DAMP: DECAY-AWARE MIXED-PRECISION RECURRENT-STATE QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex reasoning and agentic applications increasingly rely on long-context inference, where growing KV caches increase both memory usage and decoding overhead. Hybrid models reduce these costs by combining Softmax Attention with Gated DeltaNet (GDN) or Kimi Delta Attention (KDA), which maintain fixed-size recurrent states. These states are commonly stored in FP32 and consume substantial GPU memory, while their updates are limited by memory bandwidth. Quantization can reduce both storage footprint and memory traffic, but we find that uniform INT8 and FP8 degrade complex reasoning accuracy, while INT4 and NVFP4 collapse it to near zero. To our knowledge, this is the first study of post-training recurrent-state quantization for GDN and KDA. Our analysis reveals that outliers in GDN and KDA states are concentrated in particular key channels and value dimensions. Learned decay influences how much quantization error is retained. We find that largely the same GDN heads and KDA key channels exhibit slow decay across tasks. Based on these insights, we propose DAMP, which jointly considers quantization error and decay-based error retention to select high-risk key channels offline. Under a fixed storage budget, it retains these channels in FP16 and stores the remainder in INT8. We evaluate DAMP on Qwen3.6-35B, Kimi-Linear-48B and Kimi-K3 across six reasoning and code generation benchmarks. At 9.9 bits per state value, DAMP maintains average accuracy close to FP32. In SGLang, DAMP reduces recurrent-state storage by 69.1%, accelerates the recurrent-state update kernel by up to 2.59×, and lowers full-model time per output token by up to 19.0%.

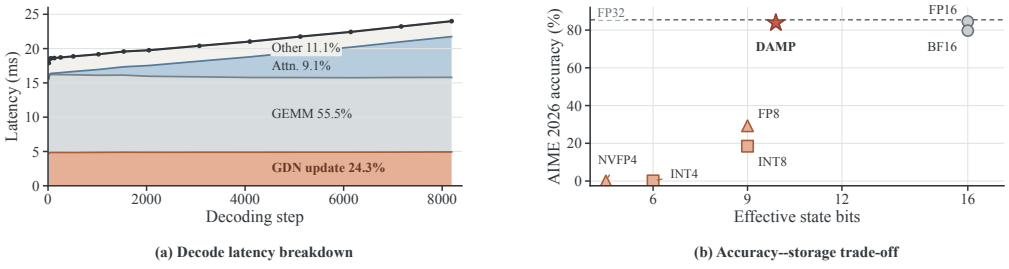


Figure 1: Recurrent-state update cost and accuracy–storage trade-off on Qwen3.6-35B. (a) Decode latency breakdown at batch size 256. (b) AIME 2026 accuracy versus effective state-storage bits.

1 INTRODUCTION

Complex reasoning and agentic applications increasingly rely on long contexts and extended sequences of reasoning and interaction (Team et al., 2025a; 2026). The growing KV cache, however, makes long-context inference memory-intensive. Recent models such as Qwen3.8 (Qwen Team, 2026c;a), Qwen3.8-Flash-Next (Qwen Team, 2026d), Kimi-K3 (Team et al., 2026), and GLM-5.3-Flash (GLM-5-Team et al., 2026) adopt hybrid architectures that interleave Gated DeltaNet (GDN) (Yang et al., 2025) or Kimi Delta Attention (KDA) (Team et al., 2025b) layers with full or sparse attention layers.

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.

These linear-attention layers summarize past tokens in a fixed-size state matrix. Serving systems commonly store these states in FP32. For example, serving Qwen3.6-35B (Qwen Team, 2026b) with SGLang at batch size 256 requires an estimated 80 GB of GPU memory for FP32 recurrent states, including active states and checkpoints retained for prefix caching. Recurrent states must also be read and updated at every decoding step. These updates are limited by memory bandwidth and account for 24.3% of decoding latency (Figure 1a). State quantization therefore offers a direct way to reduce both the storage footprint and the memory traffic of recurrent updates.

However, recurrent states are difficult to quantize without degrading accuracy. We find that uniform INT8 and FP8 quantization already degrades complex reasoning accuracy, while INT4 and NVFP4 reduce it to near zero (Figure 1b). Recurrent states are updated and requantized at every decoding step. In GDN and KDA, each update applies learned decay to the previous state and adds a delta-rule correction along the current key direction (Yang et al., 2025; Team et al., 2025b). Quantization error therefore enters subsequent updates and propagates through the recurrence, as formalized in Section 3.3. Prior work has explored recurrent-state quantization in Mamba (Chiang et al., 2025a; Tianqi et al., 2025). To our knowledge, however, post-training quantization of recurrent states in GDN and KDA remains unexplored.

To understand this error, we examine the distribution of state magnitudes in GDN and KDA. Their magnitudes vary markedly across key channels and value dimensions (Figure 2a–b). After preprocessing along the value dimension, most quantization error is concentrated in a few key channels (Figure 2c). These errors are subsequently affected by the recurrent updates: learned decay scales the previous state, while the delta-rule correction partially erases information along the current key direction. Stronger decay suppresses more quantization error, while weaker decay allows more to persist. Although this decay vary across decoding steps, largely the same GDN heads and KDA key channels exhibit slow decay across prompts and tasks (Figure 2d–f).

Motivated by these observations, we propose DAMP, which uses offline calibration to estimate each key channel's accumulated-error risk from its quantization error and decay-based error retention. Under a fixed state-storage budget, it stores the highest-risk channels in higher precision and the remaining channels in the low-precision format. We evaluate DAMP on Qwen3.6-35B-A3B, Kimi-Linear-48B-A3B-Instruct (Team et al., 2025b), and Kimi-K3 (Team et al., 2026). At 9.9 bits per state value, DAMP remains close to the FP32 baseline across the evaluated mathematical reasoning, general reasoning, and code generation benchmarks. Relative to FP32-state inference, it reduces effective state storage by 69.1%, accelerates the recurrent-update operator by up to $2.59\times$, and reduces full-model time per output token (TPOT) by up to 19.0%. Our contributions are summarized as follows:

- To our knowledge, we are the first to study post-training quantization of recurrent states in GDN and KDA based language models. We evaluate floating-point and integer formats and show that 8-bit quantization already causes substantial accuracy loss, with 4-bit formats degrading accuracy further.
- We find magnitude concentration along both dimensions of GDN and KDA states. After value-dimension preprocessing, quantization error is concentrated in a few key channels. We further find that largely the same GDN heads and KDA key channels exhibit slow decay across prompts and tasks. DAMP combines quantization error and decay-based error retention to allocate precision under a fixed storage budget.
- Across the evaluated benchmarks on Qwen3.6-35B-A3B, Kimi-Linear-48B-A3B-Instruct, and Kimi-K3, DAMP remains close to the FP32 baseline at 9.9 bits per state value. In SGLang, it reduces recurrent-state storage by 69.1%, accelerates the recurrent-state update kernel by up to $2.59\times$, and lowers full-model TPOT by up to 19.0%.

