PRESERVING UPDATES IN LOW-BIT RECURRENT STATE QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear attention and recurrent models compress past tokens into a fixed-size state that is updated at every step. However, storing these states in high precision incurs substantial memory and bandwidth costs at large batch sizes, motivating low-bit state quantization. Unlike weights and key–value caches, a recurrent state is requantized after every update, so its quantization errors accumulate across steps. Our analysis shows that these errors are highly coherent across steps, so reducing per-step errors alone is insufficient. This coherence arises mainly from small updates that are lost in rounding. To preserve these updates, we intervene both before and at rounding. Before rounding, we propose Low-Rank Bypass (LRB), which accumulates small updates in a compact low-rank buffer and periodically merges its sufficiently large elements into the state. At rounding, we observe that a state element can even move opposite to its update, and such reversals prove particularly harmful. Therefore, we propose Bounded Directional Rounding (BDR), which rounds reversed elements toward their updates without additional storage. Although BDR enlarges the rounding error, it improves the model, suggesting that the direction of an update matters more than its rounding error at a single step. Across Gated DeltaNet, GLA, Mamba2, and RetNet, our training-free method substantially outperforms INT4, while reducing state memory by up to 5.3× and speeding up decoding by up to 1.9× relative to FP32 states.

1 INTRODUCTION

Language modeling requires storing contextual information and retrieving relevant associations as new tokens arrive. Transformers provide this capability through softmax attention, but they keep the keys and values of all previous tokens, so their memory and computation grow with the context length (Vaswani et al., 2017). To avoid this growth, linear attention and related recurrent models compress past information into a fixed-size state, so that computation grows linearly and memory stays constant with the context length (Katharopoulos et al., 2020; Schlag et al., 2021; Dao & Gu, 2024). Owing to this efficiency, these architectures have been adopted in recent large language models (Chandiramani et al., 2026).

However, since the fixed-size state accumulates the updates of all previous tokens, it is commonly stored in FP32 to preserve their precision (Chandiramani et al., 2026). With such a state maintained for each sequence in every recurrent layer, the total state memory grows linearly with the batch size and limits the number of concurrent sequences (Tianqi et al., 2025). For example, at a batch size of 128, the recurrent states of a 1.3B Mamba2 model (Dao & Gu, 2024) occupy about 12 GiB. Moreover, every decoding step reads and writes the entire state, so the state accounts for a large share of memory traffic and limits decoding throughput at large batch sizes.

A natural way to reduce these costs is to store the states in lower precision (Tianqi et al., 2025; Chandiramani et al., 2026). Quantization has been widely studied for weights, activations, and key–value caches. Existing methods suppress outliers through scaling, rotation, or per-channel quantization (Xiao et al., 2023; Lin et al., 2024; Shao et al., 2024; Ashkboos et al., 2024; Liu et al., 2024; Hooper et al., 2024), or minimize the layer-wise reconstruction error with second-order information (Frantar et al., 2022). Since these tensors are quantized only once, these methods aim to reduce the error of a single quantization. A recurrent state, however, is requantized after every update, so its accuracy depends on how the errors of all steps accumulate.

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.

We analyze this accumulation and find that the errors of different steps are highly coherent (Sec. 2). Instead of cancelling out like random noise, they add up across steps. Therefore, reducing the error of each quantization alone is insufficient. For example, a Hadamard rotation suppresses outliers to reduce the per-step error, but the perplexity at long contexts remains far above that of FP32 states (Sec. 4). To find the source of this coherence, we examine how rounding treats a single update. The quantization grid is fitted to a state that accumulates many past updates, so its spacing is much larger than a single update. As a result, rounding often leaves the state unchanged, and the update is lost. We further find that the per-step errors consist largely of these lost updates, which makes them coherent across steps.

To keep small updates from being lost, we intervene both before and at rounding. Before rounding, we propose Low-Rank Bypass (LRB), which accumulates small updates in a bypass outside the quantized state and periodically merges its sufficiently large elements into the state. LRB exploits the outer-product structure of each update to keep the bypass in low-rank form, which adds little extra storage. At rounding, we observe that a state element can even move opposite to its update, and such reversals turn out to be particularly harmful to the model. We therefore propose Bounded Directional Rounding (BDR), which rounds a reversed element toward its update without additional storage. Counterintuitively, BDR increases the rounding error but still improves the model, suggesting that the direction of an update matters more than its rounding error at a single step. Our contributions are summarized as follows.

- An analysis of error accumulation in low-bit recurrent states. We show that the per-step errors add up coherently across steps, so reducing the error of each quantization is insufficient, and we trace this coherence to small updates lost in rounding.
- A training-free state quantization method, in which LRB preserves the magnitude of small updates before rounding and BDR preserves their direction at rounding. We demonstrate the method on Gated DeltaNet (Yang et al., 2025), GLA (Yang et al., 2024), Mamba2 (Dao & Gu, 2024), and RetNet (Sun et al., 2023). It substantially improves model quality over INT4, while reducing state memory by 4.7–5.3× and speeding up decoding by up to 1.9× over FP32 states.

2 Error Accumulation in Quantized Recurrent States

2.1 Quantized Recurrent States

Linear attention and recurrent models compress past key–value associations into a fixed-size state $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$, keeping state storage independent of context length (Katharopoulos et al., 2020; Schlag et al., 2021). At each step, a key $\mathbf{k}_t \in \mathbb{R}^{d_k}$ and a value $\mathbf{v}_t \in \mathbb{R}^{d_v}$ update the state as $\mathbf{S}_t = \mathbf{A}_t\mathbf{S}_{t-1} + \mathbf{B}_t$, and a query $\mathbf{q}_t \in \mathbb{R}^{d_k}$ reads it as $\mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t$. Here, $\mathbf{A}_t$ controls the decay of historical information, and $\mathbf{B}_t$ is the update written by the current token. Table 1 lists both terms for the models we study. Specifically, the decay is a scalar multiple of the identity in RetNet (Sun et al., 2023), Mamba2 (Dao & Gu, 2024), and Gated DeltaNet (GDN) (Yang et al., 2025), and a diagonal matrix in GLA (Yang et al., 2024). Moreover, every update in Table 1 is an outer product of the key and a written value.

Model	Decay $\mathbf{A}_t$	Update $\mathbf{B}_t$
RetNet	$\gamma\mathbf{I}$	$\mathbf{k}_t\mathbf{v}_t^\top$
Mamba2	$\alpha_t\mathbf{I}$	$\mathbf{k}_t\mathbf{v}_t^\top$
GLA	$\mathrm{Diag}(\boldsymbol{\alpha}_t)$	$\mathbf{k}_t\mathbf{v}_t^\top$
GDN	$\alpha_t\mathbf{I}$	$\beta_t\mathbf{k}_t\big(\mathbf{v}_t - (\mathbf{A}_t\mathbf{S}_{t-1})^\top\mathbf{k}_t\big)^\top$

Table 1: Decay and update terms of the state recurrence.

