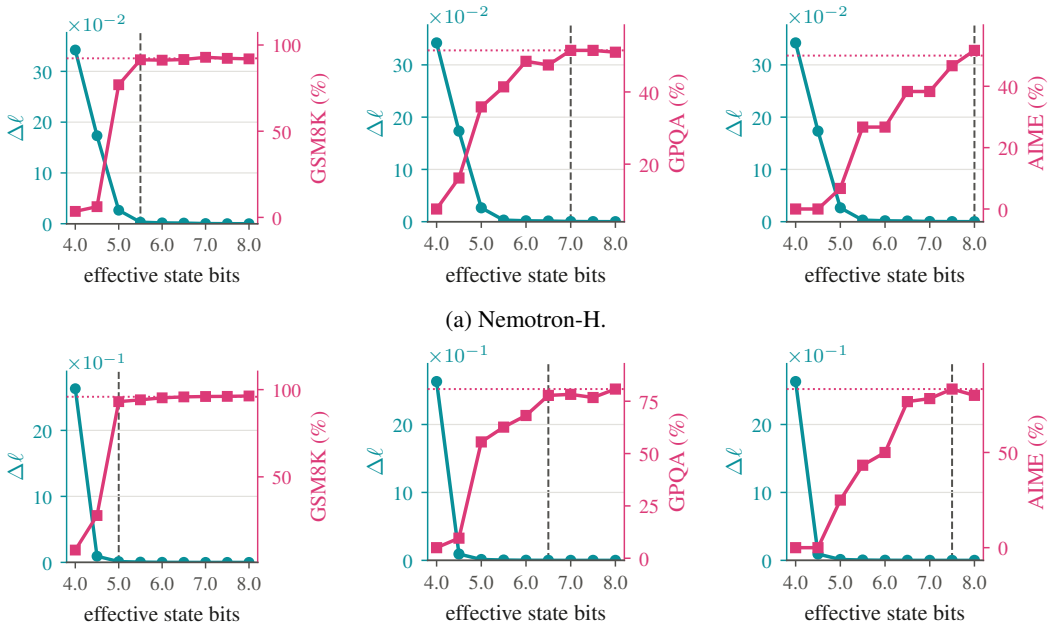
where the median is taken over all sequence–token pairs. Figure 9 compares the teacher-forced proxy $\Delta\ell$ with full-test GSM8K accuracy for four INT6 configurations under round-to-nearest with symmetric grids. The two metrics follow the same overall trend.



(a) Nemotron-H. (b) Qwen-3.5.

Figure 9: Teacher-forced $\Delta\ell$ versus full-test GSM8K accuracy for four INT6 state configurations under round-to-nearest with symmetric grids. The horizontal axis is reversed so that better results lie to the right.

Calibration-to-generation transfer under promotion. Figure 10 compares the teacher-forced proxy $\Delta\ell$ with generation accuracy as decay-ranked heads are promoted.



(a) Nemotron-H.

(b) Qwen-3.5.

Figure 10: Promotion sweeps on GSM8K, GPQA, and AIME. Teal denotes $\Delta\ell$, magenta accuracy, dotted lines the FP32-state baseline, and dashed lines the minimum precision under the 5% criterion.

Table 18 gives the complete accuracy sweeps. Its entries are exactly the accuracy points plotted in Figure 10 and Figure 5.

F.2 Generation Length

Table 19 shows that state INT8 can either lengthen or shorten the median response, depending on the model and benchmark. The accuracy results are therefore not an artifact of systematically shorter quantized generations. The AIME comparison below examines stochastic and nearest rounding.

Table 18: Generation accuracy (%) as a function of effective bits. GPQA and AIME entries are pass@1.