2 RELATED WORK

Post-training quantization of LLMs. Post-training quantization compresses model weights, activations, and KV caches to reduce inference costs. Recent work improves quantization accuracy through learned rotations and scaling transformations (Liu et al., 2025; Hu et al., 2025) and mixed-precision allocation (Saxena et al., 2025). KV-cache quantization reduces the memory cost of long-context inference (Liu et al., 2024; Zandieh et al., 2026; Zhang et al., 2026).

Quantization and compression of recurrent states. Prior work on quantizing linear-attention models has mainly focused on the Mamba family. Quamba and MambaQuant use activation-range control and rotation-based transformations to improve weight and activation quantization (Chiang et al., 2025b; Yue et al., 2025). Quamba2 applies grouped quantization to cached recurrent states (Chiang et al., 2025a), while Q-Mamba uses separate scaling factors along the two state dimensions to reduce magnitude imbalance (Tianqi et al., 2025). DAMP instead quantizes the recurrent states of GDN and KDA, using quantization error and decay-based error retention to select high-precision key channels offline.

3 RECURRENT-STATE QUANTIZATION IN GDN AND KDA

3.1 GDN AND KDA STATE UPDATES

GDN and KDA store key–value associations in a recurrent state matrix. For each head, the recurrent state is a matrix $\boldsymbol{S}_t \in \mathbb{R}^{d_k \times d_v}$. We denote the row corresponding to key channel u by $\boldsymbol{S}_{t,u}$. At token step t, the query, key, and value vectors are $\boldsymbol{q}_t, \boldsymbol{k}_t \in \mathbb{R}^{d_k}$ and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$. Both architectures apply ℓ_2 normalization to queries and keys within each head. The query reads the updated state to produce

$$\boldsymbol{o}_t = \boldsymbol{S}_t^\top \boldsymbol{q}_t. \tag{1}$$

Both architectures apply learned decay to the previous state and then a delta-rule correction (Yang et al., 2025; Team et al., 2025b):

$$\boldsymbol{S}_t = \big(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\big)\boldsymbol{\Lambda}_t \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top. \tag{2}$$

Here, $\boldsymbol{\Lambda}_t$ applies learned decay to the previous state, with diagonal entries $a_{t,u}$. GDN uses a single decay factor $\alpha_t \in (0,1)$ shared across all key channels within a head, so $\boldsymbol{\Lambda}_t = \alpha_t \boldsymbol{I}$. KDA instead applies channel-wise decay, with $\boldsymbol{\Lambda}_t = \mathrm{diag}(\exp(\boldsymbol{g}_t))$, where $\boldsymbol{g}_t \in \mathbb{R}^{d_k}$. The factor $\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top$ partially erases the association with $\boldsymbol{k}_t$ from the decayed state, while $\beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$ writes the current value $\boldsymbol{v}_t$ along the same key direction. These two operations form the delta-rule correction, whose strength is controlled by $\beta_t \in (0,1)$.

We denote all operations applied to the previous state as the transition matrix,

$$\boldsymbol{A}_t = \big(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\big)\boldsymbol{\Lambda}_t, \tag{3}$$

Hence, Equation 2 can be written as

$$\boldsymbol{S}_t = \boldsymbol{A}_t \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top. \tag{4}$$

3.2 QUANTIZED STATE STORAGE

Our quantization targets the stored recurrent state $\boldsymbol{S}_t$ in Equation 2. During decoding, we reconstruct the previous state from quantized storage, perform the recurrent update, and quantize the updated state before writing it back to GPU memory. We use symmetric b-bit integer quantization. Let $\mathbf{x}$ denote a row of $\boldsymbol{S}_t$. The quantization scale is

$$s = \frac{\max(|\mathbf{x}|)}{2^{b-1} - 1}. \tag{5}$$

The reconstructed values after quantization are

$$Q(\mathbf{x}) = s \ \mathrm{round}\Big(\frac{\mathbf{x}}{s}\Big). \tag{6}$$

Specifications for all evaluated formats are provided in Appendix A.1.

3.3 ERROR PROPAGATION UNDER STATE QUANTIZATION

The quantization operation at each step introduces error, and those errors propagate over time due to recurrent updates. Let $\boldsymbol{S}_t^{\mathrm{q}}$ denote the state reconstructed from quantized storage at step t:

$$\boldsymbol{S}_t^{\mathrm{q}} = Q\big(\boldsymbol{A}_t \boldsymbol{S}_{t-1}^{\mathrm{q}} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top\big). \tag{7}$$

To measure accumulated quantization error up to step t, we define the **accumulated state error** as the difference between the reconstructed state $\boldsymbol{S}_t^{\mathrm{q}}$ and the unquantized state $\boldsymbol{S}_t$ computed using Equation 4 without state quantization:

$$\Delta \boldsymbol{S}_t = \boldsymbol{S}_t^{\mathrm{q}} - \boldsymbol{S}_t. \tag{8}$$

To measure the error brought by the quantization operation at time step t, we define **local quantization error**, which is the difference between the output and input of Q in Equation 7:

$$\boldsymbol{R}_t = \boldsymbol{S}_t^{\mathrm{q}} - \left(\boldsymbol{A}_t \boldsymbol{S}_{t-1}^{\mathrm{q}} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top\right), \tag{9}$$

Using Equation 4 and $\boldsymbol{S}_{t-1} = \boldsymbol{S}_{t-1}^{\mathrm{q}} - \Delta \boldsymbol{S}_{t-1}$ (derived from Equation 8), we rewrite the accumulated state error as

$$\begin{aligned}
\Delta \boldsymbol{S}_t &= \boldsymbol{S}_t^{\mathrm{q}} - \left(\boldsymbol{A}_t \left(\boldsymbol{S}_{t-1}^{\mathrm{q}} - \Delta \boldsymbol{S}_{t-1}\right) + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top\right) && (10)\\
&= \boldsymbol{A}_t \Delta \boldsymbol{S}_{t-1} + \boldsymbol{S}_t^{\mathrm{q}} - \left(\boldsymbol{A}_t \boldsymbol{S}_{t-1}^{\mathrm{q}} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top\right) && (11)\\
&= \boldsymbol{A}_t \Delta \boldsymbol{S}_{t-1} + \boldsymbol{R}_t. && (12)
\end{aligned}$$

Unrolling Equation 12 starting with $\Delta \boldsymbol{S}_0 = \boldsymbol{0}$, we have

$$\Delta \boldsymbol{S}_t = \boldsymbol{R}_t + \sum_{i=1}^{t-1} \left(\boldsymbol{A}_t \cdots \boldsymbol{A}_{i+1}\right) \boldsymbol{R}_i. \tag{13}$$

From Equation (13), the accumulated state error depends not only on the local quantization errors introduced at each step but also on how subsequent state transitions transform these errors. Both learned decay and the key-dependent factor in $\boldsymbol{A}_t$ affect this propagation.

4 Structure in Quantization Error and Decay

We examine how the distribution of state magnitudes in GDN and KDA affects local quantization error, and whether learned decay provides a stable signal of error retention.

