

SampKV: Saliency-Aware Mixed-Precision KV Cache for Efficient Diffusion Language Model Inference

Zhenrui Wang¹ Haodong Chang¹ Yang Sui^{3*} Rongjian Liang⁴
Bo Yuan² Jiang Hu^{1,5†}

¹Texas A&M University ²Rutgers University

³Microsoft AI ⁴NVIDIA ⁵Technical University of Munich

{jerrywang95, jianghu}@tamu.edu

Abstract

Diffusion language models (dLLMs) have recently shown strong potential in various tasks with competitive decoding efficiency, but their bidirectional attention incurs quadratic compute and memory complexity. Many prior works rely on KV cache optimizations to accelerate inference. However, the full-sequence BF16 KV cache remains the dominant memory bottleneck. We propose SampKV, a **Saliency-aware Mixed-Precision KV** cache for dLLMs that stores most history in 2 bits while promoting a small budget of attention-salient tokens to high precision. A customized Triton kernel operates directly on the mixed-precision cache and avoids materializing full-precision KV tensors, alleviating the storage and traffic bottleneck. Experiments on LLaDA and Dream models show that the memory bottleneck is substantially alleviated: up to 54% peak memory reduction at matched workload, or up to 2.06× throughput improvement at matched peak memory compared to the BF16 baseline, while maintaining the accuracy.

1 Introduction

Diffusion language models (dLLMs) such as LLaDA and Dream (Nie et al., 2026; Ye et al., 2025) have recently shown strong potential on a range of reasoning and coding tasks while supporting parallel decoding, making them a competitive alternative to autoregressive (AR) decoding. Unlike AR decoding, where the KV states grow linearly and can be cached incrementally, bidirectional attention forces the model to *refresh* the historical KV state across the full sequence as denoising progresses, so both compute and memory scale quadratically with sequence length. Empirically, these quadratic costs make dLLMs severely

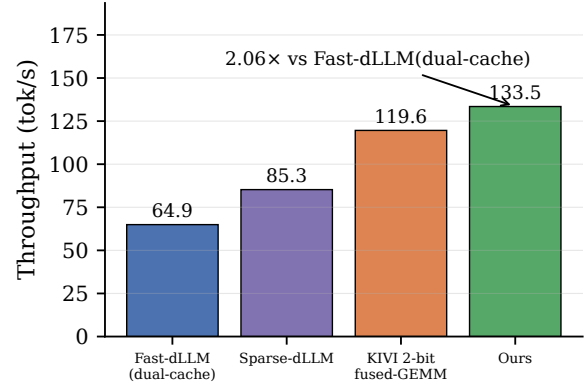


Figure 1: **Matched-memory throughput on LLaDA-8B-Instruct.** At a fixed ~ 34 GiB peak GPU memory (prompt 1024, A100-80 GB), our 2-bit cache with a high-precision overlay served by the fused mixed-precision attention kernel delivers 2.06× the Fast-dLLM (BF16 dual-cache) throughput, ahead of both the Sparse-dLLM and the KIVI 2-bit fused-GEMM at the same memory footprint.

memory-bound and drive them into out-of-memory (OOM) at modest batch size and sequence length on commodity GPUs, with throughput trailing well behind their AR counterparts.

Existing work on dLLM KV cache optimization largely follows two directions. (1) **Selective refresh and reuse** methods (Wu et al., 2026; Hu et al., 2026; Ma et al., 2026; Jiang et al., 2026; Liu et al., 2025; Nguyen-Tri et al., 2026; Cheong et al., 2026) decide when and where to recompute or reuse cached states across denoising sub-steps. They reduce repeated full-sequence refreshes, but the resident cache is still stored in high precision, so peak memory remains dominated by KV storage. (2) **Sparse attention or cache eviction** methods (Kwon et al., 2026; Wang et al., 2026b; Xi et al., 2026; Long et al., 2026; Song et al., 2026; Huang et al., 2026) route queries to a selected subset, or prune and evict low-importance entries. They reduce per-step attention computation, but they either

*Project advisor.

†Corresponding author.

We publish our code at: <https://github.com/jerryw95/SampKV>

keep the original cache intact or rely on expensive refresh to recover the evicted caches, which spikes memory back to the dense baseline. As a result, prior work leaves the peak-memory bottleneck largely unresolved.

To target this memory bottleneck, we propose SampKV (Saliency-aware Mixed-Precision KV cache), which replaces the binary “evict or refresh” choice with a saliency-aware mixed-precision KV cache. Most historical KV stays at 2 bits and still contributes to attention rather than being discarded. The accuracy gap caused by low-precision quantization is then closed by promoting a small, block-conditioned, saliency-selected subset to a high-precision overlay that is refreshed at subsequent steps. To prevent the full-sequence refresh from spiking memory back to the unquantized baseline, each layer’s KV is quantized immediately after refresh, so the largest memory footprint ever resident in HBM during a refresh is a single layer’s KV states rather than the entire stack. A fused Triton kernel then serves the mixed-precision cache with tiled online softmax: it consumes packed 2-bit KV tiles and the high-precision overlay in one pass, avoids materializing the dense attention-score matrix in HBM, and keeps only tile-local intermediates plus the online softmax state on chip.

Contributions.

- **2-bit KV cache quantization for diffusion LLMs.** We demonstrate that 2-bit KV cache quantization effectively alleviates the dLLM peak-memory bottleneck, with up to 54% peak-memory reduction compared with BF16 baseline on matched workload.
- **A saliency-aware mixed-precision cache mechanism.** A saliency-aware high-precision overlay is proposed to recover accuracy to match the BF16 baseline. The saliency signal uses the current block-query’s attention against prefix keys, measured by a once-per-block probe forward that materializes post-softmax scores (Section 3.3).
- **A fused mixed-precision attention Triton kernel.** We implemented a Triton (Tillet et al., 2019) kernel which streams 2-bit KV together with the high-precision overlay under tiled online softmax, avoiding writing dequantized full-precision KV tensors to HBM while keeping only small intermediate states on chip (Section 3.4).

On LLaDA-8B-Instruct, SampKV achieves up to 54% **peak-memory reduction at matched workload** and up to $2.06\times$ **throughput** at matched peak

memory (Figure 1), while restoring accuracy to the BF16 baseline (Wu et al., 2026) across all four reasoning and coding benchmarks (Table 1).

2 Related Work

Diffusion-based large language models. Diffusion models, established for continuous image and video synthesis, were extended to text through *discrete* diffusion (Austin et al., 2021a; Lou et al., 2024), with masked (absorbing-state) formulations (Sahoo et al., 2024; Shi et al., 2024) becoming dominant. Scaling this recipe up (Nie et al., 2025) produced large-scale dLLMs such as LLaDA (Nie et al., 2026), Dream (Ye et al., 2025), and their successors (Bie et al., 2025, 2026), which match auto-regressive (AR) models on reasoning and coding tasks while decoding multiple tokens in parallel within a denoising window. Block-Diffusion (Arriola et al., 2025) and Discrete Diffusion Forcing (Wang et al., 2026a) further add semi-autoregressive structure across blocks. The bidirectional attention mechanism recomputes the historical KV state across the full sequence as denoising progresses, leaving inference both compute- and memory-bound.