Executing this recurrence requires reading and writing the entire state at every decoding step. Consequently, both storage and memory traffic grow with the batch size, which motivates low-precision state storage (Tianqi et al., 2025). A common remedy is to compute updates in full precision while storing the updated state in low precision (Chandiramani et al., 2026). We follow this setting and use group-wise asymmetric quantization (Nagel et al., 2021), which rounds each element to the nearest point of a uniform grid. The grid spacing Δ_t is determined by the range of each group and is refitted at every step (details in App. A.1). The quantized state $\widehat{\mathbf{S}}_t$ then evolves as

$$\begin{aligned}\widehat{\mathbf{S}}_t &= \mathcal{Q}_t\big(\mathbf{A}_t\widehat{\mathbf{S}}_{t-1} + \mathbf{B}_t\big) = \mathbf{A}_t\widehat{\mathbf{S}}_{t-1} + \widetilde{\mathbf{B}}_t,\\ \mathbf{R}_t &= \mathcal{Q}_t\big(\mathbf{A}_t\widehat{\mathbf{S}}_{t-1} + \mathbf{B}_t\big) - \big(\mathbf{A}_t\widehat{\mathbf{S}}_{t-1} + \mathbf{B}_t\big) = \widetilde{\mathbf{B}}_t - \mathbf{B}_t,\end{aligned} \tag{1}$$

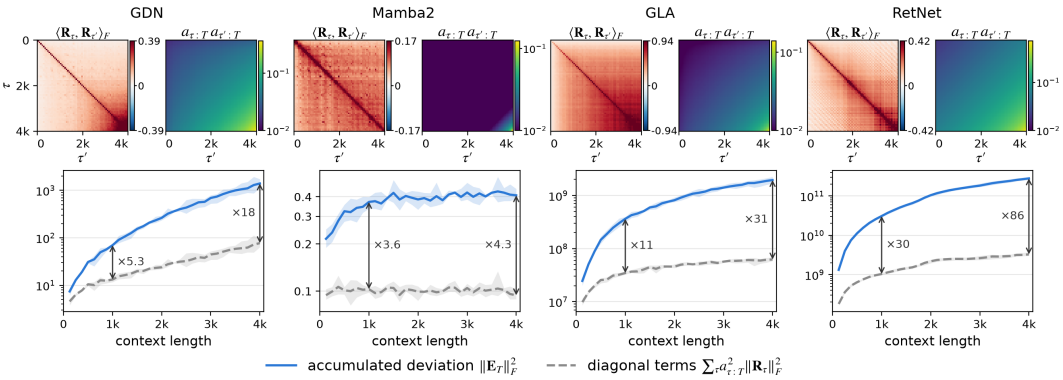

Figure 1: Accumulation of per-step errors under 4-bit state quantization. Top: inner products $\langle \mathbf{R}_\tau, \mathbf{R}_{\tau'} \rangle_F$ (left), summed over 64-step blocks and normalized by the mean diagonal entry for each layer and head before averaging, and decay weights $a_{\tau:T}\, a_{\tau':T}$ at $T = 4$K (right). Bottom: $\|\mathbf{E}_T\|_F^2$ and its diagonal terms as the context grows, with their ratio g_T marked by arrows.

where $\widetilde{\mathbf{B}}_t$ is the effective update written into the quantized state, and $\mathbf{R}_t$ is the per-step error introduced by rounding. As a result, quantization replaces each intended update $\mathbf{B}_t$ with $\widetilde{\mathbf{B}}_t$. Moreover, the per-step error $\mathbf{R}_t$ is carried into all subsequent states and thus affects later outputs.

2.2 Coherent Accumulation of Per-Step Errors

Quantization methods for weights, activations, and key–value caches reduce the error of each quantization independently (Frantar et al., 2022; Lin et al., 2024; Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2024; Hooper et al., 2024). This suffices because each of these tensors is quantized only once. In contrast, the recurrent state is updated and quantized again at every step, so its per-step errors $\mathbf{R}_t$ accumulate. As a result, the degradation grows with context length. For example, under 4-bit quantization, the WikiText-103 (Merity et al., 2016) perplexity of RetNet rises from 59.2 at 1K tokens to 1193.2 at 8K, whereas that of FP32 states stays below 26 (Table 2). To understand this growth, we unroll both recurrences with the same updates $\mathbf{B}_\tau$. With $\mathbf{A}_{\tau:T} = \prod_{j=\tau+1}^{T} \mathbf{A}_j$ denoting the decay from step τ to step T, the accumulated deviation of the quantized state from the full-precision state is

$$\mathbf{E}_T = \widehat{\mathbf{S}}_T - \mathbf{S}_T = \sum_{\tau \le T} \mathbf{A}_{\tau:T} \widetilde{\mathbf{B}}_\tau - \sum_{\tau \le T} \mathbf{A}_{\tau:T} \mathbf{B}_\tau = \sum_{\tau \le T} \mathbf{A}_{\tau:T} \mathbf{R}_\tau. \tag{2}$$

For RetNet, Mamba2, and Gated DeltaNet, whose decay is scalar with $\mathbf{A}_{\tau:T} = a_{\tau:T}\mathbf{I}$, the squared Frobenius norm of this deviation expands as

$$\|\mathbf{E}_T\|_F^2 = \sum_{\tau \le T} a_{\tau:T}^2 \, \|\mathbf{R}_\tau\|_F^2 + \sum_{\tau \ne \tau'} a_{\tau:T}\, a_{\tau':T} \, \langle \mathbf{R}_\tau, \mathbf{R}_{\tau'} \rangle_F. \tag{3}$$

For GLA, whose decay differs across rows, the same expansion holds for each row of the state. The first sum contains the diagonal terms with $\tau = \tau'$, which depend only on the norm of each per-step error. Conventional quantization methods aim to minimize the error of each quantization and therefore account only for these terms. In contrast, the second sum contains the off-diagonal terms between different steps. Each of them is the inner product of two per-step errors, weighted by the decay weight $a_{\tau:T}\, a_{\tau':T}$ that measures how much of the two errors the state retains at step T. The inner product is positive when the two errors point in similar directions.

In Eq. 3, each term of $\|\mathbf{E}_T\|_F^2$ is the product of two factors: the inner product $\langle \mathbf{R}_\tau, \mathbf{R}_{\tau'} \rangle_F$ of the per-step errors and the decay weight $a_{\tau:T}\, a_{\tau':T}$. Fig. 1 (top) plots both factors on WikiText-103 sequences. First, the inner products are positive not only on the diagonal but also off the diagonal for nearly all pairs of steps, even for steps several thousand tokens apart. Second, the decay weights determine how many past steps the state retains. They remain large over several thousand tokens for GDN, GLA, and RetNet, but only over the last few hundred tokens for Mamba2. The number of diagonal terms grows linearly, whereas the number of off-diagonal terms grows quadratically.

Hence, the stronger the off-diagonal correlation and the slower the decay, the more the positive off-diagonal terms dominate $\|\mathbf{E}_T\|_F^2$, even though each of them is smaller than a diagonal term.

To quantify this effect, the coherent gain $g_T = \|\mathbf{E}_T\|_F^2 / \sum_{\tau \le T} a_{\tau:T}^2 \|\mathbf{R}_\tau\|_F^2$ compares the squared Frobenius norm of the accumulated deviation with that of its diagonal terms. If the errors of different steps were uncorrelated, g_T would remain close to 1. Instead, Fig. 1 (bottom) shows that the accumulated deviation far exceeds its diagonal terms on all four models, and the gap widens with context to up to $86\times$ at 4K tokens. Consistent with the decay weights, the gain of Mamba2 grows only slightly after the first thousand tokens and stays around 4, whereas that of RetNet grows the most. Perplexity under 4-bit quantization follows the same trend: on WikiText-103 at 4K tokens, it increases by 8.6 for Mamba2 but by 157.1 for RetNet (Table 2). Consequently, **the errors of different steps combine coherently, and the resulting off-diagonal terms dominate the accumulated deviation**, especially in models that decay slowly.