4.1 State Structure and Error Concentration

Outliers in GDN and KDA states are concentrated in particular key channels and value dimensions (Figure 2a–b). These outliers can lead to substantial quantization error in smaller-magnitude entries that share the same quantization scale. We therefore quantize key channels independently, so that outliers in one channel do not affect the quantization scales of others.

Within each key channel, we divide state entries into groups and use the same quantization scale for all entries in a group. Before quantization, we apply scaling and reordering (SR) along the value dimension. We rescale each value dimension to reduce magnitude differences, then reorder the dimensions so that those with similar magnitudes share a quantization scale. Applying SR to INT8 quantization increases the AIME 2026 accuracy of Qwen3.6-35B from 18.48% to 63.98%, still well below the FP32 baseline of 85.46% (Table 1).

Figure 2c shows that this remaining error is highly concentrated across key channels. The 12.5% of key channels with the largest quantization errors account for approximately 69% of the total squared error in GDN and 76% in KDA. This motivates storing a small subset of key channels in higher precision.

Finding 1. Outliers in GDN and KDA states are concentrated in particular key channels and value dimensions. After value-dimension scaling and reordering, a small subset of key channels accounts for most of the quantization error.

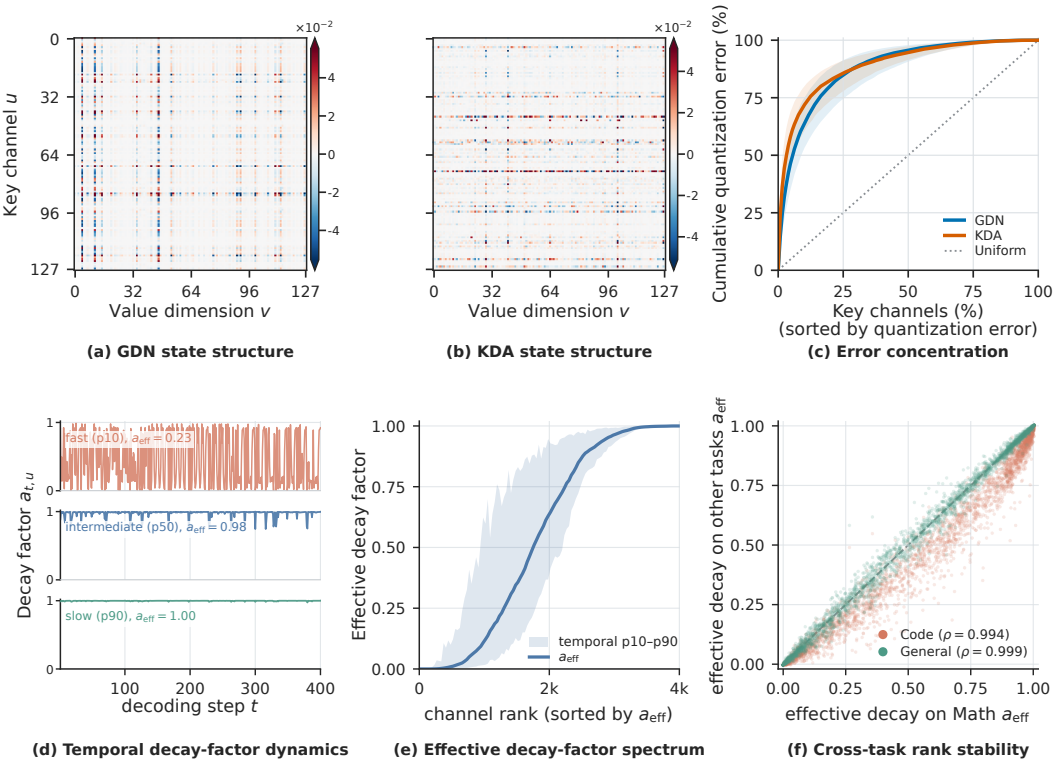


Figure 2: State structure, quantization error, and decay. (a–b) Outliers across key channels and value dimensions in GDN and KDA. (c) INT8 quantization error remains concentrated in a few key channels after value-dimension scaling and reordering. (d–e) KDA decay varies across steps and channels. (f) Channel rankings by effective decay factor remain stable across tasks.

4.2 Stable Structure in Learned Decay

Equation (13) shows that local quantization errors propagate through subsequent state transitions. Within these transitions, learned decay directly scales the error carried by the previous state. We find substantial differences in decay strength across GDN heads and KDA key channels. Figure 2d illustrates these differences using three KDA key channels with fast, intermediate, and slow decay. The decay factor $a_{t,u}$ fluctuates substantially in the fast-decaying channel, whereas it remains close to one in the slow-decaying channel.

Since decay factors multiply across steps, we define the effective decay factor for key channel u as their geometric mean, $a_{\text{eff},u} = \exp(\mathbb{E}_t[\log a_{t,u}])$, where $\mathbb{E}_t$ averages over decoding steps. An effective decay factor close to one indicates weak decay, which preserves much of the incoming state and its quantization error. Figure 2e shows a broad range of effective decay factors across KDA key channels, from strong suppression to near-complete retention under decay. GDN exhibits similarly broad variation across heads (Figure 6b).

We rank GDN heads and KDA key channels by their effective decay factors on Math, Code, and General. Although decay factors vary across decoding steps, largely the same heads and key channels exhibit slow decay across tasks. For KDA, the Spearman rank correlations with Math are 0.994 for Code and 0.999 for General (Figure 2f). GDN shows similarly stable rankings at the head level (Figure 6c). This cross-task stability supports offline estimation of decay-based error retention.

Finding 2. Decay strength varies substantially across GDN heads and KDA key channels. Although decay factors vary across decoding steps, largely the same heads and key channels exhibit slow decay across tasks.

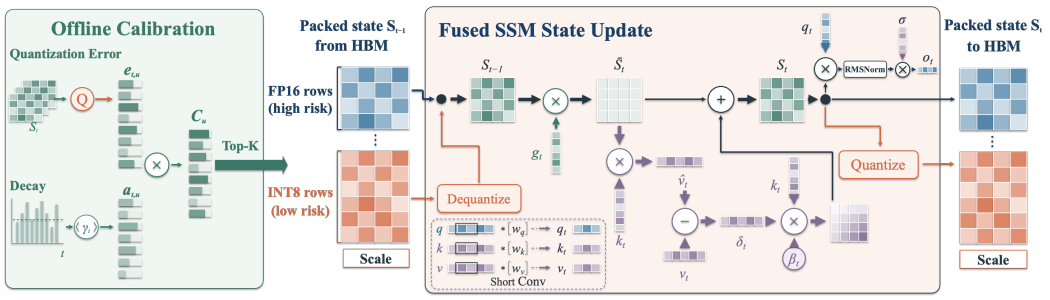


Figure 3: Overview of DAMP. Offline calibration selects high-precision key channels. A fused kernel updates the packed FP16/INT8 state, combining low-precision reconstruction, the recurrence, and requantization.