	Qwen-3.5			Nemotron-H		
effective bits	GSM8K	GPQA	AIME	GSM8K	GPQA	AIME
4.0	8.0	5.1	0.0	3.5	7.6	0.0
4.5	27.8	9.6	0.0	6.2	16.2	0.0
5.0	93.1	55.6	25.0	77.0	35.9	6.7
5.5	94.2	62.6	43.3	91.4	41.4	26.7
6.0	95.3	68.2	50.0	91.2	48.5	26.7
6.5	95.8	77.8	76.7	91.6	47.5	38.3
7.0	96.1	78.3	78.3	92.9	51.5	38.3
7.5	96.1	76.8	83.3	92.3	51.5	46.7
8.0	96.4	80.8	80.0	92.0	51.0	51.7
FP32	95.9	80.8	83.3	92.3	51.5	50.0

Table 19: Median response length in generated tokens for the FP32-state baseline and state INT8 at W/A $= 16/16$. Δ is relative to FP32 state.

	Qwen-3.5			Nemotron-H			Kimi-Linear		
benchmark	FP32	INT8	Δ (%)	FP32	INT8	Δ (%)	FP32	INT8	Δ (%)
GSM8K	2096	2212	+5.5	592	588	−0.7	89	88	−1.1
MMLU-Pro	4050	4036	−0.3	1112	1128	+1.4	2080	2054	−1.2
GPQA	9169	9302	+1.5	5449	5439	−0.2	7249	6949	−4.1
AIME	25223	25784	+2.2	10994	10846	−1.3	11839	11359	−4.1
RULER	16	16	0.0	21	20	−4.8	12	12	0.0

Table 20 reports the generation lengths of the AIME runs behind Table 17. For Qwen-3.5 and Nemotron-H, stochastic rounding on the selected grid keeps mean response length close to the FP32-state baseline, whereas round-to-nearest lengthens responses and increases budget exhaustion. Kimi-Linear instead produces shorter responses under round-to-nearest with value-axis grouping, so budget exhaustion does not explain its accuracy loss. For all three models, the selected stochastic-rounding grid yields a smaller absolute change in mean length than round-to-nearest on the same grid.

Table 20: Mean generated AIME tokens (and traces hitting the 65536-token budget) for the arms of Table 17. Same generations as the reported accuracies. n=1920 per cell (60 problems × 32 samples).

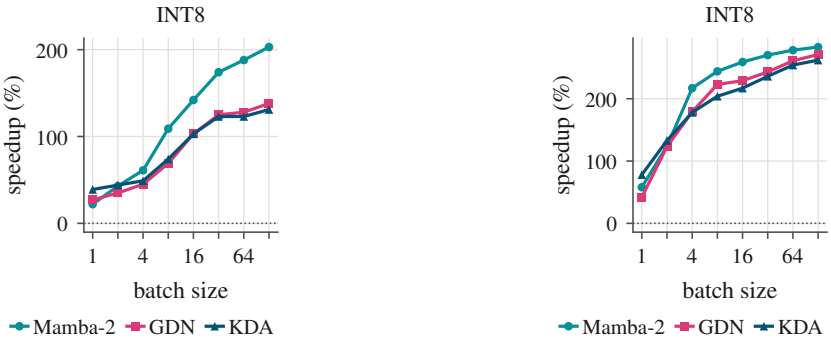
		Qwen-3.5		Nemotron-H		Kimi-Linear (value)		Kimi-Linear (key)	
rounding	grid	tokens	capped	tokens	capped	tokens	capped	tokens	capped
FP32 state		29927	240	12106	26	22565	318	22565	318
RTN	sym	46437	792	30136	459	21398	244	25728	322
RTN	asym	31571	182	18307	187	17760	94	34416	818
SR	sym	31022	277	12337	28	24096	381	22507	365
SR	asym	36985	560	12170	26	23114	364	22144	334

G Implementation and Efficiency

G.1 Kernel Throughput

The state-update kernels tile each head's state $\mathbf{S} \in \mathbb{R}^{d_v, d_k}$ over value rows. Value-axis scales are local to each tile, so quantization and dequantization fuse into the decode kernel. Key-axis scales are not local to these tiles, complicating an efficient implementation. We benchmark vLLM's FP32-

state decode kernels against our INT8-state variants with equal state element counts: GDN and KDA use 32 heads with $d_k = d_v = 128$, while Mamba-2 uses 64 heads with $d_k = 128$ and $d_v = 64$. The INT8 variants use stochastic rounding. A CUDA graph cycles through a working set larger than the L2 cache, and each kernel is launch-tuned at every concurrency. The selected configurations are then timed in seven repetitions, and speedup is the ratio of median kernel times. Figure 11 compares the H100 with the lower-bandwidth RTX 6000. The larger gains on the RTX 6000 are consistent with state traffic accounting for a larger share of decode cost.