KV cache optimization for dLLM. To overcome the KV-state bottleneck, many recent works have focused on cache optimization for dLLMs. *Selective refresh and reuse* methods reuse historical KV or intermediate states across denoising substeps and recompute only stale entries (Wu et al., 2026; Hu et al., 2026; Ma et al., 2026; Liu et al., 2025; Cheong et al., 2026; Jiang et al., 2026), cutting per-step compute but leaving peak memory largely untouched. *Sparse attention or cache eviction* methods instead drop, prune, or route around low-importance entries (Kwon et al., 2026; Wang et al., 2026b; Xi et al., 2026; Long et al., 2026; Song et al., 2026; Huang et al., 2026). They reduce per-step attention computation, but they either keep the original cache intact or rely on expensive refresh to recover the evicted caches, which spikes memory back to the dense baseline. Orthogonal token-level compute-skipping systems such as FOCUS and DyLLM (Liang et al., 2026; Lee et al., 2026) reduce computation for selected tokens, but do not directly compress the historical KV cache. dLLM sink studies further show that high-attention positions shift across denoising steps and are not always high-value (Rulli et al., 2026; Zhang et al., 2026; Myrzakhan et al., 2026), mo-

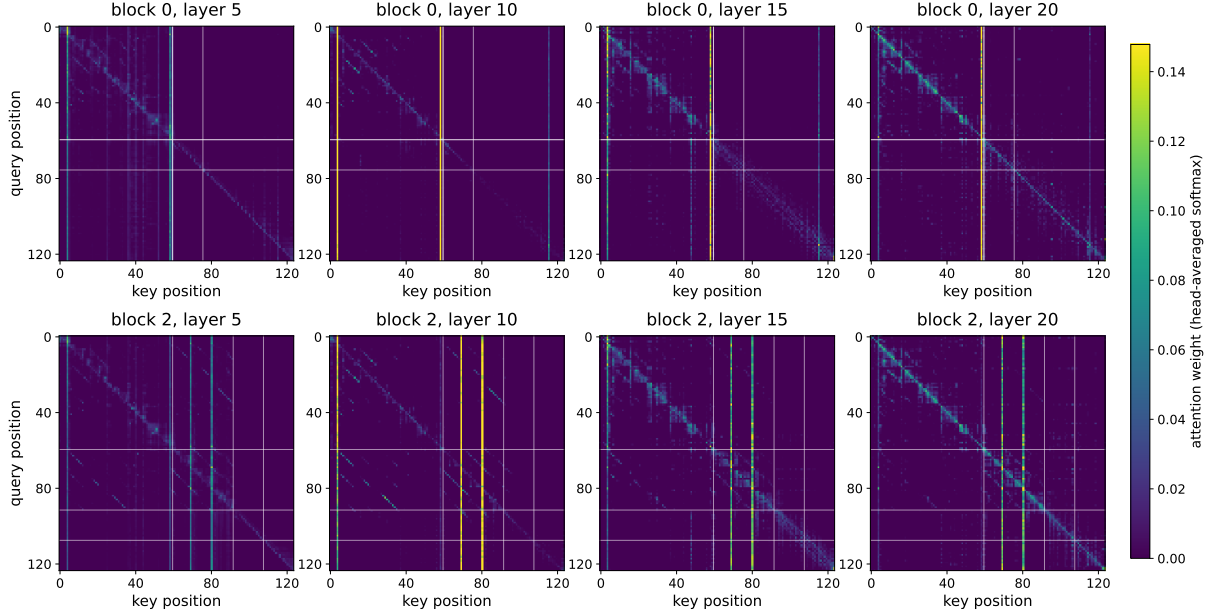


Figure 2: **Attention maps across decoding blocks and layers on LLaDA-8B-Instruct.** Rows: attention map at the start of block 0 and block 2; columns: layers 5, 10, 15, 20. Diagonal stripes show strong attention to neighboring tokens near the current-block boundary, motivating boundary protection. Bright vertical stripes mark a small number of salient keys that sit *inside* the sequence and drift across blocks, motivating per-block saliency selection rather than a static initial-token protection rule.

tivating a *block-conditioned* saliency signal rather than a static first- N rule. Our method instead keeps non-salient tokens in cache at 2-bit precision, so a missed saliency call costs quantization error rather than an irreversible drop (“soft eviction”).

KV quantization and mixed precision. KV quantization compresses the cache rather than pruning it. KIVI (Liu et al., 2024) established a widely used 2-bit baseline (per-channel keys, per-token values, asymmetric scaling), with later work adding outlier handling (Hooper et al., 2024; Su et al., 2025), sink-aware precision (Xiao et al., 2024; Su and Yuan, 2025), layer-wise mixed precision (Li et al., 2025, 2026), and fused kernels for heterogeneous-precision layouts (Xia et al., 2026; Sun et al., 2026). These methods target AR decoding and exploit stable structures in autoregressive KV caches, such as channel-wise outliers and attention sinks; none exploits the block-conditioned saliency exposed by dLLM decoding.

3 Method

3.1 Background

Inference of diffusion language models. A dLLM generates text by iterative denoising rather than left-to-right prediction. Let $\mathbf{x}^t \in \mathcal{V}^L$ be the length- L sequence state at step t , where the vocab-

ulary $\mathcal{V}$ includes a special [MASK] token. Decoding starts from a fully masked response appended to a prompt $(c_0, \dots, c_{p-1})$ of length p ,

$$\mathbf{x}^T = (c_0, \dots, c_{p-1}, \underbrace{[\text{MASK}], \dots, [\text{MASK}]}_{L-p}).$$

At each step $t = T, \dots, 1$, a mask predictor f_θ scores the whole sequence at once, and a token is predicted for every masked position,

$$\hat{x}_i^t = \arg \max_{v \in \mathcal{V}} \text{softmax}(f_\theta(\mathbf{x}^t)_i)_v;$$

where $i : x_i^t = [\text{MASK}]$. A transition rule then commits a confident subset of these predictions, $\mathbf{x}^{t-1} = S(\hat{\mathbf{x}}^t, \mathbf{x}^t)$, until $\mathbf{x}^0$ contains no masks (Nie et al., 2026; Song et al., 2026). Because f_θ uses *bidirectional* attention, each step conditions on the entire sequence (prompt, decoded, and masked positions alike) and recomputes the KV states of all L positions; over the T steps this gives $\mathcal{O}(L^2)$ inference cost against the $\mathcal{O}(L)$ of autoregressive decoding (Song et al., 2026; Jiang et al., 2026). Block-wise semi-autoregressive decoding reduces this cost by splitting the response into B contiguous blocks decoded sequentially. Since KV states outside the current block change little between adjacent steps (Wu et al., 2026; Liu et al., 2025; Ma et al., 2026), they are cached and reused while only the current block is recomputed.

Grouped KV cache quantization. Quantization shrinks the cache by storing keys and values as low-bit integers. The standard tuning-free scheme is group-wise round-to-nearest (Liu et al., 2024): a tensor X is split into groups of g elements, and each group is mapped to b -bit integers and back by

$$Q(X) = \left\lfloor \frac{X - z_X}{s_X} \right\rfloor, \quad X' = Q(X) s_X + z_X,$$

with asymmetric per-group zero-point $z_X = \min X$ and step $s_X = (\max X - \min X)/(2^b - 1)$. KIVI (Liu et al., 2024) shows that the grouping *axis* is critical: the key cache has a few large-magnitude channels and must be grouped *per channel* to contain outlier errors, whereas the value cache, which acts as the mixing weights of the attention output, must be grouped *per token*. We adopt this recipe (per-channel keys, per-token values) as the storage format for all non-salient KV in our cache.