5 DAMP: DECAY-AWARE MIXED-PRECISION STATE QUANTIZATION

5.1 BUDGETED KEY-CHANNEL ALLOCATION

Quantization-error measurement. Our goal is to protect key channels whose quantization errors have the largest estimated cumulative impact under a limited high-precision budget. For each layer and head, we estimate local quantization error (Section 3.3) on sampled calibration states using the quantizer in Section 4.1. For key channel u, its squared magnitude is

$$e_{t,u} = \left\| [Q(\boldsymbol{S}_t) - \boldsymbol{S}_t]_u \right\|_2^2. \tag{14}$$

Accumulated-error risk. We estimate the contribution of each local quantization error to subsequent states under learned decay. Using the observed decay factors, the estimated squared-error contribution after j updates is $e_{t,u} \prod_{r=1}^{j} a_{t+r,u}^2$. We sum the initial squared error and these decay-weighted contributions over the next H updates, then average over calibration samples to define the accumulated-error risk score for key channel u:

$$C_u = \mathbb{E}_{\text{cal}} \left[e_{t,u} \left(1 + \sum_{j=1}^{H} \prod_{r=1}^{j} a_{t+r,u}^2 \right) \right]. \tag{15}$$

Here, $\mathbb{E}_{\text{cal}}$ averages over sampled calibration states with H subsequent updates available within the same trajectory. The score favors key channels with large local quantization errors and weak subsequent decay.

Budgeted selection. Let K_{hi} denote the number of key channels stored in high precision per head. We use the same count across layers and heads to maintain a uniform memory layout and simplify fused recurrent updates. For each layer ℓ and head h, we select the K_{hi} channels with the highest scores $C_{\ell,h,u}$, and denote their indices by $\mathcal{H}_{\ell,h}$. For a fixed channel count, this selection minimizes the total risk score of the channels left in low precision. The selected indices are determined independently for each layer and head and remain fixed throughout inference.

5.2 PACKED LAYOUT AND FUSED STATE UPDATE

We reorder the key channels so that the selected high-precision channels occupy the first K_{hi} rows of the state matrix, followed by the low-precision channels. Before inference, we rearrange the rows of the query and key projection weights to produce their outputs in this order. The value-dimension reordering used by SR is implemented similarly: we rearrange the rows of the value projection weights and the corresponding columns of the output projection weights. Convolution, gating, and normalization parameters are reordered to match. The model then produces recurrent inputs in the required channel order, eliminating explicit reordering during decoding.

We choose FP16 for the high-precision channels, as FP16 state achieves accuracy close to FP32 (Table 1). The packed state consists of contiguous FP16 and INT8 regions. At each decoding step,

Table 1: Main results on Qwen3.6-35B-A3B and Kimi-Linear-48B-A3B-Instruct. "Avg. bits" reports effective state-storage cost per element, including scales and zero points. SR denotes scaling and reordering along the value dimension. DAMP results are highlighted in **bold**.

		Math Reasoning			General Reasoning		Code
Method	**Avg. bits**	**AIME 2026**	**HMMT**	**IMO-Ans**	**GPQA-D**	**MMLU-Pro**	**LCB-v6**
Qwen3.6-35B-A3B							
FP32	32	85.46	57.57	50.29	81.97	84.66	86.95
FP16	16	84.58	55.82	49.42	82.13	84.61	86.80
BF16	16	79.71	54.97	43.71	81.66	84.65	84.10
FP8 (E4M3)	9.0	29.27	14.06	10.23	76.26	77.66	15.02
INT8	9.0	18.48	20.12	16.38	79.48	83.88	32.23
INT8+SR	9.0	63.98	50.09	39.31	81.89	84.59	64.51
NVFP4	4.5	0.03	0.05	0.98	16.57	9.31	1.02
INT4	6.0	0.07	0.14	1.03	28.69	32.19	0.85
INT4+SR	6.0	0.03	0.09	1.02	24.59	13.28	0.02
DAMP INT8	9.9	**83.82**	**55.11**	**48.89**	**82.05**	**84.57**	**85.67**
Kimi-Linear-48B-A3B-Instruct							
FP32	32	62.34	38.02	28.04	64.17	68.48	61.17
FP16	16	63.07	37.83	28.10	64.74	68.42	60.58
BF16	16	58.56	32.77	25.37	63.48	68.51	60.79
FP8 (E4M3)	9.0	29.47	13.16	11.20	54.26	56.90	22.16
INT8	9.0	49.90	27.08	22.21	63.22	68.37	57.97
INT8+SR	9.0	58.07	32.05	24.67	62.61	68.47	59.49
NVFP4	4.5	3.38	1.47	3.31	46.75	58.47	6.04
INT4	6.0	7.50	2.46	6.53	46.65	61.00	0.79
INT4+SR	6.0	0.52	0.47	2.42	40.02	55.68	2.49
DAMP INT8	9.9	**62.94**	**38.54**	**27.75**	**64.56**	**68.64**	**61.02**

a single CUDA kernel fuses state reconstruction, the recurrent update, and requantization before writing back the packed state (Figure 3). Implementation details are provided in Appendix A.4.

6 EXPERIMENTS

We evaluate DAMP in three stages. We first report downstream accuracy at the target storage budget, then measure recurrent-update and end-to-end serving efficiency, and finally ablate the key-channel selector and precision budget.

6.1 EXPERIMENTAL SETUP

Models. We evaluate Qwen3.6-35B-A3B (Qwen Team, 2026b) and Kimi-Linear-48B-A3B-Instruct (Team et al., 2025b), two hybrid MoE models with approximately 3B activated parameters. Qwen3.6 uses 30 GDN and 10 full-attention layers, while Kimi-Linear uses 20 KDA and 7 MLA layers. We also evaluate Kimi-K3 (Team et al., 2026), a 2.8T-parameter hybrid MoE model with 104B activated parameters, 69 KDA layers, and 24 gated MLA layers.

Benchmarks. Mathematical reasoning uses AIME 2026 Parts I and II and HMMT February 2026 (Dekoninck et al., 2026), as well as IMO-AnswerBench (Luong et al., 2025). General reasoning uses GPQA-Diamond and MMLU-Pro (Rein et al., 2024; Wang et al., 2024); code generation uses LiveCodeBench-v6 (Jain et al., 2025).