(a) H100.

(b) RTX 6000.

Figure 11: Speedup of vLLM mixer kernels using INT8 versus FP32 states.

G.2 End-to-end throughput

We time Nemotron-H-8B-Reasoning-128K and Qwen-3.5 9B with vLLM on a dedicated H100 SXM5, using CUDA graphs and asynchronous decode scheduling. The weight settings are the vendor BF16 checkpoint and rotated, weight-only GPTQ at W4 with group size 128. Activations remain in BF16. Online rotations remain active for W4 during timing and use Dao-AILab's fast CUDA Hadamard kernels.[1] We compare FP32 recurrent states with INT8 states using stochastic rounding and value-axis grouping, with asymmetric quantization for Nemotron-H and symmetric quantization for Qwen-3.5.

Both sweeps use fixed batches of c requests with synthetic token prompts and greedy decoding, forcing 32 output tokens per request. All prefills and the first output token finish before timing starts. We then measure engine wall time for the remaining 31 decode steps, synchronizing the GPU at both timing boundaries, and divide by 31 to obtain TPOT. Prefill, tokenization and HTTP overhead are excluded. Prefix caching is disabled, GPU-memory utilization is set to 0.9, and the engine limits are 64 active sequences and 8192 batched tokens per step.

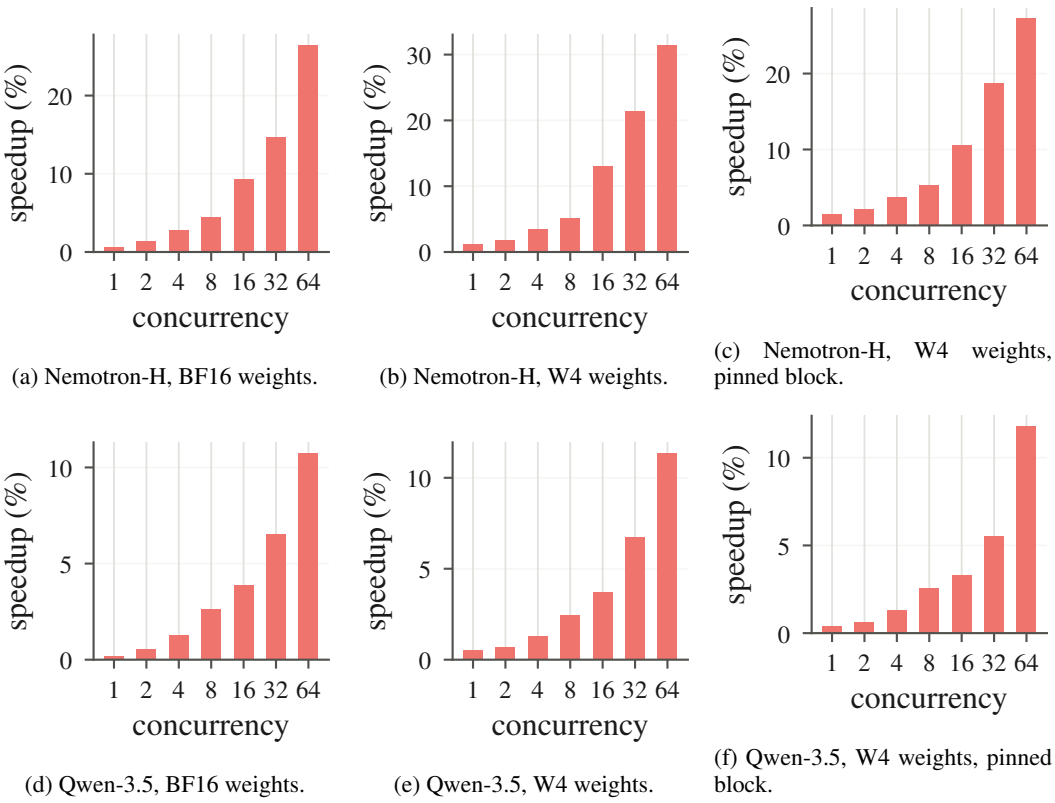
The concurrency sweep uses $c \in \{1, 2, 4, 8, 16, 32, 64\}$ and two-token prompts. The context-length sweep uses $c \in \{2, 4, 8, 16, 32\}$ and prompt lengths $\{16, 32, 512, 1024, 4096\}$, plus 8160 tokens for Nemotron-H or 8192 for Qwen-3.5. Each cell has two warmup batches followed by 16 measured batches per repetition. FP32 and INT8 states are measured in seven paired repetitions on the same GPU, with fresh engine processes, matching prompts and seeds, and alternating measurement order. Reported TPOT is the median of the seven per-repetition means. Speedup is the ratio of the corresponding median TPOTs.