3.2 Empirical observations

To ground our design choices, we run two diagnostic probes on the base LLaDA-8B-Instruct (Nie et al., 2026) (i) the dynamics of attention maps across decoding blocks, (ii) the contribution of low-importance KV to the attention output, and the drift of salient KV states across decoding steps.

Attention dynamics during decoding. A common recipe in KV quantization for AR models is to protect the *initial* attention sinks (Xiao et al., 2024; Xia et al., 2026) (a handful of leading prompt tokens that absorb most of the attention mass) at high precision while quantizing the rest. A natural question is what analogous structures appear in dLLM attention. We visualize attention maps at the start of different decoding blocks across multiple layers (Figure 2) and observe two consistent patterns. **(1) Local attention appears as a diagonal stripe.** Queries place high attention on neighboring tokens, motivating an adaptation of AR sink protection into *boundary protection*: the cached tokens immediately before and after the current block remain in high precision. **(2) Salient keys emerge as moving vertical stripes.** Each map also shows a small number of bright vertical stripes that absorb most of the attention mass. Unlike AR initial sinks, these columns sit *inside* the prompt rather than at its head and shift across blocks, so a static first-token or once-for-all high-precision set would miss them. Together, these patterns motivate a static high-precision boundary protection band for neighboring tokens and a *per-block, query-conditioned*

saliency set for the moving vertical stripes (Section 3.3).

Evicted-token contribution and KV-state drift.

To validate our quantization-based approach against eviction-based methods, we further examined the attention output contribution and KV-state drift corresponding to the evicted/salient tokens. We found that the evicted bottom half still contributes $\approx 12\%$ of the aligned attention output on average, rising to 25–35% in the first six layers. Moreover, the top-1% salient prefix tokens drift $1.6\text{--}2\times$ more than the average token in the early and middle layers. This observation motivates the second part of our mixed-precision design: keep most positions in cache at low precision, but refresh the high-attention salient subset at every sub-step instead of treating it as a stationary cache. Appendix Section A provides the detailed figures and analysis.

3.3 Saliency-aware mixed-precision KV quantization

The observations in Section 3.2 motivate replacing the binary keep-or-evict decision with a *soft* alternative: instead of dropping low-saliency KV, we retain the entire cache but at mixed precision. Most historical KV is stored at 2 bits and still attends to the current block. A small, block-conditioned, saliency-selected subset is promoted to a high-precision overlay and recomputed at every denoising sub-step. Figure 3 gives an overview.

Logical precision layout. The cache stores only the states outside the current block, namely the prefix before the block and the suffix after it. We describe the logical precision assignment of the keys as:

$$\underbrace{K_{2\text{bit}}^{\text{pre}} \parallel K_{\text{fp}}^{\text{pre}}}_{\text{cached prefix}} \parallel \underbrace{K_{\text{fp}}^{\text{cur}}}_{\text{fresh current not cached}} \parallel \underbrace{K_{\text{fp}}^{\text{suf}} \parallel K_{2\text{bit}}^{\text{suf}}}_{\text{cached suffix}}.$$

Note that this equation is not a physical memory layout. The packed tensors and their scale/min metadata are stored as separate prefix/suffix segments. The current-block KV is freshly projected in BF16 during the block forward and is not part of the resident cache. Within the cached prefix and suffix, the far segments are stored at 2 bits, while the boundary protection bands adjacent to the current block stay in BF16 (i.e. 32 tokens per side). Cached keys are post-RoPE (Su et al., 2024), so per-

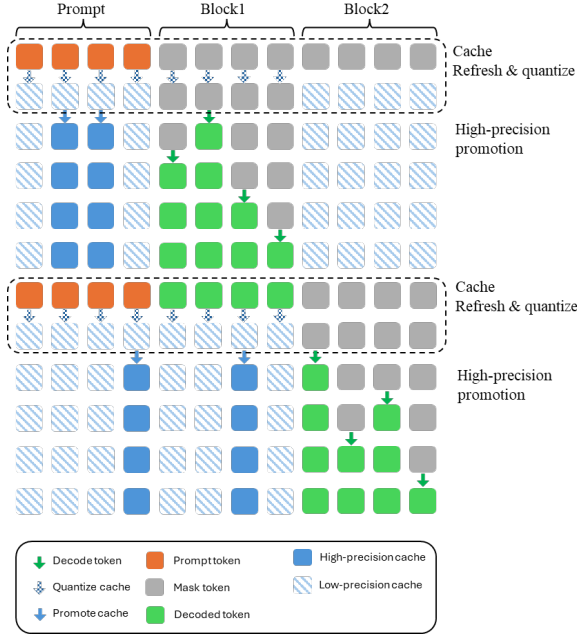


Figure 3: **Overview of our saliency-aware mixed-precision KV cache.** When a block is finished, its KV is refreshed and quantized to 2 bits (low-precision cache), so the resident footprint outside the current block stays low-precision. While decoding a block, a small saliency-selected subset of historical positions is promoted to a high-precision overlay (high-precision cache) and recomputed at every denoising sub-step, while all remaining KV continues to attend at 2 bits.

step attention only rotates the current block. Values follow the same prefix/suffix precision assignment.

Refresh-time quantization. At the start of each decoding block, the refresh forward computes high-precision Q , K , V , then immediately converts the prefix/suffix KV into the 2-bit quantized cache. Keys are quantized per channel and values per token into 2-bit packed tensors with their scale/min metadata; the high-precision KV temporaries are released before leaving the layer rather than being materialized as a full-layer cache. Algorithm 1 in the Appendix gives the pseudocode.

Block-conditioned saliency probe. For the current block b we promote a small budget K of historical prefix positions to high precision. The saliency score is the attention mass flowing from current-block queries to historical prefix keys, summed across heads, queries, and layers:

$$s_b[p] = \sum_{\ell=1}^L \sum_{h=1}^H \sum_{q \in Q_b} \text{softmax}\left(\frac{1}{\sqrt{d}} q^\top K^{(\ell)}\right)_p, \quad (1)$$

where Q_b are the current-block query positions, $K^{(\ell)}$ are the keys at layer ℓ (prefix segment only), and p ranges over historical prefix positions. We evaluate Equation (1) **once per block** using a saliency-probe forward served by the KIVI fused-GEMM attention path. This probe materializes the post-softmax attention scores, reduces them over heads and current-block queries, and accumulates the result into a layer-keyed buffer; the orchestrator then sums across layers and picks the top- K positions:

$$\mathcal{S}_b = \text{Top-k}(s_b; K), \quad (2)$$

so $|\mathcal{S}_b| = K$. Two properties make this signal suitable for dLLMs. (1) It is *block-conditioned*: each block sees a different masked-token distribution and denoising context, so the most useful prefix tokens for block b differ from those for $b + 1$. (2) Its cost is amortized at block granularity: the probe is one forward per block. After $\mathcal{S}_b$ is fixed, the repeated mixed-precision forwards for the block use the fused kernel in Section 3.4, which uses online softmax and never writes the full attention-score tensor to HBM. We quantify the probe overhead in Appendix Section F.

High-precision overlay. Sub-steps $i \geq 2$ treat $\mathcal{S}_b$ as the salient positions promoted on every layer’s cache. The input fed into each transformer block becomes the concatenation

$$\tilde{x}_b = [x_{\mathcal{S}_b}; x_{[s,e)}],$$

where $x_{\mathcal{S}_b}$ are token IDs at the salient positions and $x_{[s,e)}$ is the current block (positions s to $e - 1$). Inside the attention layer, the freshly-projected K and V tensors are split into a salient-token half and a current half. The salient-token half is rotated by RoPE at the absolute positions in $\mathcal{S}_b$ (per-batch gather of sin/cos), giving fresh full-precision K_{sal} and V_{sal} for every sub-step.