Baselines. We implement all methods in SGLang (Zheng et al., 2024), using its default FP32 recurrent-state storage as the reference. Baselines include FP16 and BF16 storage, FP8 and NVFP4

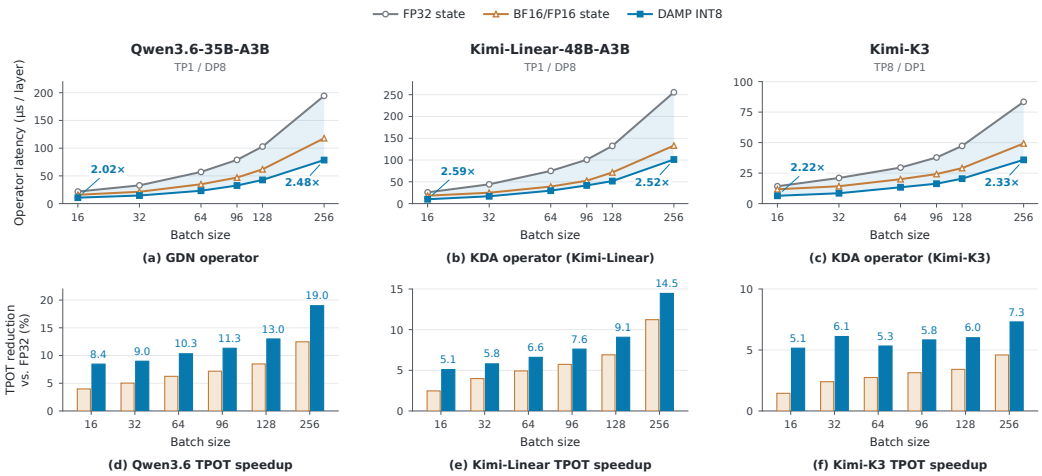


Figure 4: Decoding efficiency of DAMP-INT8, FP32, and BF16/FP16 state storage. (a–c) Recurrent-update latency per layer; (d–f) TPOT reduction relative to FP32. Columns show Qwen3.6-35B-A3B, Kimi-Linear-48B-A3B, and Kimi-K3, respectively.

quantization, and INT8/INT4 with and without SR. Quantizer specifications and generation settings are provided in Appendices A.1 and A.5, respectively.

Calibration. We calibrate each model using 36 prompts spanning mathematical reasoning, code generation, and general tasks. Each input contains 1,024 tokens, and the model generates 4,096 tokens using FP32 recurrent-state storage. We compute the channel scores in Equation (15) from quantization errors measured on sampled states and the observed decay factors over the next $H = 256$ updates after each sampled state. In the main experiments, DAMP stores $K_{\text{hi}} = 16$ key channels per head in FP16 and the remainder in INT8+SR, requiring 9.9 bits per state value. Data sources, splits, and state-sampling details are provided in Appendix A.2.

6.2 MAIN RESULTS

Table 1 compares DAMP with full-precision references and uniform state quantization. With only 0.9 additional bits per state value, DAMP recovers much of the accuracy lost under uniform INT8+SR. For example, DAMP improves AIME 2026 accuracy on Qwen3.6-35B by 19.84 percentage points over uniform INT8+SR. The additional evaluation on Kimi-K3 extends these results to a larger model: at 9.9 bits per state value, DAMP achieves accuracy close to FP32 on both AIME 2026 and HMMT (Table 2).

Table 2: Kimi-K3 accuracy (%).

Method	Avg. bits	AIME 2026	HMMT
FP32	32	96.88	97.92
BF16	16	97.29	96.78
FP8	9.0	59.83	40.91
INT8	9.0	89.17	82.58
INT8+SR	9.0	93.96	91.67
INT4	6.0	4.38	7.58
DAMP INT8	**9.9**	**97.50**	**97.35**

FP16 maintains accuracy close to FP32 on both Qwen3.6 and Kimi-Linear, while outperforming BF16 on average at the same 16-bit storage cost. This contrast suggests that these recurrent states benefit more from FP16's finer numerical resolution than from BF16's wider exponent range. Kimi-K3 is less sensitive to BF16 state storage, retaining accuracy close to FP32 on both AIME 2026 and HMMT. We speculate that quantization-aware training for BF16 recurrent-state storage may contribute to this robustness. With INT8+SR, accuracy losses vary substantially across tasks. On Qwen3.6-35B, accuracy remains within 0.1 percentage points of FP32 on GPQA-Diamond and MMLU-Pro, but drops by more than 20 points on AIME 2026 and LiveCodeBench-v6. INT4 and NVFP4 cause severe accuracy degradation in mathematical reasoning and code generation, making the evaluated 4-bit configurations impractical for these tasks.

6.3 EFFICIENCY

We compare DAMP with FP32 and 16-bit (BF16/FP16) state storage in SGLang at decode batch sizes from 16 to 256. Qwen3.6 and Kimi-Linear use tensor parallelism (TP) of 1 and data parallelism (DP) of 8, while Kimi-K3 uses TP8/DP1. Requests are sampled from the ShareGPT-V3 conversation dataset[1], with input and output lengths fixed at 256 and 64 tokens, respectively. We report per-layer recurrent-update latency, including state loads, dequantization, quantization, and writeback, together with full-model time per output token (TPOT).

Figure 4 shows that DAMP reduces recurrent-update latency across all three models, outperforming both FP32 and 16-bit state storage and achieving up to a $2.59\times$ speedup over FP32. These gains translate into faster full-model decoding: at batch size 256, DAMP reduces TPOT relative to FP32 by 19.0% on Qwen3.6, 14.5% on Kimi-Linear, and 7.3% on Kimi-K3. The smaller TPOT reduction on Kimi-K3 may partly reflect the inter-device communication overhead of its TP8 configuration.

Multi-turn serving. Beyond decoding latency, we evaluate time to first token (TTFT) and prefix-cache reuse on Kimi-K3 under a fixed total memory budget for recurrent states and the KV cache. The workload consists of 256 four-turn conversations from WildChat-1M (Zhao et al., 2024). Compared with FP32 and BF16, DAMP reduces mean TTFT by 20.7% and 14.5%, respectively, while achieving higher prefix-cache hit rates (Table 3). Workload details are provided in Appendix E.

Table 3: Multi-turn TTFT and cache hit rates on Kimi-K3.

State storage	TTFT (s) ↓		Mean cache hit rate (%) ↑
	Mean	P95	
FP32	38.427	72.471	47.20
BF16	35.646	69.500	56.96
DAMP INT8	**30.464**	**61.559**	**65.29**

6.4 ABLATION STUDIES

Key-channel selector. We compare the criteria used to select high-precision key channels under the same INT8+SR quantizer and storage budget (Table 4). Random selects channels uniformly, while State energy ranks them by their mean squared norm. Error-only selection ranks channels by their average local quantization error, while retention-only selection ranks them by their average decay-based error retention weight. DAMP achieves the highest accuracy on all three benchmarks, which supports incorporating error retention into channel selection rather than relying on local quantization error alone. Additional GDN results are provided in Appendix D.

Precision budget. Figure 5 shows AIME 2026 accuracy as we vary the number of key channels retained in FP16. Increasing the FP16 budget generally improves accuracy, but provides limited additional gains beyond $K_{\text{hi}} = 16$. These results suggest that protecting a small subset of key channels can recover most of the accuracy lost to INT8+SR quantization. We therefore use $K_{\text{hi}} = 16$ in the main experiments.

Table 4: Matched-budget KDA selector ablation at 9.9 bits per state value.