2.3 Coherence Arises from Lost Updates

To identify the source of the coherence in Sec. 2.2, we examine how each update is rounded. The grid spacing Δ_t is fitted to the range of each group, whose elements accumulate many past updates. A single update $\mathbf{B}_t$ is therefore typically much smaller than Δ_t in each element. Adding such an update often does not change the rounding result, so the quantized state is the same as if the update had never been written. In this case, we say that the update is lost in rounding. We mark the elements with lost updates by the element-wise indicator $\mathbf{M}_t = \mathbf{1}\big[\mathcal{Q}_t(\mathbf{A}_t\widehat{\mathbf{S}}_{t-1} + \mathbf{B}_t) = \mathcal{Q}_t(\mathbf{A}_t\widehat{\mathbf{S}}_{t-1})\big]$, where both sides are rounded onto the same grid. Such losses are common: under 4-bit quantization, the median ratio of a nonzero update element to its grid spacing is only 0.02–0.21 across the four models, and 68–96% of these elements are lost at each step (details in App. A.2). On these elements, Eq. 1 becomes

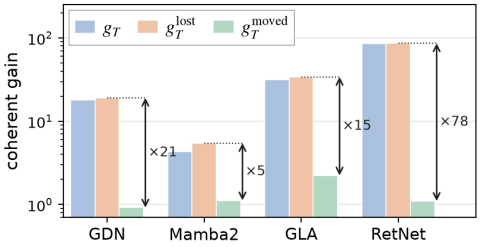

Figure 2: Coherent gains of all per-step errors (g_T) and of the errors on elements whose update is lost (g_T^{lost}) or moved to a different grid point (g_T^{moved}). Arrows mark $g_T^{\text{lost}}/g_T^{\text{moved}}$.

$$\mathbf{R}_t = \big(\mathcal{Q}_t(\mathbf{A}_t\widehat{\mathbf{S}}_{t-1}) - \mathbf{A}_t\widehat{\mathbf{S}}_{t-1}\big) - \mathbf{B}_t. \tag{4}$$

The first term is the rounding error of the decayed state and does not directly depend on $\mathbf{B}_t$, whereas the second term is the update itself with the opposite sign. The error thus contains the entire update that quantization fails to write. Consequently, errors on elements with lost updates are expected to inherit the correlation of the updates across steps.

To test this, we split each per-step error into $\mathbf{R}_t^{\text{lost}} = \mathbf{M}_t \odot \mathbf{R}_t$ and $\mathbf{R}_t^{\text{moved}} = (\mathbf{1} - \mathbf{M}_t) \odot \mathbf{R}_t$, where the latter covers the elements whose update changes the rounding result. We then measure the coherent gain of each part in the same way as g_T:

$$g_T^{\text{lost}} = \frac{\big\|\sum_{\tau \le T} a_{\tau:T}\mathbf{R}_\tau^{\text{lost}}\big\|_F^2}{\sum_{\tau \le T} a_{\tau:T}^2 \|\mathbf{R}_\tau^{\text{lost}}\|_F^2}, \qquad g_T^{\text{moved}} = \frac{\big\|\sum_{\tau \le T} a_{\tau:T}\mathbf{R}_\tau^{\text{moved}}\big\|_F^2}{\sum_{\tau \le T} a_{\tau:T}^2 \|\mathbf{R}_\tau^{\text{moved}}\|_F^2}. \tag{5}$$

Since each gain is normalized by its own diagonal terms, the two gains remain comparable even though the two parts cover different numbers of elements. Fig. 2 shows a sharp contrast between the two parts on WikiText-103 sequences at $T = 4$K. On the elements with lost updates, g_T^{lost} is comparable to g_T on all four models. In contrast, g_T^{moved} stays close to 1 (at most about 2), so the errors on the remaining elements are only weakly correlated across steps. Consequently, **the coherence of the per-step errors arises mainly from the small updates lost in rounding.** Reducing the accumulated deviation therefore calls for writing these small updates into the state, rather than reducing per-step errors alone.

3 Preserving Updates under Low-Bit Storage

Sec. 2.3 shows that the harmful part of the accumulated deviation comes from small updates lost in rounding. These updates are lost because each of them is rounded at a single step onto a grid much

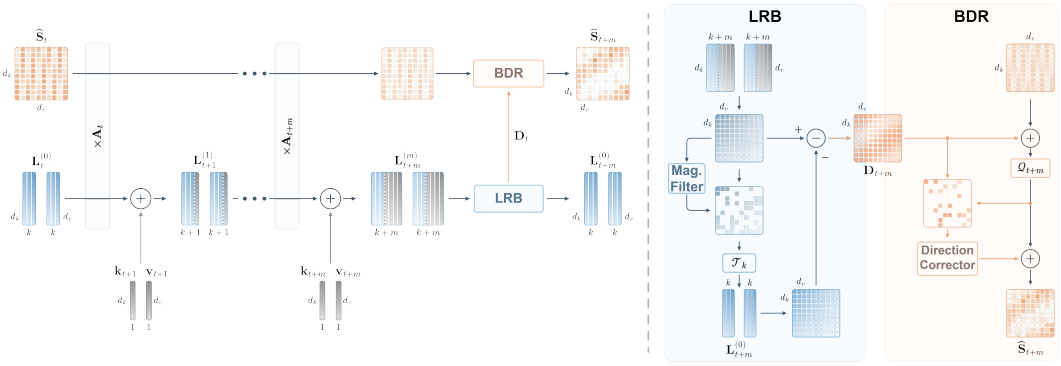

Figure 3: Overview of our method. LRB accumulates updates in a low-rank bypass and writes them into the quantized state every m steps, and BDR corrects the reversed elements during rounding.

coarser than itself. We therefore preserve them at two stages of writing an update into the state: before rounding and at rounding. Before rounding, we propose Low-Rank Bypass (LRB), which changes *what* is rounded. It accumulates small updates in a compact low-rank buffer and periodically merges its sufficiently large elements into the state, which preserves their magnitude at a small storage cost (Sec. 3.1). At rounding, we propose Bounded Directional Rounding (BDR), which changes *how* an update is rounded. When rounding would move an element against its update, BDR instead rounds it toward the update, which preserves the direction of the update without additional storage (Sec. 3.2). Since the two methods act at different stages, each can be applied on its own, and combining them gives the best results (App. A.5).

3.1 LRB: Preserving the Magnitude of Small Updates

Accumulating updates outside the quantized state requires extra storage, which would amount to a full-precision copy of the state if kept densely. To store them compactly, LRB exploits the structure of the updates. Each update is the outer product $\mathbf{k}_t\bar{\mathbf{v}}_t^\top$ of a key and a written value[1], so the sum of several updates has rank at most the number of updates. LRB therefore keeps them in a low-rank bypass $\mathbf{L}_t = \mathbf{K}_t\bar{\mathbf{V}}_t^\top$, where $\mathbf{K}_t$ and $\bar{\mathbf{V}}_t$ collect their keys and written values as columns. Every m steps, LRB merges the bypass into the quantized state. Between two merges, the quantized state only decays, and each new update is appended to the bypass:

$$\widehat{\mathbf{S}}_t = \mathbf{A}_{t_0:t}\widehat{\mathbf{S}}_{t_0}, \qquad \mathbf{L}_t = \mathbf{A}_t\mathbf{L}_{t-1} + \mathbf{k}_t\bar{\mathbf{v}}_t^\top = \big[\mathbf{A}_t\mathbf{K}_{t-1},\, \mathbf{k}_t\big]\big[\bar{\mathbf{V}}_{t-1},\, \bar{\mathbf{v}}_t\big]^\top, \tag{6}$$

where t_0 is the most recent merge step. We store $\widehat{\mathbf{S}}_{t_0}$ unchanged together with the decay $\mathbf{A}_{t_0:t}$ accumulated since then, so the stored state is not rounded again until the next merge. The last equality shows that the bypass is updated through its factors, by decaying the keys and appending one column to each factor. A query also reads the bypass through its factors: $\mathbf{o}_t = (\widehat{\mathbf{S}}_t + \mathbf{L}_t)^\top\mathbf{q}_t = \widehat{\mathbf{S}}_t^\top\mathbf{q}_t + \bar{\mathbf{V}}_t\big(\mathbf{K}_t^\top\mathbf{q}_t\big)$. Hence $\mathbf{L}_t$ is never materialized between merges.