3.4 Fused mixed-precision Triton kernel

The mixed-precision cache of Section 3.3 leaves a systems problem: at attention time the current-block query must attend to two key/value sets stored at different precisions, namely the bulk 2-bit quantized prefix/suffix and the small high-precision salient overlay. The naive route dequantizes the entire 2-bit cache into a contiguous high-precision tensor in HBM and applies the high-precision overlay on top of the dequantized tensor, followed by

a standard attention kernel. This materializes the full high-precision cache in HBM and forfeits exactly the memory saving that 2-bit quantization was meant to buy. KIVI’s fused-GEMM path avoids this cache-dequantization, but it still materializes the dense attention-score matrix in HBM. For dLLM decoding, this score matrix grows with context length and becomes a major source of memory traffic. We therefore fuse the mixed-precision attention into a *single* tiled online-softmax pass. The quantized prefix/suffix is walked tile by tile and dequantized only in on-chip SRAM, while the high-precision salient overlay enters as a small dense GEMM whose scores feed the same softmax state (m, ℓ, o) . Because both streams share one running denominator, the overlay and the quantized cache normalize against a single softmax with no post-hoc blending of two attention outputs. Note that every salient position also has a 2-bit cached copy in the quantized prefix. To avoid double-counting, the kernel adds a $-\infty$ mask at the salient position while processing the quantized stream, so each salient position contributes only through its fresh high-precision overlay. The full algorithm and packed layout are in Appendix Section C.

4 Experiments

4.1 Setup

Our evaluation has two parts. Section 4.2 establishes that the mixed-precision cache preserves task *accuracy* on reasoning and coding benchmarks. Section 4.3 gives a more comprehensive characterization of *throughput and peak memory* under a range of workloads, including batch size, prompt length, and matched-memory operating points.

Models, datasets, metrics, and hardware. The main experiments use LLaDA-8B-Instruct (Nie et al., 2026) and Dream-v0-Base-7B (Ye et al., 2025) with the same generation parameters. Unless noted otherwise, block length is $L_{\text{blk}} = 32$, total steps are $T_{\text{tot}} = 256$, generation length is $L_{\text{gen}} = 256$, so that each step decodes exactly one token. The quantized cache uses 2-bit with group size $g = 32$. Accuracy is evaluated on four reasoning and coding benchmarks, GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021b). We use all datasets in accordance with their original licenses and terms of use. Efficiency is reported as decode throughput (tokens/s) and peak GPU memory (GiB). All runs are on

a single NVIDIA A100 80 GB PCIe. For each setting, we run three trials and report the mean throughput/peak-memory.

Baselines. We compare four methods at matched configuration. *Fast-dLLM* (dual cache) (Wu et al., 2026) is the BF16 baseline with standard SDPA attention (Paszke et al., 2019). *KIVI* (Liu et al., 2024) stores the cache at 2-bit and serves it with a fused-GEMM kernel. *Sparse-dLLM* (Song et al., 2026) is based on KV eviction (max-pool kernel size 3, keep ratio 0.5). *SampKV* (*ours*) is the saliency-aware mixed-precision cache served by the fused mixed-precision attention kernel of Section 3.4; the full recipe is *2-bit + boundary protection + adaptive-saliency*: one high-precision boundary protection group on each side of the current block (32 tokens). For accuracy we report salient token budgets $K \in \{16, 32\}$; for throughput/memory sweeps we use $K = 16$ unless noted otherwise.

4.2 Accuracy on reasoning and coding benchmarks

Table 1 reports accuracy across the BF16 baseline, the Sparse-dLLM eviction baseline, KIVI uniform 2-bit, and our full recipe at saliency budgets 16 and 32. On LLaDA-8B-Instruct, uniform 2-bit is roughly lossless on GSM8K but degrades on harder benchmarks, with a worst drop of -3.40 pp on MBPP. The saliency-aware overlay closes this gap. At budget 32, SampKV matches or exceeds BF16 on every benchmark, with the largest recovery on MBPP. On Dream-v0-Base-7B, naive 2-bit is more fragile, losing -4.24 pp on GSM8K and -5.26 pp on MATH. Boundary protection alone recovers part of this loss, but remains below the adaptive overlay. SampKV gives the best quantized-cache result on all tasks, while exceeding Dream’s BF16 dual cache baseline on GSM8K, HumanEval, and MBPP. Dream MATH remains the hardest case, where Sparse-dLLM and SampKV at budget 32 are nearly tied but both remain below BF16 dual cache.

4.3 Throughput and memory

Figures 4 and 5 report the batch and prompt sweeps with their throughput and peak-memory results. *Throughput*: on the batch-16 prompt sweep, KIVI fused-GEMM reaches $1.68\text{--}1.71\times$ BF16 throughput, while the proposed SampKV reaches $1.76\text{--}2.00\times$. SampKV also consistently outperforms KIVI across the batch sweep. The

Table 1: Accuracy on four reasoning / coding benchmarks for LLaDA-8B-Instruct and Dream-v0-Base-7B. Uniform 2-bit quantization loses accuracy on the harder benchmarks; SampKV narrows the gap by keeping a small saliency-selected high-precision overlay.

Base model	Method	GSM8K	MATH	HumanEval	MBPP
LLaDA	Fast-dLLM (BF16, dual cache) (Wu et al., 2026)	77.63	32.92	37.23	36.60
	Sparse-dLLM (BF16) (Song et al., 2026)	77.79(+0.16)	32.80(−0.12)	33.54(−3.69)	36.80(+0.20)
	KIVI (2-bit, no protection) (Liu et al., 2024)	77.63(±0.00)	31.58(−1.34)	36.59(−0.64)	33.20(−3.40)
	KIVI (2-bit, boundary protection)	77.33(−0.30)	32.48(−0.44)	37.80(+0.57)	34.40(−2.20)
	SampKV (2-bit, budget= 16)	77.71(+0.08)	33.04(+0.12)	38.41 (+1.18)	35.40(−1.20)
	SampKV (2-bit, budget= 32)	78.39 (+0.76)	34.38 (+1.46)	37.80(+0.57)	36.60 (±0.00)
Dream	Fast-dLLM (BF16, dual cache) (Wu et al., 2026)	74.07	34.72	39.02	36.40
	Sparse-dLLM (BF16) (Song et al., 2026)	74.22(+0.15)	32.48(−2.24)	36.58(−2.44)	34.40(−2.00)
	KIVI (2-bit, no protection) (Liu et al., 2024)	69.83(−4.24)	29.46(−5.26)	37.80(−1.22)	34.00(−2.40)
	KIVI (2-bit, boundary protection)	71.34(−2.73)	31.32(−3.40)	37.80(−1.22)	35.40(−1.00)
	SampKV (2-bit, budget= 16)	72.93(−1.14)	31.64(−3.08)	41.46 (+2.44)	35.60(−0.80)
	SampKV (2-bit, budget= 32)	74.15 (+0.08)	32.46 (−2.26)	39.63(+0.61)	36.60 (+0.20)

Model names abbreviate LLaDA-8B-Instruct and Dream-v0-Base-7B. Parenthesized numbers are accuracy deltas against the BF16 dual cache baseline within the same base-model block. Bold marks the best quantized-cache result in each benchmark.