Selector	AIME	GPQA	LCB
Kimi-Linear (KDA)			
Random	59.06	62.12	58.68
State eng.	60.36	63.38	59.67
Error-only	61.40	63.85	60.34
Retention-only	61.22	63.69	60.69
DAMP (C)	**62.94**	**64.56**	**61.02**

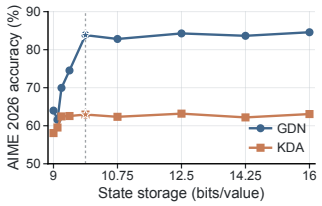


Figure 5: AIME 2026 accuracy versus bits per state value.

7 CONCLUSION

We introduced DAMP, a post-training method for compressing the recurrent states of GDN- and KDA-based language models. It ranks key channels using quantization error and decay-based error retention estimated during calibration, then realizes the allocation as a static mixed-precision layout. At 9.9 bits per state value, DAMP maintains average accuracy close to FP32 across the three evaluated checkpoints. Relative to FP32-state inference, it reduces state storage by 69.1%, accelerates the recurrent-state update kernel by up to $2.59\times$, and lowers full-model TPOT by up to 19.0%.

[1] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

AI USE STATEMENT

We used AI for language editing, code debugging, and figure and table prototyping. The authors reviewed and revised all AI-assisted text. All measurements reported in this paper were produced by the evaluation procedures described in the paper. The authors take full responsibility for the final content, claims, and research artifacts.

ETHICS STATEMENT

This work studies an inference-time compression method using publicly released model checkpoints, calibration corpora, and established evaluation benchmarks. It does not involve human participants or the collection of private data. The method changes the storage and execution of recurrent states rather than the intended capabilities of the underlying models; consequently, deployments retain the limitations and potential risks of those models and their training data.

REPRODUCIBILITY STATEMENT

Section 6.1 describes the evaluated models, benchmarks, baselines, and calibration setup. Appendices A.1–A.4 describe the quantization formats, calibration procedure, key-channel selection algorithm, and implementation. Generation settings and the number of generations per problem are provided in Appendix A.5. Section 6.3 and Appendix E describe the serving evaluation setup.

REFERENCES

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 *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 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 *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=mnna9LUg7P.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with LLMs. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026. URL https://openreview.net/forum?id=DmPE4byHuN.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li,

Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021a. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/c24cd76e1ce41366a4bbe8a49b02a028-Paper-round2.pdf.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Xing Hu, Yuan Cheng, Zhixuan Chen, Dawei Yang, Jiangyong Yu, XUCHEN, Zhihang Yuan, Zhe jiang, and Sifan Zhou. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 37492–37517, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5cebc89b113920dbff7c79854ba765a3-Paper-Conference.pdf.

Naman Jain, Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/94074dd5a072d28ff75a76dabed43767-Paper-Conference.pdf.

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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e5b1c0d4866f72393c522c8a00eed4eb-Paper-Conference.pdf.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. Towards robust mathematical reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of*

the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 35418–35442, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1794. URL https://aclanthology.org/2025.emnlp-main.1794/.

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

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026c. URL https://qwen.ai/blog?id=qwen3.8.

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

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

Utkarsh Saxena, Sayeh Sharify, Kaushik Roy, and Xin Wang. Resq: mixed-precision quantization of large language models with low-rank residuals. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Weixin Xu, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, Zonghan Yang, and Zongyu Lin. Kimi k1.5: Scaling reinforcement learning with llms, 2025a. URL https://arxiv.org/abs/2501.12599.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025b. URL https://arxiv.org/abs/2510.26692.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen

Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Chen Tianqi, Yuanteng Chen, Peisong Wang, Weixiang Xu, Zeyu Zhu, and Jian Cheng. Q-mamba: Towards more efficient mamba models via post-training quantization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10594–10610, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.551. URL https://aclanthology.org/2025.findings-acl.551/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task

language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4904fad153f6434a7bcf04465d4be2cc-Paper-Conference.pdf.

Yuxuan Yue, Xing Hu, Dawei Yang, Zhihang Yuan, Zixu Jiang, Zhixuan Chen, Jiangyong Yu, XUCHEN, and Sifan Zhou. Mambaquant: Quantizing the mamba family with variance aligned rotation methods. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 33231–33250, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/51ba8a68f471d952af625d1faf55e6c6-Paper-Conference.pdf.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tO3ASKZlok.

Tao Zhang, Ziqian Zeng, Hao Peng, Huiping Zhuang, and Cen Chen. MixKVQ: Query-aware mixed-precision KV cache quantization for long-context reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7189–7204, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.326. URL https://aclanthology.org/2026.acl-long.326/.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 34590–34605, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9421261e06f1a63a352b068f1ac90609-Paper-Conference.pdf.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

A Additional Method Details

A.1 Storage Quantizers

All groupwise quantizers partition the value dimension of each key channel into contiguous blocks. Group quantization parameters are recomputed whenever the state is written.

Integer formats. INT8 uses the symmetric mapping in Equation 6 with groups of 32 values. Each group stores one FP32 scale and no zero point. INT4 uses asymmetric quantization with the same group size, storing one FP32 scale s and one FP32 zero point z per group.

Value scaling and reordering. We rescale each value dimension to reduce magnitude differences. The scaling factors are calibrated separately for each layer and head. We then sort the value dimensions by their RMS magnitudes after scaling, placing dimensions with similar magnitudes next to each other. Each low-precision key channel is quantized in groups of 32 consecutive entries, with one quantization scale per group. The scaling factors and value order are selected during offline calibration and remain fixed during inference. In contrast, group quantization parameters are recomputed at each state write.

Floating-point formats. FP8 uses E4M3 without SR, with one FP32 scale per group of 32 state values and no zero point. The scale is computed as the group's maximum absolute value divided by 448. Values are divided by this scale before conversion to E4M3 and multiplied by it during reconstruction. NVFP4 uses blocks of 16 E2M1 values, one E4M3 scale per block, and one FP32 global scale.

A.2 Calibration Protocol

Data and splits. All three models use 36 prompts: 12 from the MATH training split (Hendrycks et al., 2021b), 12 from the APPS training split (Hendrycks et al., 2021a), and 12 from the ARC-Challenge training split (Clark et al., 2018). We divide these prompts into disjoint fitting, parameter-selection, and validation splits of 12 each. Every split contains four mathematical, four code, and four general prompts.

Reference trajectories. Each input contains 1,024 tokens, including the chat template and special tokens. The model generates 4,096 tokens with FP32 recurrent-state storage; early termination is suppressed to maintain the fixed output length. We record states and decay factors over two windows per sequence, starting at decoding-update positions 0 and 2,048 and covering 1,024 updates each. The states after the 256th, 512th, and 768th updates within each window serve as scoring anchors. For each anchor, the following $H = 256$ updates provide a complete decay sequence for Equation 15. This gives six anchors per sequence and 72 per split.

