ASK THE MASKED BLOCK: PREDICTING DENOISED-STATE RELEVANCE FOR KV CACHE EVICTION IN DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) incur substantial inference costs by repeatedly processing the full sequence with bidirectional attention. Temporal caching reduces redundant computation but introduces additional memory overhead by retaining full-sequence KV states. Spatial eviction reduces this cache footprint, yet current-state attention may not identify the entries needed after denoising. We introduce Ask-dLLM, a KV cache eviction framework that predicts denoised-state relevance conditioned on the current masked block. A lightweight scorer learns from relevance measured after the same frozen dLLM denoises the block and ranks external cache candidates once per block using masked-state representations. Across 23 reasoning, code, and long-context benchmarks, Ask-dLLM improves the overall mean score over the prior eviction baseline by 7.75 points on LLaDA and 2.39 points on Dream while retaining only 10% of external KV entries. Its overall mean score remains within 1 point of the unmodified Origin model on both backbones. On four LongBench datasets at an 8K context length, Ask-dLLM achieves approximately $9.7\times$ and $11.8\times$ the throughput of Origin on LLaDA and Dream, respectively, while matching or exceeding its average task score. Code is available at `https://anonymous.4open.science/r/36675/README.md`.

1 INTRODUCTION

Masked diffusion language models (dLLMs) (Nie et al., 2026) have emerged as an alternative to autoregressive (AR) generation (Berglund et al., 2024; Dziri et al., 2023), whose token-by-token dependency limits parallelism across output positions. By iteratively resolving masked positions with bidirectional attention, dLLMs enable parallel token updates and have shown promise in overcoming limitations of autoregressive modeling, including the reversal curse (Berglund et al., 2024; Nie et al., 2026), while supporting advanced reasoning capabilities (Yang et al., 2026; Zhao et al., 2026; Zhu et al., 2026). However, while causal attention enables exact reuse of existing key–value (KV) states in AR models (Touvron et al., 2023), dLLM states can change throughout denoising, requiring repeated $\mathcal{O}(N^2)$ computation over the prompt and response in naive inference. Temporal caching reduces this redundancy through approximate KV reuse across denoising steps (Wu et al., 2026; Ma et al., 2026; Liu et al., 2026), but retaining all context positions leaves cache size proportional to sequence length.

To further reduce the memory cost left by temporal caching, recent work has explored spatial KV cache eviction, retaining only a subset of KV entries for subsequent denoising steps. Sparse-dLLM (Song et al., 2026) demonstrates substantial decoding speedups with comparable task performance through attention-guided eviction. However, Sparse-dLLM (Song et al., 2026) ranks cache entries using query–key affinities from the block's early masked state. Our motivating observation is that selection informed by the denoised state can succeed where this early ranking fails. Fig. 1 illustrates this contrast under block-wise decoding with cache refreshes between blocks (Wu et al., 2026; Song et al., 2026). We compare Sparse-dLLM with a Denoised-State Reference (DSR), an analysis-only policy that selects entries using attention measured after full-cache denoising resolves all masked positions in the same block. At a 10% keep ratio, the two policies exhibit different attention patterns: DSR generates the correct answer, whereas Sparse-dLLM does not. The full block-by-block decoding

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.