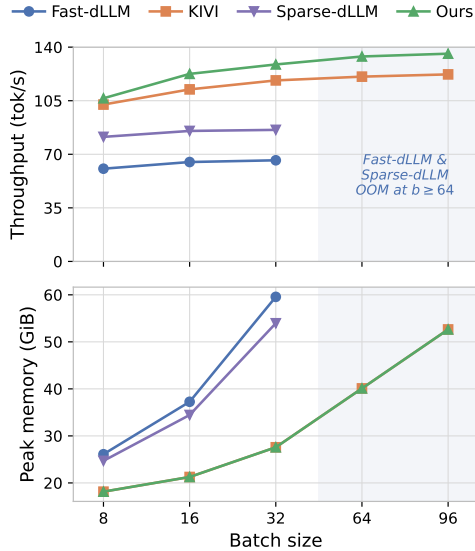


Figure 4: **Batch-size sweep at prompt 1024.** *Top*: decode throughput; *bottom*: peak GPU memory. SampKV scales to $b = 96$ while the unquantized methods run out of memory at $b > 32$. KIVI and ours store the same 2-bit cache, so their memory curves coincide.

advantage widens at longer contexts and larger batches. This is consistent with the structural difference between the kernels: KIVI’s fused-GEMM path consumes the packed cache without writing a dequantized KV tensor to HBM, but it still materializes the dense attention-score matrix, whereas our fused attention kernel computes attention in tiles and uses online softmax to keep only small intermediate states on chip. *Memory*: on the prompt sweep, both quantized paths reduce peak memory by 34–55% relative to BF16; on the batch sweep,

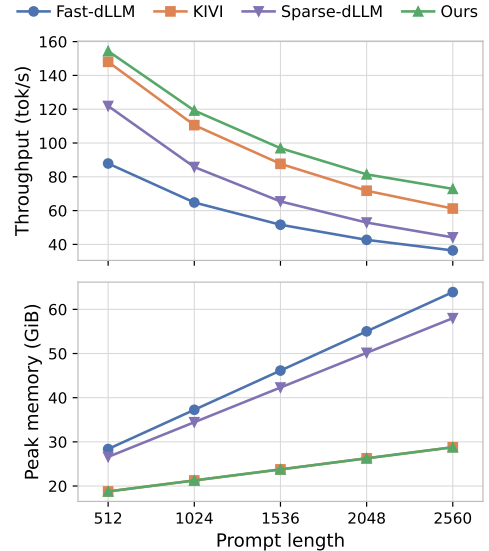


Figure 5: **Prompt-length sweep at batch 16.** *Top*: decode throughput; *bottom*: peak GPU memory. SampKV delivers $1.76\text{--}2.00\times$ BF16 throughput and reduces peak memory by 34–55%; our method beats KIVI fused-GEMM at every prompt length, with the gap widening at longer contexts.

they cut peak memory roughly in half relative to BF16 and Sparse-dLLM, which OOM at $b > 32$, while SampKV scales to $b = 96$ at 52.6 GiB peak.

Figure 6 sweeps batch size at prompt 1024 and plots the resulting (peak memory, tok/s) operating points. At every matched-memory anchor, SampKV delivers $\geq 1.9\times$ the BF16 throughput, peaking at $2.06\times$ around ~ 34 GiB. Sparse-dLLM lifts throughput modestly above BF16 but, because its cache stays in BF16, lags behind our 2-bit path

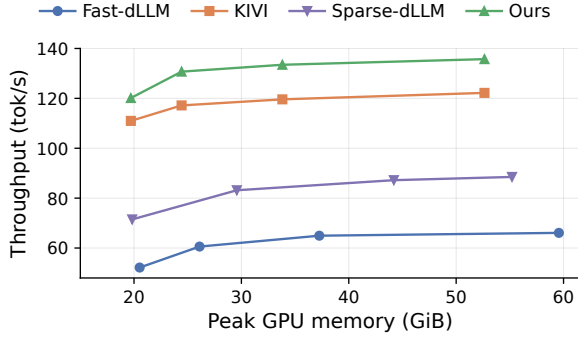


Figure 6: **Iso-memory throughput at prompt 1024.** At every matched-memory anchor SampKV delivers $\geq 1.9\times$ the BF16 throughput, reaching $2.06\times$ near 34 GiB.

Table 2: Accuracies for different boundary protection strategies at saliency budget $K=32$; 0+0, 1+0, and 1+1 protect no boundary, the prefix side, and both sides.

Protection	GSM8K	HumanEval	MBPP
0+0 (none)	77.41	37.20	35.40
1+0 (prefix only)	77.71	37.20	35.80
0+1 (suffix only)	77.48	35.37	36.60
1+1 (both, default)	78.39	37.80	36.60

at every memory budget and shares the BF16 OOM ceiling.

5 Analysis and Ablations

5.1 Ablation on the boundary protection band

Boundary protection keeps the $p \cdot g$ tokens immediately adjacent to the current block at high precision, with each protection group containing $g=32$ tokens (Section 3.3). At saliency budget $K=32$, we compare 0+0 (no protection), 1+0 (prefix side only), 0+1 (suffix side only) and 1+1 (both sides, our default).

Table 2 reports the result. The updated $K=32$ sweep shows a consistent trend: adding prefix boundary protection gives a small gain on GSM8K (+0.30 pp) and MBPP (+0.40 pp), while adding the suffix band on top of it (1+0 $\rightarrow$ 1+1) improves all three benchmarks. Suffix-only protection (0+1) improves GSM8K and MBPP over no protection but reduces HumanEval, which indicates that prefix- and suffix-side protection provide complementary benefits. Overall, 1+1 is consistently best, improving over no boundary protection by +0.60 to +1.20 pp. This supports the attention-map pattern in Section 3.2: the diagonal stripe shows strong attention near the current-block boundary, making boundary protection a useful dLLM analogue of

Table 3: Accuracies under different saliency signals (%); Att is aggregated attention mass, SpD is Sparse-dLLM-style logits, and AOC is aligned output contribution.

Signal	GSM8K	MATH	HumanEval	MBPP
Att (ours)	77.71	33.04	38.41	35.40
SpD	77.48	32.74	39.02	34.20
AOC	77.33	32.52	37.80	34.80

sink protection in AR cache quantization. We therefore use 1+1 by default.

5.2 Ablation on the saliency signal

We compare the block-conditioned attention-mass score of Equation (1) against two alternatives at matched budget $K=16$: *Att*, the aggregated post-softmax attention mass (Section 3.3); *SpD*, the importance score used by Sparse-dLLM (raw $\langle \bar{q}, k \rangle$ logits, head-mean, max-pooled with kernel 3); and the aligned attention-output contribution (*AOC*) defined as $\sum A_{q,p} \langle v_p, O_q \rangle / \|O_q\|^2$ (Appendix Section A). Table 3 reports the result. Attention mass wins on **three of the four** benchmarks (GSM8K, MATH, MBPP), with margins of +0.23, +0.30, and +0.60 pp over the strongest alternative on those tasks. Its only loss is on HumanEval, where SpD leads by 0.61 pp. Attention mass is also the cheapest of the three: it needs only a sum-reduction over the post-softmax attention scores, while SpD adds a max-pool and AOC adds a per-layer aligned product against the attention output. We therefore adopt attention mass as the default saliency signal.

6 Conclusion

We presented SampKV, a saliency-aware mixed-precision KV cache for diffusion language models. It keeps most history in 2 bits, promote a small budget of attention-salient prefix tokens to high precision, and serve both through a fused Triton kernel that streams the quantized cache and the high-precision overlay in one pass. Our approach achieves two key results: (i) it demonstrates that 2-bit KV cache quantization in dLLMs can maintain the accuracy of the BF16 baseline through a small high-precision salient-token overlay; and (ii) it significantly alleviates the memory bottleneck in dLLM inference, with up to 54% peak memory reduction and $2.06\times$ throughput improvement at matched memory consumption compared to BF16 baseline.