At a merge, some accumulated updates are still small, and merging them loses them in rounding. A merge should therefore hold back the elements marked by $\mathbf{M}_t$ (Sec. 2.3), whose updates are lost in rounding, and leave them in the bypass to accumulate until a later merge. However, $\mathbf{M}_t$ is not available before the merge: it relies on the grid, which is fitted only after the merged elements are chosen. We therefore approximate $\mathbf{M}_t$ by the magnitude of each accumulated update, and keep in the bypass the elements marked by $\mathbf{M}_t^L = \mathbf{1}\big[\,|\mathbf{L}_t| < \epsilon_L\Delta_t\,\big]$, where Δ_t is the grid spacing of $\mathbf{A}_{t_0:t}\widehat{\mathbf{S}}_{t_0} + \mathbf{L}_t$. Since the kept elements no longer form a few outer products, we retain only their rank-k approximation, denoted by $\mathcal{T}_k$, to keep the bypass in factored form, and merge the residual into the state. A merge at step t thus computes

$$\mathbf{L}'_t = \mathcal{T}_k\big(\mathbf{L}_t \odot \mathbf{M}_t^L\big), \qquad \mathbf{D}_t = \mathbf{L}_t - \mathbf{L}'_t, \qquad \widehat{\mathbf{S}}_t = \mathcal{Q}_t\big(\mathbf{A}_{t_0:t}\widehat{\mathbf{S}}_{t_0} + \mathbf{D}_t\big), \tag{7}$$

where $\mathbf{D}_t$ is the update written into the quantized state, after which t_0 is set to t. We compute $\mathcal{T}_k$ with a randomized SVD (Halko et al., 2011), which finds the top-k components from a few random

[1] $\bar{\mathbf{v}}_t = \beta_t\big(\mathbf{v}_t - (\mathbf{A}_t\mathbf{S}_{t-1})^\top\mathbf{k}_t\big)$ for GDN and $\bar{\mathbf{v}}_t = \mathbf{v}_t$ for the other models (Table 1).

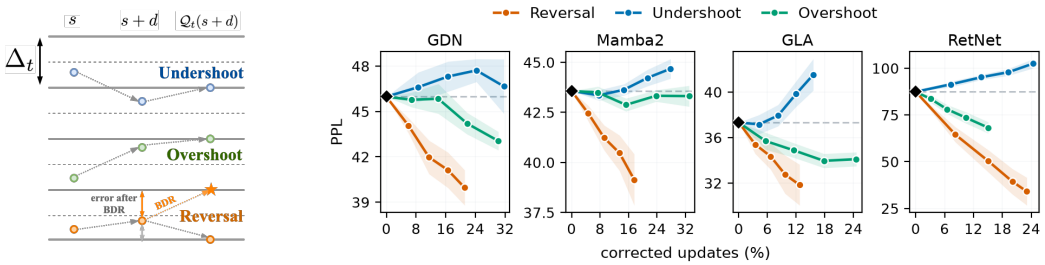

Figure 4: Left: three ways rounding distorts an update d, and the correction of BDR. Right: perplexity after removing the rounding errors of each case.

projections instead of a full SVD (implementation details in App. A.4). It returns $\mathbf{L}'_t$ in factored form, so the bypass restarts from k columns after each merge and holds at most $k+m$ columns, i.e., $(k+m)(d_k+d_v)$ elements. Fig. 3 illustrates the steps between two merges, where $\mathbf{L}^{(j)}$ denotes the bypass after j updates are appended since the last merge, so that $\mathbf{L}^{(0)} = \mathbf{L}'$.

3.2 BDR: Preserving the Direction of Lost Updates

Beyond whether an update is lost (Sec. 2.3), how it is rounded also determines the update actually written into the state. Consider a single element at a merge (Eq. 7), and let s, d, and $\widetilde{d}$ denote its elements in $\mathbf{A}_{t_0:t}\widehat{\mathbf{S}}_{t_0}$, $\mathbf{D}_t$, and the effective update $\widetilde{\mathbf{D}}_t = \widehat{\mathbf{S}}_t - \mathbf{A}_{t_0:t}\widehat{\mathbf{S}}_{t_0}$. The effective update then splits into two parts:

$$\widetilde{d} = \mathcal{Q}_t(s+d) - s = \underbrace{\mathcal{Q}_t(s+d) - \mathcal{Q}_t(s)}_{\text{rounded update}} + \underbrace{\mathcal{Q}_t(s) - s}_{\text{re-rounding shift}} . \tag{8}$$

The two parts relate to d differently. The rounded update keeps the sign of d, since it is either zero or a multiple of Δ_t with the sign of d. In contrast, the re-rounding shift does not follow d. Since the grid is refitted at every rounding, s generally no longer lies on the grid and is moved to its nearest grid point, which can lie on either side of s. The shift can therefore have the opposite sign of d. Consequently, the effective update deviates from d in one of three ways, as illustrated in Fig. 4 (left): (i) *undershoot*, where $\widetilde{d}$ keeps the sign of d but $|\widetilde{d}| < |d|$; (ii) *overshoot*, where $\widetilde{d}$ keeps the sign of d but $|\widetilde{d}| > |d|$; and (iii) *reversal*, where $\widetilde{d}$ has the opposite sign of d. A reversal arises only from a small update that is lost in rounding, with $|d| < \Delta_t/2$ (Lemma 1 (i)).

Reversals are the most harmful. To compare the three cases, we remove the rounding errors of each case separately under 4-bit quantization and measure the change in perplexity. At every step, we set $\widetilde{d} = d$ on a random fraction of the elements in one case. As Fig. 4 (right) shows, removing the errors of reversals reduces perplexity the most on all four models at the same fraction. In contrast, removing the errors of overshoots helps less, and removing those of undershoots does not help. This agrees with Sec. 2.3. When an update is lost, the rounded update in Eq. 8 vanishes, so the error consists of $-d$ and the re-rounding shift. The part $-d$ accumulates coherently, and in a reversal the re-rounding shift points the same way and further enlarges it.

Bounded directional rounding. Since reversals are the most harmful, we correct them by moving the rounded element $\mathcal{Q}_t(s+d)$ one grid step in the direction of d, as marked by the star in Fig. 4 (left). The effective update then follows d. The corrected element, however, lies farther from $s+d$, so the rounding error grows and the reversal becomes an overshoot (Lemma 1 (ii)–(iii)). Since the move spans a full grid step, the corrected update exceeds $\Delta_t/2$ even when d is tiny, which amplifies a small d many times (Lemma 1 (ii)). To bound this amplification, we correct a reversal only when $|d| > \epsilon_B \Delta_t$. This gives BDR, which rounds $s+d$ as

$$\mathcal{Q}_t^{\mathrm{BDR}}(s+d) = \begin{cases} \mathcal{Q}_t(s+d) + \mathrm{sign}(d)\,\Delta_t, & \text{if } \widetilde{d}\,d < 0 \text{ and } |d| > \epsilon_B \Delta_t, \\ \mathcal{Q}_t(s+d), & \text{otherwise.} \end{cases} \tag{9}$$

The corrected update is then less than $|d|/\epsilon_B$ (Lemma 1 (iv)). A smaller ϵ_B thus corrects more reversals but allows larger amplification, and we examine this trade-off in App. A.5. Although BDR

Model	Method	Bits	PG-19 PPL ↓				WikiText-103 PPL ↓			
			1K	2K	4K	8K	1K	2K	4K	8K
GDN	FP32	32.0	84.78	76.12	70.64	66.84	30.84	26.38	24.85	24.19
	INT4	4.50	96.21	101.08	141.76	220.36	36.10	36.49	48.15	79.76
	INT4+Had	4.50	95.56	96.98	112.52	137.94	35.23	34.69	40.92	53.71
	Ours	5.25	85.91	77.90	74.11	72.28	31.38	27.23	26.27	26.85
GLA	FP32	32.0	64.71	60.89	57.96	55.85	25.06	21.66	20.69	21.57
	INT4	4.50	93.64	104.05	125.15	158.79	30.02	32.53	41.68	66.55
	INT4+Had	4.50	88.28	89.00	94.07	99.13	31.22	31.21	34.18	41.69
	Ours	5.31	65.82	62.51	60.89	62.24	25.47	22.36	22.19	24.86
Mamba2	FP32	32.0	77.88	70.40	66.40	63.99	29.43	26.20	25.32	25.19
	INT4	4.50	86.79	81.90	83.45	87.96	32.96	31.65	33.89	37.43
	INT4+Had	4.50	88.59	84.87	86.01	88.21	32.22	30.54	32.36	35.74
	Ours	5.64	80.30	73.37	70.68	69.72	30.30	27.60	27.66	28.78
RetNet	FP32	32.0	66.45	62.46	61.07	60.02	25.66	22.38	22.51	24.30
	INT4	4.50	190.65	255.75	533.85	3170.42	59.22	75.61	179.61	1193.21
	INT4+Had	4.50	147.93	203.44	460.38	2749.92	46.25	60.32	142.60	918.77
	Ours	5.25	69.66	67.88	70.22	75.96	26.68	23.76	24.97	28.80