Scaling and channel selection. For each layer and head, we use state RMS statistics from the fitting split to construct candidate value-scaling factors and value orders (Appendix A.1). For each candidate c, we simulate INT8 quantization at the fitting anchors and restore the original value scales and channel order before measuring $e_{t,u}$ in Equation 14. We compute C_u using the next 256 observed decay factors as in Equation 15 and select the K_{hi} highest-scoring key channels to form the FP16 set $\mathcal{H}_c$.

We then freeze $\mathcal{H}_c$ and evaluate the complete mixed-precision configuration on the parameter-selection split:

$$J(c) = \mathbb{E}_{\mathrm{tune}}\left[\sum_{u \notin \mathcal{H}_c} W_{t,u}\, e_{t,u}^{(8,c)} + \sum_{u \in \mathcal{H}_c} W_{t,u}\, e_{t,u}^{(16,c)}\right]. \tag{16}$$

Here, $e_{t,u}^{(8,c)}$ and $e_{t,u}^{(16,c)}$ are the squared errors after INT8 quantization and FP16 storage, respectively, both measured in the original state coordinates. $W_{t,u}$ is the decay-based error retention weight in Equation 15, including the initial contribution with weight one, and $\mathbb{E}_{\mathrm{tune}}$ averages over the parameter-selection split. All calibration averages give equal weight to anchors within a window, windows within a sequence, sequences within a domain, and the three domains. We choose the

Algorithm 1: Offline calibration and layout selection

Input: fitting and parameter-selection trajectories with scoring anchors and per-update decay factors; $H = 256$; high-precision key-channel count K_{hi}
Output: SR configurations, protected key-channel sets, and key permutations
foreach *recurrent layer* ℓ *and head* h **do**
  Construct SR candidates from state RMS statistics on the fitting split;
  foreach *candidate* c **do**
    Compute $C^{(c)}_{\ell,h,:}$ on fitting anchors using Equations 14 and 15;
    $\mathcal{H}_c \leftarrow \mathrm{TopK}(C^{(c)}_{\ell,h,:}, K_{\mathrm{hi}})$;
    Evaluate $J(c)$ on the parameter-selection split with $\mathcal{H}_c$ fixed, using Equation 16;
  $c^*_{\ell,h} \leftarrow \arg\min_c J(c)$;
  $\mathcal{H}_{\ell,h} \leftarrow \mathcal{H}_{c^*_{\ell,h}}$;
  Construct $\pi_{\ell,h}$ so that the selected channels occupy the first K_{hi} state rows, preserving channel order within each precision region;
return $\{c^*_{\ell,h}, \mathcal{H}_{\ell,h}, \pi_{\ell,h}\}_{\ell,h}$;

candidate with the smallest $J(c)$ and retain its fitted protected set and key permutation (Algorithm 1). The independent validation split evaluates the frozen configuration.

Calibration cost. Calibration takes approximately 7 minutes for Qwen3.6-35B, 9 minutes for Kimi-Linear-48B, and 78.75 minutes for Kimi-K3.

A.3 State Scaling and Reordering

Algorithm 1 constructs a key-channel permutation that places the selected high-precision channels in the first K_{hi} rows of the state matrix. Along the value dimension, SR applies the scaling and reordering described in Appendix A.1. To update the state in this representation, the recurrent inputs must use the same channel orders and value scales.

For one head, let $\mathbf{P}_K$ and $\mathbf{P}_V$ denote the fixed permutation matrices for key channels and value dimensions, respectively. Let $\mathbf{D}_c = \mathrm{diag}(c_1, \ldots, c_{d_v})$, with $c_v > 0$, contain the value-scaling factors. The transformed state $\boldsymbol{T}_t$ and recurrent inputs are

$$\begin{aligned} \boldsymbol{T}_t &= \mathbf{P}_K \boldsymbol{S}_t \mathbf{D}_c^{-1} \mathbf{P}_V^\top, \\ \boldsymbol{q}'_t &= \mathbf{P}_K \boldsymbol{q}_t, \qquad \boldsymbol{k}'_t = \mathbf{P}_K \boldsymbol{k}_t, \qquad \boldsymbol{v}'_t = \mathbf{P}_V \mathbf{D}_c^{-1} \boldsymbol{v}_t. \end{aligned} \tag{17}$$

The decay factors follow the same key-channel order, giving $\boldsymbol{\Lambda}'_t = \mathbf{P}_K \boldsymbol{\Lambda}_t \mathbf{P}_K^\top$ and $\boldsymbol{A}'_t = \mathbf{P}_K \boldsymbol{A}_t \mathbf{P}_K^\top$. The scaling factors in the reordered value coordinates form $\mathbf{D}'_c = \mathbf{P}_V \mathbf{D}_c \mathbf{P}_V^\top$. Substituting these transformations into Equations 1 and 2 gives the transformed update and readout:

$$\begin{aligned} \boldsymbol{T}_t &= \boldsymbol{A}'_t \boldsymbol{T}_{t-1} + \beta_t \boldsymbol{k}'_t \boldsymbol{v}'^\top_t, \\ \boldsymbol{o}'_t &= \mathbf{D}'_c \boldsymbol{T}_t^\top \boldsymbol{q}'_t = \mathbf{P}_V \boldsymbol{o}_t. \end{aligned} \tag{18}$$

Multiplication by $\mathbf{D}'_c$ restores the numerical scale while keeping the value dimensions in the reordered coordinates. Normalization and output-gating parameters follow the same value order. For this head, the output-projection weights become $\mathbf{W}'_O = \mathbf{W}_O \mathbf{P}_V^\top$, absorbing the inverse value permutation without an explicit reordering operation. With the initial state transformed accordingly, the block is functionally equivalent to the original in the absence of quantization.

The fixed channel permutations are absorbed into the projection weights and associated channel-wise parameters before inference, so the recurrent state does not need to be reordered at each decoding step. Value scaling and scale restoration are fused with the recurrent update and readout.

A.4 Packed Layout and Fused Update

State storage. The cache stores the transformed state $\boldsymbol{T}_t$ from Equation 17 in two contiguous regions: FP16 values for the K_{hi} selected key channels and INT8 codes for the remaining channels.

Each group of 32 INT8 values along the value dimension has one FP32 quantization scale stored in the cache. The contiguous regions support vectorized loads and stores.

Fused decoding. At each decoding step, a single CUDA kernel loads the packed state and dequantizes its INT8 codes. It evaluates the transformed update and readout in Equation 18 using FP32 arithmetic, then requantizes the low-precision channels with recomputed group scales. The kernel writes back FP16 values, INT8 codes, and FP32 scales. State tiles are reused in registers across these stages, reducing intermediate memory traffic.

Prefill path. Prefill uses FP32 arithmetic within each chunk and writes its final state to the same packed cache. Subsequent prefill chunks and decoding steps read this compressed state.

A.5 Evaluation Settings

Generation parameters. For downstream accuracy evaluation, Qwen3.6-35B and Kimi-Linear-48B run in thinking mode. Qwen3.6 uses temperature 0.6, top-$p = 0.95$, top-$k = 20$, and at most $65{,}536$ output tokens. Kimi-Linear uses temperature 1.0, top-$p = 1.0$, no top-k truncation, and at most $262{,}144$ output tokens. Kimi-K3 uses temperature 1.0, top-$p = 0.95$, zero presence and frequency penalties, and at most $262{,}144$ output tokens.