Limitations

SampKV primarily targets the memory bottleneck introduced by KV cache in dLLM inference. It is most effective for context-heavy workloads, especially large batches and long prompts, and for models whose KV caches place substantial pressure on GPU memory. Its benefit can be diminishing when the workload has little context or the model is lightweight, because inference in that regime is more likely to be compute-bound and reducing cache storage or cache bandwidth alone leaves less room for speedup. A natural follow-up is to compose SampKV with token-level compute-skipping methods, so that the compute and memory bound can be optimized simultaneously.

Another direction to improve the throughput of dLLM inference is parallel multi-token decoding, which is orthogonal to our work. In this work, we intentionally use single-token decoding in the experiments, so that every compared method performs exactly the same number of model forwards, isolating the cache and kernel effects from reductions in the number of denoising steps. However, parallel decoding reduces the number of repeated forwards per block, so the once-per-block probe is amortized over fewer steps, which could result in smaller end-to-end throughput gain.

Ethical Considerations

This work is an inference-efficiency method and does not introduce new training data, model objectives, or user-facing generation capabilities beyond those of the underlying dLLM. Its main risk is therefore enabling cheaper or higher-throughput deployment of models that may already produce harmful, biased, copyrighted, or misleading content. In addition, mixed-precision cache approximation can change model outputs in ways that are not fully predictable from aggregate benchmark accuracy, so it should be validated carefully before use in safety-critical or high-stakes settings. We view SampKV as an optimization layer that should be combined with the same content-safety, monitoring, and evaluation procedures required for the base model.

Acknowledgments

This work is partially supported by the NSF (CCF-2106725, CCF-2212346, CCF-2425399, CCF-2529764, CCF-2529763) and the Hans Fischer Se-

nior Fellowship of the Technical University of Munich.

References

- Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pages 50726–50753.
- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fu-Chen Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, and 31 others. 2026. LLaDA2.1: Speeding Up Text Diffusion via Token Editing. *arXiv preprint arXiv:2602.08676*.
- Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, and 12 others. 2025. LLaDA2.0: Scaling Up Diffusion Language Models to 100B. *arXiv preprint arXiv:2512.15745*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 34 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Minsoo Cheong, Donghyun Son, Woosang Lim, and Sungjoo Yoo. 2026. EntropyCache: Decoded Token Entropy Guided KV Caching for Diffusion Language Models. *arXiv preprint arXiv:2603.18489*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303.
- Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed Abdelfattah, Jae-sun Seo, Zhiru Zhang, and Udit Gupta. 2026. Flashdlm: Accelerating diffusion language model inference via efficient kv caching and guided diffusion. In *International Conference on Learning Representations*, volume 2026, pages 153529–153544.
- Jianuo Huang, Yaojie Zhang, Yicun Yang, Benhao Huang, and Linfeng Zhang. 2026. Mask tokens as prophet: Fine-grained cache eviction for efficient dLLM inference. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 3456–3479.
- Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, et al. 2026. d²Cache: Accelerating diffusion-based LLMs via dual adaptive caching. In *International Conference on Learning Representations*, volume 2026, pages 137511–137531.
- Omin Kwon, Yeonjae Kim, Doyeon Kim, Minseo Kim, Yeonhong Park, and Jae W Lee. 2026. MAGE: All-[MASK] Block Already Knows Where to Look in Diffusion LLM. *arXiv preprint arXiv:2602.14209*.
- Younjoo Lee, Junghoo Lee, Seungkyun Dan, Jaiyoung Park, and Jung Ho Ahn. 2026. DyLLM: Efficient Diffusion LLM Inference via Saliency-based Token Selection and Partial Attention. *arXiv preprint arXiv:2603.08026*.
- Fei Li, Song Liu, Weiguo Wu, Shiqiang Nie, and Jinyu Wang. 2026. KVmix: Gradient-Based Layer Importance-Aware Mixed-Precision Quantization for KV Cache. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31563–31572.
- Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Wulong Liu, Yiwu Yao, Sinno Jialin Pan, and Mingxuan Yuan. 2025. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. *arXiv preprint arXiv:2502.04420*.
- Kaihua Liang, Xin Tan, An Zhong, Hong Xu, and Marco Canini. 2026. FOCUS: DLLMs Know How to Tame Their Compute Bound. *arXiv preprint arXiv:2601.23278*.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. 2025. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.
- Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. *arXiv preprint arXiv:2402.02750*.
- Lingkun Long, Yushi Huang, Shihao Bai, Ruihao Gong, Jun Zhang, Ao Zhou, and Jianlei Yang. 2026. Focus-dLLM: Accelerating Long-Context Diffusion LLM Inference via Confidence-Guided Context Focusing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12146–12160.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2026. dkv-cache: The cache for diffusion language models. *Advances in Neural Information Processing Systems*, 38:149009–149033.
- Aidar Myrzakhan, Tianyi Li, Bowei Guo, Shengkun Tang, and Zhiqiang Shen. 2026. Sink-aware pruning for diffusion language models. *arXiv preprint arXiv:2602.17664*.
- Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. 2026. Attention is all you need for KV cache in diffusion LLMs. In *International Conference on Learning Representations*, volume 2026, pages 34915–34946.
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025. Scaling up masked diffusion models on text. In *International Conference on Learning Representations*, volume 2025, pages 82974–82997.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.
- Maximo Eduardo Rulli, Simone Petrucci, Edoardo Michielon, Fabrizio Silvestri, Simone Scardapane,

- and Alessio Devoto. 2026. Attention sinks in diffusion language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 43244–43256.
- Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.
- Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. 2026. Sparse-dLLM: Accelerating diffusion LLMs with dynamic cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33038–33046.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Ling Li, Huangqi Yu, and Kehong Yuan. 2025. RotateKV: Accurate and Robust 2-Bit KV Cache Quantization for LLMs via Outlier-Aware Adaptive Rotations. *arXiv preprint arXiv:2501.16383*.
- Zunhai Su and Kehong Yuan. 2025. Kvsink: Understanding and enhancing the preservation of attention sinks in kv cache quantization for llms. *arXiv preprint arXiv:2508.04257*.
- Chengyu Sun, Yaqi Xia, Hulin Wang, Donglin Yang, Xiaobo Zhou, and Dazhao Cheng. 2026. Janusquant: Accurate and efficient 2-bit kv cache quantization for long-context inference. In *Proceedings of the 31st ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 301–314.
- Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA. Association for Computing Machinery.
- Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. 2026a. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. In *International Conference on Learning Representations*, volume 2026, pages 2282–2299.
- Zeqing Wang, Gongfan Fang, Xinyin Ma, Xingyi Yang, and Xinchao Wang. 2026b. SparseD: Sparse attention for diffusion language models. In *International Conference on Learning Representations*, volume 2026, pages 99320–99337.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2026. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pages 57027–57051.
- Haocheng Xi, Harman Singh, Yuezhou Hu, Coleman Hooper, Rishabh Tiwari, Aditya Tomar, Minjae Lee, Wonjun Kang, Michael W. Mahoney, Chenfeng Xu, Kurt Keutzer, and Amir Gholami. 2026. LoSA: Locality Aware Sparse Attention for Block-Wise Diffusion Language Models. *arXiv preprint arXiv:2604.12056*.
- Haojun Xia, Jisen Li, Tsai-chuan Wu, Junxiong Wang, Jue Wang, Chenxi Li, Aman Singhal, Alay D Shah, Alpay Ariyak, Donglin Zhuang, et al. 2026. Kitty: Accurate and efficient 2-bit kv cache quantization with dynamic channel-wise precision boost. *Proceedings of Machine Learning and Systems*, 8:511–525.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, volume 2024, pages 21875–21895.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.
- Zihou Zhang, Zheyong Xie, Li Zhong, Haifeng Liu, and Shaosheng Cao. 2026. One token is enough: Improving diffusion language models with a sink token. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 6479–6490.