Table 2: Language modeling quality of 1.3B models on PG-19 and WikiText-103. INT4+Had applies a Hadamard rotation before INT4 quantization. Bits is the average storage per state element over decoding. The configurations of our method are listed in Table 6.

enlarges the rounding error (Lemma 1 (iii)), it improves the model (App. A.5), suggesting that the direction of an update matters more than its rounding error at a single step. Since BDR only changes the rounding rule, it requires no additional storage. Fig. 3 (right) illustrates the procedure of BDR, where the direction corrector applies Eq. 9 to the reversed elements with $|d| > \epsilon_B \Delta_t$.

4 Experiments

Models. We evaluate four recurrent models at the 1.3B scale: Gated DeltaNet (Yang et al., 2025), GLA (Yang et al., 2024), Mamba2 (Dao & Gu, 2024), and RetNet (Sun et al., 2023). For GLA and RetNet, we use the public checkpoints released by the flash-linear-attention (FLA) library[2]. Since FLA provides no 1.3B checkpoints for GDN and Mamba2, we pretrain them ourselves on 100B tokens of FineWeb-Edu (Penedo et al., 2024) with a sequence length of 4,096. Training uses AdamW (Loshchilov & Hutter, 2017) with a peak learning rate of 4×10^{-4}, cosine decay, a weight decay of 0.1, and a batch size of 0.5M tokens. All state quantization methods compared below are applied directly to these checkpoints without fine-tuning. The hyperparameters of our method for each model are listed in Table 6, and ablation studies are provided in App. A.5.

Language modeling and commonsense reasoning. We measure word-level perplexity on PG-19 (Rae et al., 2020) and WikiText-103 (Merity et al., 2016) at context lengths from 1K to 8K tokens (Table 2). We further evaluate LAMBADA (Paperno et al., 2016) and commonsense reasoning on SciQ (Welbl et al., 2017), COPA (Roemmele et al., 2011), and HellaSwag (Zellers et al., 2019) with the LM Evaluation Harness (Gao et al., 2021) (Table 3). We compare our method with full-precision FP32 states and two 4-bit baselines. INT4 stores the state as 4-bit integers using the quantizer of App. A.1 at every step, and INT4+Had applies a Hadamard rotation before INT4 quantization to suppress outliers (Ashkboos et al., 2024). First, the degradation of INT4 grows with context length and follows the coherent gain in Sec. 2.2, being mildest for Mamba2 and most severe for RetNet. Likewise, INT4 clearly degrades LAMBADA, which relies on a broad context, but changes the commonsense tasks with short inputs only slightly. Second, INT4+Had improves INT4 at long contexts on GDN, GLA, and RetNet, but still degrades sharply as the context grows. Reducing the per-step error thus cannot prevent the accumulation analyzed in Sec. 2. Third, our method stays far closer to FP32 than both baselines on all four models, and its advantage grows with context length. On LAMBADA, it brings perplexity close to FP32 and keeps accuracy within 0.5 points of FP32 on GDN, GLA, and Mamba2.

Recall-intensive tasks. We further evaluate the recall-intensive tasks collected by Arora et al. (2024), as shown in Table 4. These tasks require the model to retrieve specific information from

[2] https://huggingface.co/fla-hub

Model	Method	LMB-PPL ↓	LMB-Acc ↑	SciQ ↑	COPA ↑	HS ↑
GDN	FP32	12.53	46.38	90.9	73.0	55.79
	INT4	17.94	39.90	90.3	74.0	55.54
	Ours	12.85	46.01	91.2	73.0	55.84
GLA	FP32	15.56	46.46	80.7	70.0	48.86
	INT4	22.48	38.02	79.1	69.0	48.51
	Ours	15.75	46.05	80.7	70.0	48.86
Mamba2	FP32	10.82	50.18	88.9	80.0	59.36
	INT4	12.13	47.76	89.4	81.0	59.05
	Ours	10.89	49.85	89.5	79.0	59.27
RetNet	FP32	17.00	43.08	83.3	72.0	48.14
	INT4	19.00	42.67	81.5	73.0	46.64
	Ours	17.84	42.42	82.5	72.0	48.07

Table 3: Zero-shot performance of 1.3B models on language modeling and commonsense reasoning benchmarks. LMB: LAMBADA; HS: HellaSwag, reported as length-normalized accuracy.

Model	Method	F-512	F-1K	S-512	S-1K	N-512	N-1K	SQ	TQA	DR	Avg
GDN	FP32	51.2	41.1	42.9	36.7	31.1	31.1	41.0	57.7	21.7	39.4
	INT4	30.1	25.4	33.7	25.5	26.5	24.4	39.0	56.7	16.3	30.8
	Ours	52.5	39.1	41.2	35.7	28.2	29.4	41.7	58.0	20.7	38.5
GLA	FP32	64.9	46.8	44.2	31.3	34.0	30.3	40.3	53.7	20.7	40.7
	INT4	20.4	13.7	36.1	28.2	27.7	25.2	33.7	47.3	17.0	27.7
	Ours	59.9	41.1	41.2	31.6	34.5	30.7	41.0	52.7	24.7	39.7
Mamba2	FP32	48.5	34.1	41.5	29.9	35.7	34.0	38.7	58.7	20.0	37.9
	INT4	28.1	15.4	31.0	19.0	29.8	23.1	39.3	55.7	20.0	29.0
	Ours	36.5	24.1	35.7	21.1	34.9	27.7	40.0	57.3	20.7	33.1
RetNet	FP32	68.2	39.8	48.0	32.3	29.4	28.6	44.7	47.3	23.0	40.1
	INT4	13.7	4.7	31.0	17.3	17.2	14.7	34.7	43.3	16.0	21.4
	Ours	59.9	32.8	45.6	25.5	26.5	23.9	42.0	47.3	20.7	36.0

Table 4: Accuracy ↑ on recall-intensive tasks (Arora et al., 2024) with 1.3B models. F: FDA, S: SWDE, N: NQ, SQ: SQuADv2, TQA: TriviaQA, DR: DROP; 512 and 1K denote the context length.

the context, and thus directly test the key–value associations stored in the state. They include information extraction on FDA (Arora et al., 2023) and SWDE (Lockard et al., 2019), and question answering on NQ (Kwiatkowski et al., 2019), SQuADv2 (Rajpurkar et al., 2018), TriviaQA (Joshi et al., 2017), and DROP (Dua et al., 2019). First, INT4 substantially degrades recall even at contexts of only 512 and 1K tokens. The drop is largest on FDA, which requires copying exact values from the context. Second, our method narrows the gap to FP32 on all four models. On GDN and GLA, its average accuracy is within about one point of FP32. On RetNet, where INT4 loses the most, it recovers most of the lost accuracy. Hence, preserving state updates also preserves the ability to recall information from the context.