Launch configurations are tuned separately for the FP32 and INT8 decode kernels at each batch size. Attention block sizes are selected automatically, except in the W4 pinned-block control, which fixes both state formats to the FP32-state block size: 1040 tokens for Nemotron-H and 528 for Qwen-3.5.

Figure 12 visualizes INT8-state speedup over an FP32 state at the same weight setting. Table 21 gives the corresponding TPOT for the automatic-block sweeps.

Figure 13 and Table 22 report the context sweep.

[1] https://github.com/Dao-AILab/fast-hadamard-transform



(a) Nemotron-H, BF16 weights.

(b) Nemotron-H, W4 weights.

(c) Nemotron-H, W4 weights, pinned block.

(d) Qwen-3.5, BF16 weights.

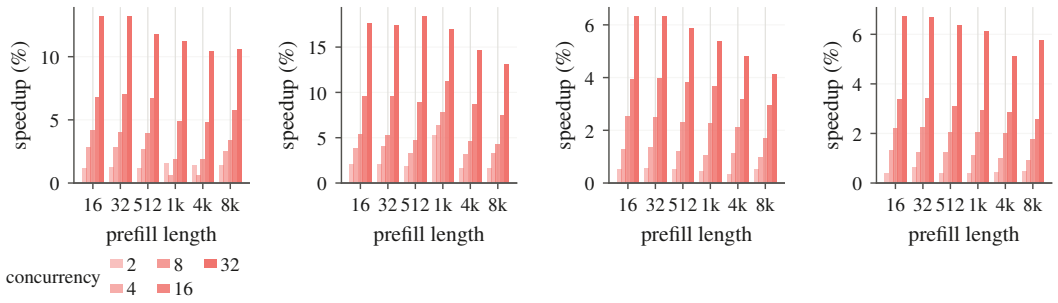
(e) Qwen-3.5, W4 weights.

(f) Qwen-3.5, W4 weights, pinned block.

Figure 12: INT8-state TPOT speedup over an FP32 state, by concurrency. Top row Nemotron-H, bottom row Qwen-3.5. Columns BF16 weights, W4 weights, and W4 weights with the attention block size pinned to its FP32-state value.

Table 21: Absolute TPOT (ms), reported as the median of seven repetitions.

W/A/S	concurrency						
	1	2	4	8	16	32	64
Nemotron-H							
16/16/32	6.31	6.41	6.62	6.96	7.63	8.78	11.40
16/16/8	6.27	6.32	6.44	6.67	6.98	7.66	9.01
4/16/32	4.46	4.52	4.73	4.99	5.62	6.77	9.86
4/16/8	4.41	4.44	4.57	4.74	4.98	5.57	7.50
Qwen-3.5							
16/16/32	6.99	7.05	7.15	7.39	7.77	8.44	9.60
16/16/8	6.98	7.01	7.06	7.20	7.48	7.92	8.67
4/16/32	5.70	5.79	5.89	6.11	6.46	7.22	9.57
4/16/8	5.67	5.74	5.81	5.96	6.23	6.76	8.59



(a) Nemotron-H, BF16 weights. (b) Nemotron-H, W4 weights. (c) Qwen-3.5, BF16 weights. (d) Qwen-3.5, W4 weights.