Table 5 lists the number of responses generated for each problem on each benchmark.

Table 5: Number of generations per problem for each model and benchmark.

Benchmark	Qwen3.6-35B	Kimi-Linear-48B	Kimi-K3
AIME 2026	128	128	16
HMMT February 2026	64	64	8
IMO-AnswerBench	16	16	–
GPQA-Diamond	32	32	–
MMLU-Pro	16	16	–
LiveCodeBench-v6	8	8	–

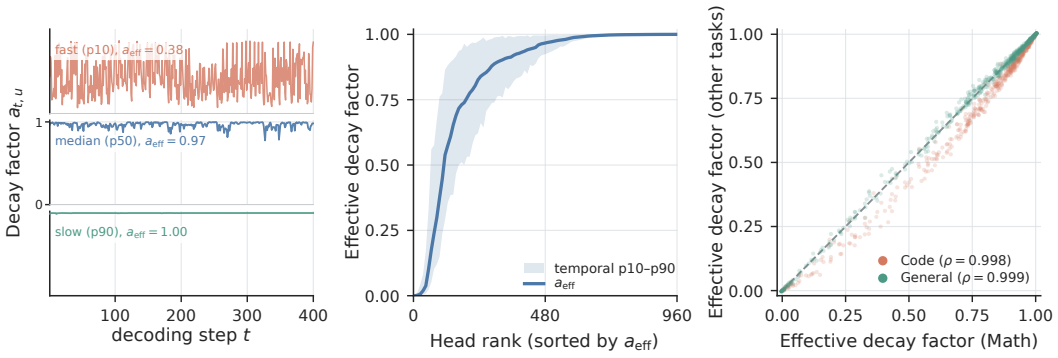
B Long-Context Evaluation

We evaluate long-context accuracy on RULER (Hsieh et al., 2024) at context lengths from 4K to 128K tokens, comparing FP32, FP16, uniform INT8+SR, and DAMP state storage (Table 6). Each format is used throughout prefill and decoding.

Table 6: RULER macro accuracy (%) across context lengths.

State format	4K	8K	16K	32K	64K	128K
Qwen3.6-35B-A3B (GDN)						
FP32 state	96.96	96.94	96.65	96.56	96.32	96.13
FP16 state	96.98	96.94	96.65	96.56	96.32	96.11
INT8+SR	96.93	96.89	96.60	96.43	96.28	96.07
DAMP INT8	**97.00**	**96.94**	**96.65**	**96.56**	**96.32**	**96.10**
Kimi-Linear-48B-A3B-Instruct (KDA)						
FP32 state	95.27	94.44	93.53	92.37	91.44	88.02
FP16 state	95.27	94.44	93.53	92.37	91.44	88.03
INT8+SR	95.27	94.35	93.41	92.24	91.36	87.87
DAMP INT8	**95.27**	**94.42**	**93.53**	**92.36**	**91.44**	**88.02**

Both INT8+SR and DAMP remain close to FP32 across the evaluated context lengths on RULER. The largest absolute accuracy differences between DAMP and FP32 are 0.04 percentage points on Qwen3.6 and 0.02 on Kimi-Linear, compared with 0.13 and 0.15 points, respectively, for uniform INT8+SR.



(a) Temporal decay-factor dynamics (b) Effective decay-factor spectrum (c) Cross-task rank stability

Figure 6: Head-level decay structure in Qwen3.6-35B (GDN). (a) Decay factors across decoding steps for heads at p10, p50, and p90 of a_{eff}. (b) Effective decay factors across all 960 heads; shading shows the p10–p90 range across decoding steps. (c) Effective decay factors on Code and General compared with Math, with Spearman rank correlations.

C Stability of Offline Signals

C.1 Calibration-Split Agreement

We assess sensitivity to the calibration data by splitting the corpus into two disjoint, domain-balanced subsets and calibrating them separately using the score in Equation 15 with $H = 256$. For each layer and head, we compare the selected top-16 key channels and the complete channel rankings. The overlap between selected sets is $|\mathcal{H}_1 \cap \mathcal{H}_2|/16$. Averaged across layers and heads, it reaches 92.0% for KDA and 93.2% for GDN, compared with 12.5% expected for random selection (Table 7). The complete rankings have mean Spearman correlations of 0.990 and 0.986, respectively.

Table 7: Agreement between calibration splits, averaged across layers and heads. Common is the number of shared top-16 channels.

Arch.	Common	Overlap	Spearman
KDA	14.72	92.0%	0.990
GDN	14.91	93.2%	0.986
Random	2.00	12.5%	–

C.2 GDN Decay Structure

Figure 6 summarizes decay statistics for all 960 heads in the 30 GDN layers of Qwen3.6-35B. Since GDN shares a decay factor across all key channels within a head, we characterize each head by its effective decay factor, $a_{\text{eff},\ell,h} = \exp(\mathbb{E}_t[\log a_{t,\ell,h}])$. The effective decay factors at the 10th, 50th, and 90th percentiles are 0.376, 0.967, and 0.9996, respectively, indicating substantial differences in decay strength across heads. Head rankings by effective decay factor are also highly consistent across tasks: the rankings on Code and General have Spearman rank correlations of 0.998 and 0.999, respectively, with those on Math.

D Additional GDN Ablations

Key-channel selection. Table 8 compares selectors using the same INT8+SR quantizer and high-precision budget. The Retention-only score is identical across key channels within a GDN head and cannot distinguish them. In DAMP, however, the shared decay weight can vary across steps and weights each channel's local errors before averaging. Thus, DAMP need not produce the same ranking as the error-only selector.

Table 8: Matched-budget GDN selector ablation at 9.9 bits per state value.

Selector	AIME 2026	GPQA-D	LCB-v6
Random	66.59	80.98	66.88
State energy	82.19	81.37	85.17
DAMP (C)	**83.82**	**82.05**	**85.67**

E MULTI-TURN SERVING ON KIMI-K3

Setting. Each configuration processes 256 four-turn WildChat-1M conversations, yielding 1,024 requests. Output lengths match the corresponding dataset responses, capped at 1,024 tokens, and each generated response is included in the next turn's input. Prefix caching is enabled in SGLang. DAMP uses the main configuration with $K_{\mathrm{hi}} = 16$ and FP32 group scales. Table 3 reports TTFT and the mean token-based cache hit rate across all four turns.

F LIMITATIONS

Our evaluation covers three released GDN- and KDA-based checkpoints in SGLang. Accuracy and efficiency gains may differ on other model architectures, hardware, and serving configurations. Although DAMP-INT8 remains close to FP32, variants with INT4 or NVFP4 as the low-precision format do not yet recover FP32 accuracy. Future work includes incorporating richer recurrent dynamics into channel selection, improving 4-bit state quantization, and evaluating these methods on additional model families and serving systems.