A Empirical observations: details

This appendix section expands the second probe of Section 3.2: the contribution of low-importance KV entries to the attention output, and the drift of high-importance KV states across decoding steps.

Evicted tokens still carry the output. A natural way to shrink the cache is to rank historical positions by an attention-importance score and drop the low-ranked portion (Song et al., 2026). To see what such an eviction removes, we rank every historical position by attention importance, split it at the median, and decompose the attention output exactly as $O = O_{\text{top}} + O_{\text{bottom}}$. We then measure the *aligned* share of the bottom half: $\langle O_{\text{bottom}}, O \rangle / \|O\|^2$, averaged over heads, queries, and layers. The evicted 50% of the cache accounts for $\approx 12\%$ of the attention output, rising to 25–35% in the first six layers (Figure 7). The discarded entries are thus low-scoring but far from negligible: because the output is value-weighted, a low-score key with a large value vector still contributes substantially. The layer-wise discrepancy also indicates that uniform treatment across layers leaves accuracy on the table.

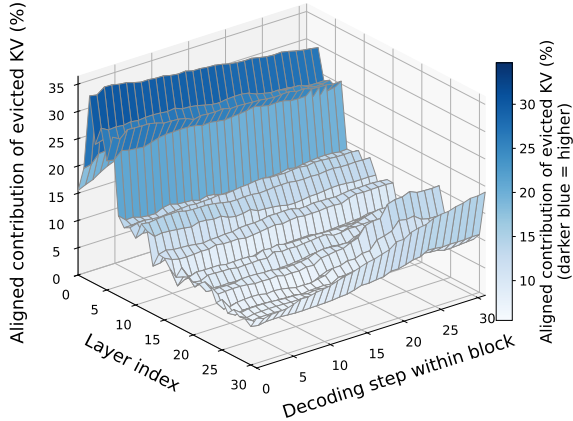


Figure 7: **Evicted-token contribution to the attention output, per layer.** The evicted bottom-50% of the cache still contributes $\approx 12\%$ of the aligned attention output on average, and 25–35% in the early layers.

The most-important KV are also the least stationary. A cache built once per block implicitly assumes its high-importance entries are stable for the block’s duration. We probe this by running unmodified LLaDA decoding (a full bidirectional forward at every step with no caching) and measuring each position’s KV drift $\| \text{state}_t - \text{state}_{t-1} \|_2$

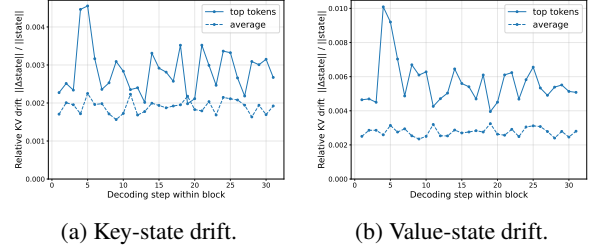


Figure 8: **KV-state drift across decoding steps.** The top-1% “important” prefix tokens drift 1.6–2 $\times$ more than the average token in the early and middle layers, for both keys (a) and values (b).

between adjacent steps. Sharpening the importance criterion to the top 1% of prefix positions exposes a clear pattern: in the early and middle layers these top-1% important tokens drift 1.6–2 $\times$ more than the average prefix token (at layer 15, their value states move 6.4% of their norm per step versus 2.2% for the rest; Figures 8a and 8b). Thus, high-attention salient tokens are a poor fit for a frozen high-precision cache and are better handled by the refreshed overlay.

B Cache refresh and quantization

At the start of every decoding block, the model performs one refresh forward over the full sequence. This forward still uses the standard high-precision attention computation, but each layer builds its own quantized prefix and suffix cache before returning, so that the full BF16 keys and values are local temporaries. Only the packed 2-bit tensors and scale/min tensors return from the layer. Thus the refresh path never materializes a full-layer BF16 KV cache outside the layer scope. Algorithm 1 shows the attention-level operation for one layer.

C Fused mixed-precision Triton kernel

This appendix expands the kernel of Section 3.4.

Fused single-pass kernel. The kernel meets the two requirements, (i) keep the cache 2-bit-packed in HBM and dequantize only in on-chip SRAM, and (ii) fold the quantized stream and the high-precision overlay into a single softmax normalization without materializing the full attention-score matrix or writing a dequantized KV tensor to HBM (Algorithm 2). It carries a single online softmax state $(m, \ell, \mathbf{o})$ in FP32 and consumes two streams in sequence. *Stream 1* is the 2-bit cache. The kernel walks the quantized prefix and suffix in tiles of BLOCK_N , unpacking the int32-packed keys and

Algorithm 1 Cache refresh and quantization for one layer.

Input: layer ℓ ; layer input hidden states H ; current block $[s, e]$; quantization width 2 bits
Output: attention output A^ℓ and layer cache $\mathcal{C}^\ell$

- 1: $Q, K, V \leftarrow W_Q^\ell H, W_K^\ell H, W_V^\ell H$
normal attention in high precision
- 2: $A^\ell \leftarrow \text{Attention}(Q, K, V)$
- 3: $K_{\text{cache}} \leftarrow K_{:s} \parallel K_{:e}; \quad V_{\text{cache}} \leftarrow V_{:s} \parallel V_{:e};$
per-channel key quantization
- 4: $(\hat{K}, S_K, Z_K) \leftarrow \text{QUANT}(K_{\text{cache}})$
per-token value quantization
- 5: $(\hat{V}, S_V, Z_V) \leftarrow \text{QUANT}(V_{\text{cache}})$
- 6: $\mathcal{C}^\ell \leftarrow (\hat{K}, S_K, Z_K, \hat{V}, S_V, Z_V, s, e)$
full BF16 KV does not escape layer
- 7: release local $Q, K, V, K_{\text{cache}}, V_{\text{cache}}$
- 8: **return** $A^\ell, \mathcal{C}^\ell$

values with their per-channel / per-token scale and zero-point directly in registers, so no dequantized tensor is ever written back to HBM. *Stream 2* is the high-precision overlay. $QK_{\text{sal}}^\top$ is a small dense high-precision GEMM whose scores feed the *same* online softmax state. Because both streams update one $(m, \ell, \mathbf{o})$, the overlay and the quantized cache share a single exact softmax denominator, with no separate normalization and no post-hoc blend of two attention outputs.

D Implementation details for reproducibility

The implementation is organized as a quantized-stream attention pass followed by high-precision online-softmax merge passes. The quantized pass processes the concatenated 2-bit prefix and suffix. The BF16 middle segments (prefix boundary protection, current block, and suffix boundary protection) and the fresh salient-token overlay are then merged using the same FP32 state $(m, \ell, \mathbf{o})$. This implementation split is equivalent to Algorithm 2: all segments share one online-softmax denominator, and no full attention-score matrix is written to HBM.