Figure 13: INT8-state TPOT speedup over the same model and weight setting with an FP32 state, across prefill length. Bar shade gives the concurrency.

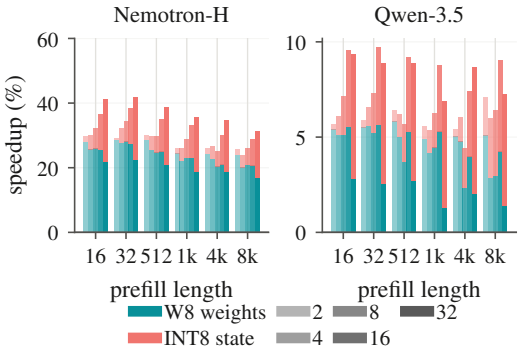


Figure 14: W8 timing without online rotations. TPOT speedup over BF16 weights with an FP32 state across prefill length. Teal denotes the W8-weight contribution, salmon the additional INT8-state contribution, and shade concurrency. Exported weights remain rotated.

Table 22: Absolute TPOT (ms, median of seven repetitions) as prefill length increases. The 8k column uses the longest benchmarked prompt: 8160 tokens for Nemotron-H and 8192 for Qwen-3.5.

W/A/S	concurrency	prefill length					
		16	32	512	1024	4096	8k
Nemotron-H							
16/16/32	2	6.4	6.4	6.5	6.4	6.3	6.4
	4	6.6	6.6	6.7	6.6	6.7	6.8
	8	7.0	7.0	7.1	6.9	7.2	7.4
	16	7.7	7.7	7.8	7.8	8.1	8.5
	32	8.9	9.0	9.0	9.1	9.7	10.5
16/16/8	2	6.3	6.3	6.4	6.3	6.2	6.3
	4	6.5	6.5	6.5	6.6	6.7	6.7
	8	6.7	6.7	6.8	6.8	7.0	7.1
	16	7.2	7.2	7.3	7.4	7.7	8.1
	32	7.9	7.9	8.1	8.2	8.8	9.5
4/16/32	2	4.5	4.5	4.6	4.6	4.6	4.6
	4	4.7	4.7	4.8	4.9	4.8	4.9
	8	4.9	5.0	5.1	5.2	5.2	5.4
	16	5.6	5.6	5.8	5.8	6.0	6.4
	32	6.7	6.8	6.9	6.9	7.5	8.3
4/16/8	2	4.4	4.4	4.5	4.4	4.5	4.5
	4	4.5	4.5	4.7	4.6	4.7	4.8
	8	4.7	4.7	4.9	4.8	5.0	5.2
	16	5.1	5.1	5.3	5.2	5.5	6.0
	32	5.7	5.8	5.9	5.9	6.6	7.4
Qwen-3.5							
16/16/32	2	6.8	6.8	6.8	6.9	7.0	7.1
	4	6.9	6.9	7.0	7.1	7.2	7.5
	8	7.2	7.2	7.3	7.4	7.7	8.1
	16	7.6	7.6	7.8	7.9	8.5	9.2
	32	8.2	8.3	8.5	8.7	9.9	11.4
16/16/8	2	6.8	6.8	6.8	6.9	7.0	7.1
	4	6.8	6.8	7.0	7.0	7.2	7.4
	8	7.0	7.0	7.2	7.2	7.6	7.9
	16	7.3	7.3	7.5	7.6	8.2	9.0
	32	7.7	7.8	8.0	8.3	9.4	11.0
4/16/32	2	5.8	5.8	5.9	6.0	6.1	6.2
	4	5.9	5.9	6.1	6.2	6.3	6.5
	8	6.1	6.2	6.4	6.4	6.8	7.2
	16	6.5	6.5	6.7	6.9	7.5	8.3
	32	7.2	7.3	7.6	7.8	8.9	10.6
4/16/8	2	5.7	5.7	5.9	6.0	6.1	6.2
	4	5.8	5.8	6.0	6.1	6.2	6.5
	8	6.0	6.0	6.3	6.3	6.6	7.0
	16	6.3	6.3	6.5	6.7	7.3	8.0
	32	6.8	6.8	7.1	7.4	8.5	10.0