Efficiency. We implement each decoding step of our method with fused Triton kernels. Fig. 5 compares our method with FP32 states on an RTX 3060 GPU at batch sizes of 16, 32, and 48. The left panel reports the memory occupied by the recurrent states of all layers, excluding weights and activations. The right panel reports the end-to-end decoding throughput of the whole model, measured under the same setup for both methods. First, our method reduces state memory by $5.3\times$ on GDN, GLA, and RetNet, and by $4.7\times$ on Mamba2. The reduction is smaller on Mamba2 because its state per head is smaller, so the bypass takes a larger share of the storage. Second, at a batch size of 48, our method speeds up decoding by 1.3–1.9$\times$, although LRB and BDR add extra computation. This is because decoding reads and writes the entire state at every step, so storing the state in low precision reduces memory traffic. Moreover, the speedup grows with batch size, as the state becomes a larger bottleneck in larger batches.

5 Related Work

LLM quantization. Post-training quantization reduces the inference cost of large language models (LLMs) through low-precision representations. For model weights, GPTQ (Frantar et al., 2022) uses approximate second-order information and AWQ (Lin et al., 2024) uses activation statistics to

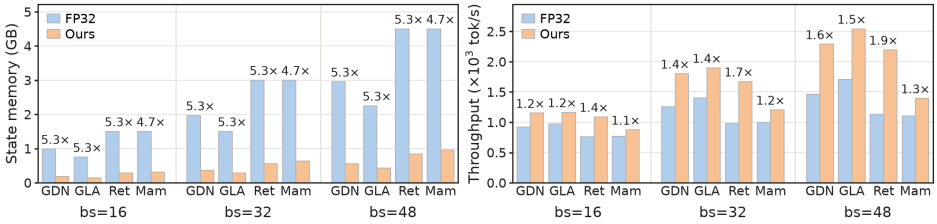

Figure 5: State memory and end-to-end decoding throughput of our method and FP32 states.

reduce quantization error. Moreover, SmoothQuant (Xiao et al., 2023), OmniQuant (Shao et al., 2024), and QuaRot (Ashkboos et al., 2024) quantize activations by suppressing outliers, while KIVI (Liu et al., 2024) and KVQuant (Hooper et al., 2024) extend quantization to the KV cache. All these methods quantize each tensor once and thus aim to reduce the error of that single quantization. In contrast, a recurrent state is requantized at every step, so its errors accumulate through the recurrence as analyzed in Sec. 2.2.

Quantization error compensation. Low-precision training faces a similar problem, as small weight updates can be lost to rounding. Mixed-precision training therefore keeps full-precision master weights (Micikevicius et al., 2017), and Zamirai et al. (2020) apply stochastic rounding and Kahan summation to BFloat16 training. Similarly, error feedback (Karimireddy et al., 2019) stores the residual of gradient compression and adds it to the next gradient before compression. Master weights, Kahan summation, and error feedback all keep a buffer as large as the tensor they protect, which is acceptable because their main savings lie in computation, activation memory, or communication. In contrast, our goal is to reduce state storage. Such a buffer would take as much memory as the full-precision state. LRB therefore keeps a low-rank buffer by exploiting the outer-product structure of state updates. Moreover, unlike weights stored on a fixed floating-point grid, a recurrent state is rounded onto a grid that is refitted each time, so rounding can also reverse an update, which BDR corrects.

Recurrent state quantization. Quantization of recurrent models has mainly targeted Mamba models. Quamba (Chiang et al., 2025b), MambaQuant (Yue et al., 2025), and Quamba2 (Chiang et al., 2025a) focus on weights and activations, and Quamba2 also quantizes the cached states. Q-Mamba (Tianqi et al., 2025) quantizes the state cache of Mamba2 to 4 bits with separate scales along the state and channel dimensions, and calibrates the quantized model by block-wise reconstruction on calibration data. DAMP (Zhang et al., 2026) quantizes the states of GDN and Kimi Delta Attention by keeping the riskiest key channels in FP16 and the rest in INT8, with the risk estimated offline from quantization-error energy and decay-based persistence. Our method differs from these works in two ways. First, whereas these methods reduce the error of each state quantization, our method targets the coherence of errors across steps by preserving the updates that rounding would lose or reverse. Second, it requires no gradient-based fine-tuning or calibration statistics, and applies to diverse recurrent models, including GDN, GLA, Mamba2, and RetNet.

6 Conclusion

We study low-bit storage of recurrent states from the perspective of the updates written into them. We show that per-step quantization errors accumulate coherently through the recurrence, and that this coherence arises mainly from updates lost in rounding. We further find that rounding can also reverse updates, which is the most harmful way it distorts an update. Guided by these findings, LRB preserves the magnitude of small updates before rounding, and BDR preserves their direction at rounding while bounding the resulting overshoot. Across GDN, GLA, Mamba2, and RetNet, our method keeps model quality much closer to FP32 than INT4, while reducing state memory by 4.7–5.3$\times$ and speeding up decoding by up to 1.9$\times$.

Discussion and Limitations. Our evaluation covers four recurrent architectures, and extending it to more models, including hybrid models that combine recurrent layers with softmax attention (Chandiramani et al., 2026), is a natural next step. Moreover, we quantize only the recurrent states and keep weights and activations in their original precision, so combining our method with weight and activation quantization may further reduce inference cost.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and revision, proofreading, and limited code development. We also used these tools to explore alternative formulations during research development. We developed the research ideas, theoretical development, method and experimental design, and core implementation ourselves. We manually reviewed and verified all AI-assisted text, mathematical feedback, and code. We did not use generative AI tools for synthetic data generation, translation, dataset cleaning or reformatting, qualitative or thematic data analysis, or interpretation of experimental results. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with generative AI.

ETHICS AND REPRODUCIBILITY STATEMENT

This work does not involve human subjects, user studies, personal or sensitive data, or the collection or release of new datasets. We use publicly available datasets in all experiments. For models that are not publicly available, we train them ourselves using open-source training frameworks, including Flash Linear Attention (Yang & Zhang, 2024) and Flame (Zhang & Yang, 2025). We provide the configurations and experimental details required for reproducibility in Sec. 4. We will release the full code upon publication.

REFERENCES

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*, 2023.

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

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.

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.

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 *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=Zm0Kper4yx.

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 *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=mnna9LUg7P.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

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

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.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53 (2):217–288, 2011.

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=0LXotew9Du.

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

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International conference on machine learning*, pp. 3252–3261. PMLR, 2019.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=L057s2Rq8O.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

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

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

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pp. 90–95, 2011.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45472–45496, 2024.

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

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.

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.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

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.

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

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ia5XvxFUJT.

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.

Yuxuan Yue, Xing Hu, Dawei Yang, Zhihang Yuan, Zixu Jiang, Zhixuan Chen, Jiangyong Yu, and Sifan Zhou. Mambaquant: Quantizing the mamba family with variance aligned rotation methods. In *International Conference on Learning Representations*, volume 2025, pp. 33231–33250, 2025.

Pedram Zamirai, Jian Zhang, Christopher R Aberger, and Christopher De Sa. Revisiting bfloat16 training. *arXiv preprint arXiv:2010.06192*, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

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. *arXiv preprint arXiv:2608.27513*, 2026.

Yu Zhang and Songlin Yang. Flame: Flash language modeling made easy, January 2025. URL `https://github.com/fla-org/flame`.

A Appendix

A.1 State Quantization Details

Layout and grouping. For every model, we write the per-head state as $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$, with rows indexed by key dimensions and columns by value dimensions. The state is quantized in groups of 128 contiguous elements along the key dimension, so each group lies within a single column. All groups use the same bit width and are quantized independently.