Tensor layout. The query input has shape $[B, n_h, M, d_h]$ in BF16. Packed keys have shape $[B, n_{kv}, d_h, T/(32/k)]$, with scale/min tensors $[B, n_{kv}, d_h, T/g]$. Packed values have shape $[B, n_{kv}, T, d_h/(32/v)]$, with scale/min tensors $[B, n_{kv}, T, d_h/g]$. The running max and log-sum-exp tensors are $[B, n_h, M]$ in FP32, and the unnor-

Algorithm 2 Fused mixed-precision attention (one query tile).

Input: query Q ; 2-bit cache $(\hat{K}, \hat{V})$ with scales (S_K, S_V) and zero-points (Z_K, Z_V) ; salient overlay $(K_{\text{sal}}, V_{\text{sal}})$; saliency masks Φ
Output: attention output O
FP32 online-softmax state

- 1: $m \leftarrow -\infty; \ell \leftarrow 0; \mathbf{o} \leftarrow \mathbf{0}$
Stream 1, 2-bit quantized prefix/suffix
- 2: **for** each key tile j of the quantized cache **do**
unpack in SRAM, no HBM write
- 3: $K_j \leftarrow \text{DEQUANT}(\hat{K}_j, S_{K,j}, Z_{K,j})$
mask $-\infty$ at salient positions
- 4: $\Sigma \leftarrow QK_j^\top / \sqrt{d} + \Phi_j$
- 5: $m' \leftarrow \max(m, \text{rowmax } \Sigma)$
- 6: $\alpha \leftarrow e^{m-m'}, P \leftarrow e^{\Sigma-m'}$
- 7: $\ell \leftarrow \alpha\ell + \text{rowsum } P$
- 8: $V_j \leftarrow \text{DEQUANT}(\hat{V}_j, S_{V,j}, Z_{V,j})$
- 9: $\mathbf{o} \leftarrow \alpha\mathbf{o} + PV_j$
- 10: $m \leftarrow m'$
Stream 2, high-precision salient overlay
- 11: $\Sigma \leftarrow QK_{\text{sal}}^\top / \sqrt{d}$
- 12: $m' \leftarrow \max(m, \text{rowmax } \Sigma)$
- 13: $\alpha \leftarrow e^{m-m'}, P \leftarrow e^{\Sigma-m'}$
- 14: $\ell \leftarrow \alpha\ell + \text{rowsum } P$
- 15: $\mathbf{o} \leftarrow \alpha\mathbf{o} + PV_{\text{sal}}$
- 16: **return** $O \leftarrow \mathbf{o}/\ell$

malized output accumulator is $[B, n_h, M, d_h]$ in FP32. Our experiments use $k = v = 2$ and $g = 32$, so each int32 word stores 16 quantized elements. The implementation requires T to be divisible by g for keys, d_h to be divisible by g for values, and d_h to be a power of two.

Launch and tiling. The wrapper flattens the leading batch-head dimensions and launches a grid of $(\lceil M/\text{BLOCK}_M \rceil, Bn_h)$. Since the decoding block is short, BLOCK_M is set to the next power of two at least 16 and the query tile is padded when necessary. The head dimension tile is fixed to $\text{BLOCK_HEADDIM} = d_h$. We use Triton autotuning to choose the sequence-tile configuration. The autotuner selects the best configuration separately for each combination of head dimension d_h , key bit width k , value bit width v , and quantization group size g . The table below lists the candidate configurations in the search space. Here, BLOCK_N denotes the number of cached KV positions processed by each tile, while `num_warps` and `num_stages` control the amount of GPU paral-

lelism and the depth of software pipelining, respectively.

BLOCK_N	(num_warps, num_stages)
32	(2, 2), (4, 3)
64	(2, 3), (4, 3), (4, 4), (8, 3)
128	(4, 2), (8, 2), (8, 3)

Each program keeps one query tile and the FP32 online-softmax state resident while looping over key/value tiles. Tail positions beyond the effective quantized length are masked to $-\infty$ before the online-softmax update.

Masks and high-precision merge. The additive column mask over the quantized stream is a FP32 tensor of shape $[B, T]$ containing 0 for normal columns and $-\infty$ for padding or salient positions that will be supplied through the fresh BF16 overlay. After the Triton quantized-stream launch, the boundary-protection bands, current block, and salient overlay are merged through the same online-softmax state. Thus each protected or salient position contributes exactly once, while the quantized cache remains packed throughout the Triton pass.

E Benchmark and evaluation details

Table 1 reports the main accuracy results on the full evaluation sets. All accuracy runs use deterministic decoding with generation length 256, block length 32, and 256 total denoising steps. The cache-compression settings follow the main text: 2-bit key/value quantization with group size $g = 32$, one boundary-protection group on each side when boundary protection is enabled, and saliency budgets $K \in \{16, 32\}$ for the adaptive overlay.

Table 4 reports the detailed information and metrics for each benchmark. GSM8K contains grade-school arithmetic word problems and is evaluated with five-shot prompting using the standard answer-extraction exact match. MATH contains competition-style math problems spanning algebra, counting and probability, geometry, intermediate algebra, number theory, prealgebra, and precalculus; we use the four-shot Minerva-style evaluation and report the verified-answer score. HumanEval contains Python function-completion problems and is evaluated zero-shot by executing generated code against the provided unit tests after code extraction. MBPP contains short Python programming tasks with natural-language descriptions and tests; we use the three-shot instruction-format task and report pass@1 after code extraction and execution.

Benchmark	Items	Shots	Metric
GSM8K	1,319	5	exact match
MATH	5,000	4	math verification
HumanEval	164	0	pass@1
MBPP	500	3	pass@1

Table 4: **Accuracy benchmark settings.** The same settings are used for both LLaDA-8B-Instruct and Dreamv0-Base-7B in Table 1.

F Inference runtime breakdown

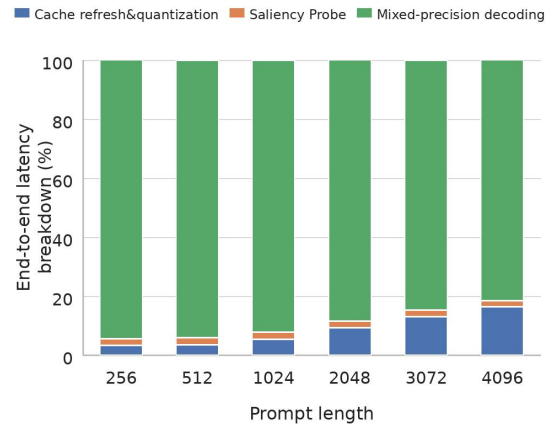


Figure 9: **Runtime breakdown.** Bars are latency percentages for refresh, saliency probe, and mixed-precision decoding; the probe stays at 2.0–2.4%.

We profiled the inference runtime of SampKV on LLaDA-Instruct-8B with batch size 1, generation length 256, block length 32, 256 decoding steps, saliency budget $K=16$, and one boundary protection group on each side of the current block. The profiler separates model forwards into three phases: the layer-wise cache refresh and quantization, the once-per-block saliency probe, and the decoding forwards within each block served by our proposed fused mixed-precision kernel. Figure 9 shows that the saliency probe adds a small, stable cost across the sweep. As context grows, the refresh-and-quantization phase increases from 3.2% to 16.4% of the profiled phases, whereas the probe share decreases slightly from 2.4% to 2.0%. Thus the probe’s overhead is negligible compared with the repeated mixed-precision decoding forwards, which remain the dominant component throughout the tests.