Quantizer. Let $x_1, \ldots, x_{128}$ be the elements of one group to be stored at step t. The quantizer fits an n-bit uniform grid to their range and rounds each element to the nearest grid point:

$$\Delta_t = \frac{\max_i x_i - \min_i x_i}{2^n - 1}, \qquad z_t = \Big\lfloor -\frac{\min_i x_i}{\Delta_t} \Big\rceil, \tag{10}$$

$$q_i = \mathrm{clamp}\Big(\Big\lfloor \frac{x_i}{\Delta_t} \Big\rceil + z_t,\ 0,\ 2^n - 1 \Big), \qquad \mathcal{Q}_t(x_i) = \Delta_t\,(q_i - z_t), \tag{11}$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer and $\mathrm{clamp}(x, a, c) = \min(\max(x, a), c)$. Since z_t is an integer, every grid point is an integer multiple of Δ_t, so zero is represented exactly. Because the grid covers the range of the group up to the rounding of z_t, nearest rounding changes each element by at most $\Delta_t/2$. We use $n = 4$ and store the integer codes q_i, together with Δ_t and z_t in FP32 once per group, giving an average of $(4 \cdot 128 + 2 \cdot 32)/128 = 4.5$ bits per state element.

Refitting and computation. The INT4 baseline quantizes the state at every step, whereas our method quantizes it only at merges. In both cases, Δ_t and z_t are refitted to the state being stored, namely $\mathbf{A}_t \widehat{\mathbf{S}}_{t-1} + \mathbf{B}_t$ in Eq. 1 or $\mathbf{A}_{t_0:t} \widehat{\mathbf{S}}_{t_0} + \mathbf{D}_t$ in Eq. 7, so the grid of each group changes over time. Between merges, our method keeps the stored codes unchanged and applies the accumulated decay when the state is read. This accumulated decay adds d_k FP32 scalars per head for GLA and one FP32 scalar per head for the other models. All computation is performed in FP32: the stored state is dequantized, decayed, and updated before being quantized again. The output at each step is read from the updated state before it is quantized.

Evaluation protocol. In all evaluations, every method runs each model in its recurrent form from an empty state and processes one token at a time, including the context tokens. For the quantized methods, the state is kept in low-precision storage throughout, exactly as in decoding, so that a context of T tokens emulates T decoding steps with a quantized state.

A.2 Statistics of Lost Updates

Table 5 supports the claim in Sec. 2.3 that a single update is typically much smaller than the grid spacing and is often lost in rounding. We use the same INT4 trajectories on WikiText-103 (Merity et al., 2016) as in Fig. 1, where the updates $\mathbf{B}_t$ are those computed by the INT4 model itself. For each nonzero element b of $\mathbf{B}_t$, we compute its ratio $|b|/\Delta_t$ to the grid spacing of its group, and report the quartiles Q_{25}, Q_{50}, and Q_{75} of this ratio. We also report the fraction of these elements whose update is lost, i.e., whose rounding result stays unchanged.

Model	Q_{25}	Q_{50}	Q_{75}	Lost fraction
GDN	0.060	0.207	0.582	68.0%
GLA	0.021	0.072	0.209	83.7%
Mamba2	0.052	0.188	0.543	72.4%
RetNet	0.005	0.017	0.046	96.4%

Table 5: Quartiles of $|b|/\Delta_t$ and fraction of lost updates under INT4 at $T = 4$K on WikiText-103.

On all four models, the median update is at most about 20% of the grid spacing, and 68–96% of the updates are lost at each step. RetNet has the smallest updates relative to the grid spacing and loses almost all of them, consistent with its largest coherent gain in Sec. 2.2.

A.3 PROPERTIES OF BDR

Consider a single element at a rounding step. As in Sec. 3.2, let s be its element in the decayed state, d the update written to it, and Δ_t the grid spacing of its group. Nearest rounding stores $\mathcal{Q}_t(s+d)$, which gives the effective update $\widetilde{d} = \mathcal{Q}_t(s+d) - s$. When BDR corrects the element, it stores $\mathcal{Q}_t(s+d) + \mathrm{sign}(d)\,\Delta_t$ (Eq. 9), which gives the corrected update $\widetilde{d}' = \widetilde{d} + \mathrm{sign}(d)\,\Delta_t$. We assume that this grid point lies within the range of the grid; otherwise, BDR leaves the element unchanged.

Lemma 1. *Suppose that the update is reversed, i.e., $\widetilde{d}\,d < 0$. Then:*

(i) $\mathcal{Q}_t(s+d) = \mathcal{Q}_t(s)$*, so the update is lost in rounding; moreover,* $|d| + |\widetilde{d}| \le \Delta_t/2$*, and in particular* $|d| < \Delta_t/2$*;*

(ii) $\widetilde{d}'$ *has the sign of* d*, and* $|d| + \Delta_t/2 \le |\widetilde{d}'| < \Delta_t$*, so the correction turns the reversal into an overshoot;*

(iii) $|\widetilde{d}' - d| \ge \Delta_t/2 \ge |\widetilde{d} - d|$*, so the correction enlarges the rounding error;*

(iv) if $|d| > \epsilon_B \Delta_t$*, then* $|\widetilde{d}'| < |d|/\epsilon_B$*.*

Proof. By symmetry, assume $d > 0$, so that $\widetilde{d} < 0$ and $\widetilde{d}' = \Delta_t - |\widetilde{d}|$. Nearest rounding changes each element by at most $\Delta_t/2$ (App. A.1), so $|\mathcal{Q}_t(s+d) - (s+d)| \le \Delta_t/2$.

(i) Since $\widetilde{d} < 0$, we have $\mathcal{Q}_t(s+d) < s < s+d$. Nearest rounding is monotone, and $\mathcal{Q}_t(s+d)$ is itself a grid point, so $\mathcal{Q}_t(s+d) = \mathcal{Q}_t\big(\mathcal{Q}_t(s+d)\big) \le \mathcal{Q}_t(s) \le \mathcal{Q}_t(s+d)$, which gives $\mathcal{Q}_t(s) = \mathcal{Q}_t(s+d)$. Moreover, the distance from $\mathcal{Q}_t(s+d)$ to $s+d$ is $(s+d) - \mathcal{Q}_t(s+d) = |d| + |\widetilde{d}|$, which is at most $\Delta_t/2$. Since $\widetilde{d} \neq 0$, this also gives $|d| < \Delta_t/2$.

(ii) By (i), $|\widetilde{d}| \le \Delta_t/2 - |d|$, so $\widetilde{d}' = \Delta_t - |\widetilde{d}| \ge |d| + \Delta_t/2 > |d| > 0$. Since $|\widetilde{d}| > 0$, we also have $\widetilde{d}' < \Delta_t$.

(iii) Before the correction, $\widetilde{d}$ and d have opposite signs, so $|\widetilde{d} - d| = |d| + |\widetilde{d}| \le \Delta_t/2$ by (i). After the correction, $\widetilde{d}' > d > 0$ by (ii), so $|\widetilde{d}' - d| = \Delta_t - |\widetilde{d}| - |d| \ge \Delta_t/2$, again by (i).

(iv) By (ii), $|\widetilde{d}'| < \Delta_t$, and the condition $|d| > \epsilon_B \Delta_t$ gives $\Delta_t < |d|/\epsilon_B$. □

A.4 IMPLEMENTATION

LRB. The factors of the bypass are stored in FP16. We compute $\mathcal{T}_k$ with a randomized SVD (Halko et al., 2011) using $k + 8$ random projections and two power iterations. Each decoding step is implemented with fused Triton kernels, and $\mathbf{L}_t$ is materialized on chip only at merges. The reported bits per state element include the storage of the bypass.

A.5 HYPERPARAMETERS AND ABLATIONS

Our method has four hyperparameters: the rank k that the bypass keeps after each merge, the merge interval m, the threshold ϵ_L below which accumulated updates stay in the bypass, and the threshold ϵ_B above which BDR corrects a reversal. Table 6 lists the configuration of each model, which we use in all experiments of Sec. 4. We then examine the contributions of LRB and BDR and the effect of each hyperparameter.

Components. Table 7 isolates the contributions of LRB and BDR. First, without LRB, the state is rounded at every step, and BDR alone corrects the reversals after each step. It reduces the perplexity of INT4 in all settings except GLA at 8K, especially on RetNet, the model most affected by INT4 quantization. The exception on GLA is likely because ϵ_B is chosen for LRB+BDR rather than for BDR alone. Second, LRB alone provides a larger improvement, bringing perplexity close to FP32 at 2K and preventing the sharp increase of INT4 at 8K. Third, combining the two gives the best results on all four models at both context lengths, with the largest gain on RetNet at 8K. The two

Model	k	m	ϵ_L	ϵ_B	Avg. bits	Peak bits
GDN	1	15	0.50	0.30	5.25	6.00
GLA	1	15	0.50	0.30	5.31	6.07
Mamba2	1	5	0.10	0.25	5.64	6.78
RetNet	1	15	0.70	0.30	5.25	6.00

Table 6: Configuration of our method for each model. Avg. bits and Peak bits are the average and peak storage per state element during decoding.

	2K (PPL ↓)				8K (PPL ↓)			
Method	GDN	GLA	Mamba2	RetNet	GDN	GLA	Mamba2	RetNet
FP32	26.38	21.66	26.20	22.38	24.19	21.57	25.19	24.30
INT4	36.49	32.53	31.65	75.61	79.76	66.55	37.43	1193.21
BDR	34.22	32.49	30.85	57.92	63.15	75.61	35.10	707.65
LRB	27.37	22.57	28.00	23.88	27.64	25.91	29.90	31.18
LRB+BDR	**27.23**	**22.36**	**27.60**	**23.76**	**26.85**	**24.86**	**28.78**	**28.80**

Table 7: Component ablation on WikiText-103 with the configurations in Table 6.

components thus complement each other: LRB preserves the magnitude of small updates, and BDR preserves the direction of those still lost in rounding.

			PPL ↓		Bits	
Model	k	m	2K	8K	Avg.	Peak
GDN	0	16	27.34	27.27	5.20	6.00
	1	15	**27.23**	**26.85**	5.25	
	2	14	27.33	26.97	5.30	
	4	12	27.33	27.29	5.39	
	8	8	27.52	27.98	5.58	
GLA	0	16	**22.35**	25.63	5.26	6.07
	1	15	22.36	**24.86**	5.31	
	2	14	22.48	25.30	5.36	
	4	12	22.53	26.13	5.45	
	8	8	23.11	28.58	5.64	
RetNet	0	16	24.26	77.86	5.20	6.00
	1	15	**23.76**	**28.80**	5.25	
	2	14	24.09	28.96	5.30	
	4	12	24.28	29.62	5.39	
	8	8	25.53	32.74	5.58	
Mamba2	0	6	28.03	29.34	5.45	6.78
	1	5	**27.60**	**28.78**	5.64	
	2	4	27.70	29.10	5.83	
	3	3	27.91	29.47	6.02	

Table 8: Effect of the bypass rank k on WikiText-103 perplexity. The number of bypass columns $k + m$ is fixed at 16 for GDN, GLA, and RetNet and at 6 for Mamba2. Bits is the storage per state element.

Bypass rank k. Table 8 varies the rank k that the bypass keeps after each merge. To compare at the same peak storage, we fix the total number of bypass columns $k + m$, so a larger k shortens the merge interval m. We fix $k + m = 6$ for Mamba2, since its smaller state makes each bypass column relatively more costly. First, keeping a single rank instead of none improves perplexity at 8K on all four models at almost no extra storage, most notably on RetNet. Without this rank, small updates that remain below the threshold are merged into the state and lost in rounding. Second, increasing k beyond 1 raises the average bits but worsens perplexity, especially at 8K. A larger k shortens m, so the state is rounded more often, while the additional ranks bring little benefit.

Model	$k+m$	PPL ↓ 2K	PPL ↓ 8K	Bits Avg.	Bits Peak
GDN	8	27.99	29.18	4.88	5.25
	12	27.49	27.71	5.06	5.63
	16	27.23	26.85	5.25	6.00
	24	26.94	26.07	5.63	6.76
GLA	8	23.16	29.23	4.94	5.31
	12	22.59	26.42	5.13	5.69
	16	22.36	24.86	5.31	6.07
	24	22.02	23.32	5.69	6.82
RetNet	8	25.69	44.86	4.88	5.25
	12	24.33	33.25	5.06	5.63
	16	23.76	28.80	5.25	6.00
	24	23.26	27.20	5.63	6.76
Mamba2	4	27.92	29.65	5.26	6.02
	6	27.60	28.78	5.64	6.78
	8	27.52	28.34	6.02	7.54

Table 9: Effect of the number of bypass columns $k+m$ on WikiText-103 perplexity, with $k=1$. Bits is the storage per state element.

Bypass size $k+m$. Table 9 varies the number of bypass columns $k+m$ with $k=1$, which sets the merge interval m and the storage of the bypass. First, perplexity improves steadily as the bypass grows, but with diminishing returns. We choose $k+m=16$ as the default ($k+m=6$ for Mamba2), since larger bypasses bring smaller gains. Second, even the smallest bypass, which adds only 0.38–0.76 average bits to INT4, already removes most of the degradation of INT4 at 8K (Table 2). Most of the improvement therefore comes from preserving updates rather than from the extra bits.

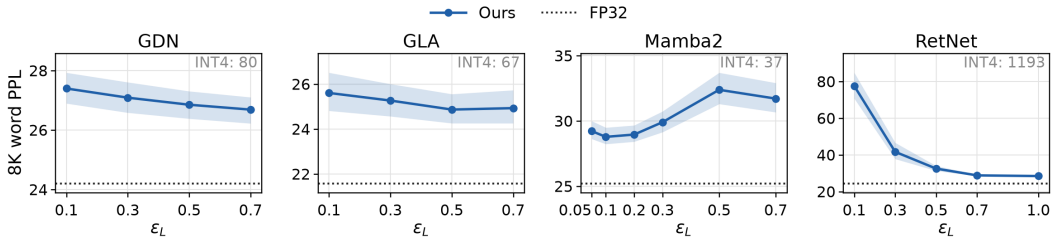

Figure 6: 8K perplexity on WikiText-103 versus ϵ_L, with the INT4 perplexity noted in each panel.

Bypass threshold ϵ_L. Fig. 6 varies ϵ_L, which decides which accumulated updates stay in the bypass at each merge. Across the tested range, our method stays far below INT4 on all four models. The threshold balances two effects. A small threshold merges small updates into the state, where rounding loses them, whereas a large threshold keeps more updates than a rank-1 bypass can represent accurately. This balance differs across models. RetNet, whose INT4 errors accumulate most coherently (Sec. 2.2), benefits from larger thresholds, whereas Mamba2 favors smaller ones.

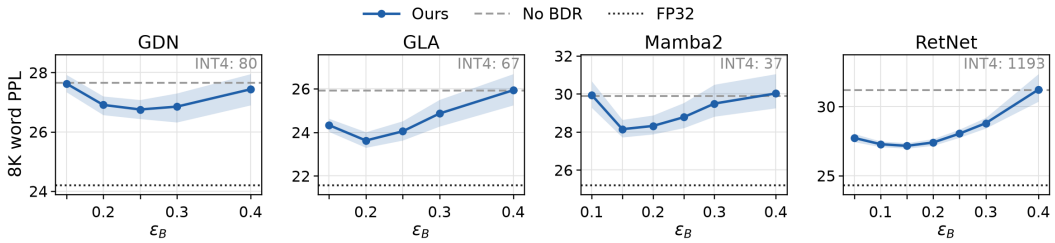

Figure 7: 8K perplexity on WikiText-103 versus ϵ_B, compared with LRB without BDR.

BDR threshold ϵ_B. Fig. 7 varies ϵ_B, which bounds the overshoot introduced by each correction (Sec. 3.2). For ϵ_B between 0.2 and 0.3, BDR lowers perplexity below LRB alone on all four models, most notably on RetNet. As ϵ_B grows, BDR corrects fewer reversals, so its perplexity approaches that of LRB alone. Conversely, a very small ϵ_B also corrects the reversals of tiny updates, whose large overshoots reduce the gain on GDN and GLA.