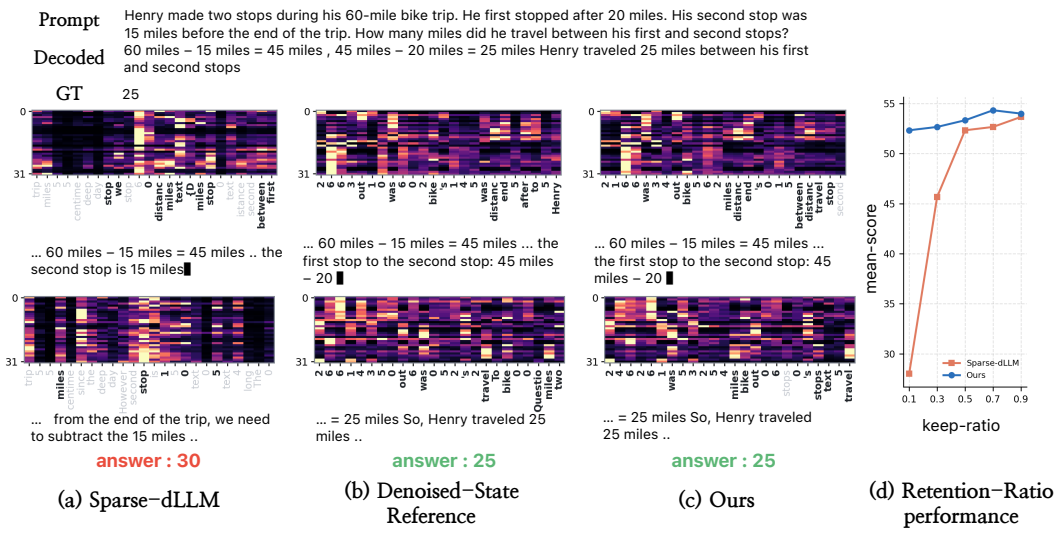

Figure 1: **What to keep depends on the denoised block.** LLaDA-8B-Instruct decodes one GSM8K question at keep ratio $\rho = 0.1$ under Sparse-dLLM (a), the Denoised-State Reference (b), and Ask-dLLM (c). (b) scores the cache with the attention of the *denoised* block, which is unavailable at eviction time; it is an analysis-only reference rather than a deployable policy. Heatmaps show attention at layer 26, KV head h, from a block's 32 query rows to the cache entries it retained; columns are a subset of the entries retained at $\rho = 0.1$, greyed when (b) did not retain them. Sparse-dLLM scores the cache from the current masked state, so what it keeps diverges from what the denoised block attends to. By block 4 the tokens forming `20 miles` are gone while those forming `15 miles` remain, and under this eviction the model subtracts `15` a second time and produces the final answer `30`; (b) and (c) retain both quantities and produce `25`, matching the ground truth. (d) Mean score over 100 samples each of GSM8K, HumanEval and MATH, ρ swept from 0.1 to 0.9 in steps of 0.2. Sparse-dLLM and Ask-dLLM track each other above $\rho = 0.5$ and separate sharply below it, differing by over 24 points at $\rho = 0.1$.

trace for this example is provided in Appendix D.1. Figure 1(d) further compares Ask-dLLM with Sparse-dLLM and shows that their performance gap widens under more aggressive eviction.

Building on this observation, we introduce Ask-dLLM, a KV cache eviction framework that predicts *denoised-state relevance* from the current masked state, where denoised-state relevance denotes the attention-based importance of each KV entry to the completed block. Masked positions already carry hidden states contextualized by the surrounding sequence through bidirectional attention, even before their token identities are resolved. These representations provide a conditioning signal specific to the block being generated. We therefore formulate cache selection as conditional prediction: the masked state supplies the input, while attention measured in the denoised state supplies the supervision. As illustrated in Fig. 2, our lightweight scorer is trained to predict denoised-state attention from masked-state representations and selects external KV entries once per block at inference, without lookahead or attention-map materialization. This prediction enables Ask-dLLM to retain cache entries consistent with the denoised-state reference, preserving the information needed for correct generation even under a tight cache budget (Fig. 1).

We evaluate Ask-dLLM on LLaDA-8B-Instruct (Nie et al., 2026) and Dream-v0-Instruct-7B (Ye et al., 2025) across mathematical reasoning, code generation, and long-context tasks. At a 10% external-cache keep ratio on LLaDA-8B-Instruct, Ask-dLLM improves GSM8K and MATH accuracy over Sparse-dLLM by 43.22 and 19.94 percentage points, respectively, and increases the average score across 16 LongBench tasks by 2.98 points. On GSM8K, it achieves a strict-match accuracy of 55.88%, compared with 54.28% for Origin.

Our contributions are summarized as follows:

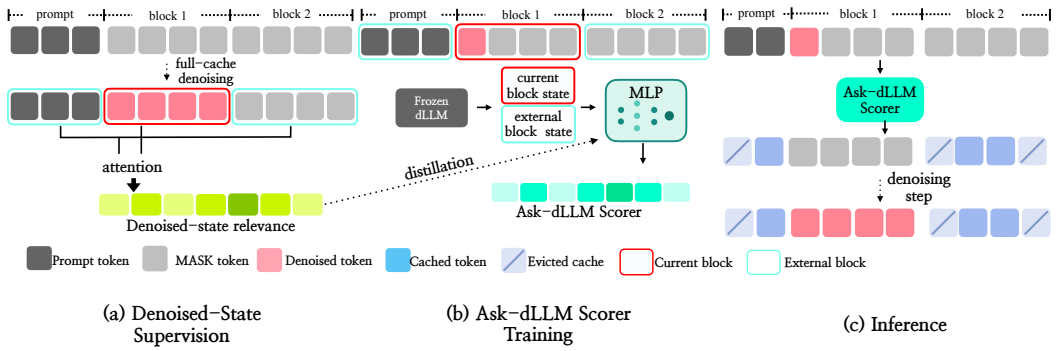

Figure 2: **Overview of Ask-dLLM.** First, the frozen dLLM denoises each current block with the full cache and derives denoised-state relevance from the attention assigned by the denoised block to external cache candidates. Second, a lightweight scorer is trained to predict this relevance from the masked-state representations of the current block and the external candidates. At inference, the supervision path is discarded; Ask-dLLM invokes only the trained scorer once per block and retains the top-K external KV entries for block-wise denoising.

- We identify a mismatch between masked-state cache rankings and denoised-state attention, and formulate dLLM cache eviction as predicting denoised-state relevance conditioned on the current masked block.
- We introduce Ask-dLLM, which learns per-head cache rankings from the frozen backbone's denoised-state attention and performs eviction once per block without lookahead or attention-map materialization at inference.
- Experiments across reasoning, code generation, and long-context tasks demonstrate improved performance under tight cache budgets. At 10% external-cache retention, Ask-dLLM improves GSM8K accuracy over Sparse-dLLM by 43.22 percentage points on LLaDA-8B-Instruct.

2 Related Work

Caching for diffusion LLMs. Diffusion LLMs (dLLMs) (Nie et al., 2026) reprocess the entire sequence with bidirectional attention at every denoising step, so unlike AR models a KV cache is not obtained for free. Nevertheless, the observation that KV states vary little between adjacent steps has motivated a range of approximate caching schemes. dKV-Cache (Ma et al., 2026) stores the KV of a token one step after it is decoded, and Fast-dLLM (Wu et al., 2026) combines a block-wise approximate KV cache, which reuses the prefix and suffix KV outside the current decoding block, with confidence-based parallel decoding. dLLM-Cache (Liu et al., 2026) decouples the refresh intervals of the prompt and response caches and updates only the few tokens with large changes, while FlashDLM (Hu et al., 2026) treats the prompt KV as a reusable constant and reduces the number of required steps itself. This family addresses *when* the KV should be recomputed (temporal reuse) rather than which tokens should be kept at all, so the cache size remains proportional to the full sequence length.

Cache eviction in diffusion LLMs. Efforts to sparsify the KV cache spatially in dLLMs have emerged only recently. Sparse-dLLM (Song et al., 2026) builds on the observation that attention in dLLMs is sparse and that pivotal tokens with high saliency remain relatively stable over decoding steps: it forms the candidate set from the prefix and suffix KV outside the current decoding block, retains only the top-k tokens ranked by the attention scores between the queries of the current block and the candidate keys, and couples this with a delayed cache update, reporting up to $10\times$ higher throughput at a 4k sequence length. However, since the selection signal is the attention observed up to the current step, it remains unknown whether the retained tokens are the ones the completed answer actually attends to.

3 METHOD

3.1 PRELIMINARIES

Notation. We consider a dLLM (Nie et al., 2026; Ye et al., 2025) with L Transformer (Vaswani et al., 2017) layers, hidden dimension D, and N_h KV heads indexed by h. An input consists of a prompt of length N_P and a response of length N_G, giving a total sequence length of $N = N_P + N_G$. The response is divided into $M = N_G/B$ blocks of length B, which are generated sequentially.

Let $\mathcal{I}_P$ denote the prompt positions and $\mathcal{I}_b$ the positions of the b-th response block. For block b, $\mathcal{I}_{<b}$ contains previously denoised positions, whereas $\mathcal{I}_{>b}$ contains future masked positions. Excluding the current block, we define the external candidate set as $\mathcal{C}_b = \mathcal{I}_P \cup \mathcal{I}_{<b} \cup \mathcal{I}_{>b}$. Its size is $N_C = |\mathcal{C}_b| = N - B$. Given a keep ratio $\rho \in (0, 1]$, each head retains $K = \lfloor \rho N_C \rfloor$ external KV entries.

Let $\mathbf{H}^{(\ell,b)}_{\text{mask}} \in \mathbb{R}^{N\times D}$ denote the layer-ℓ hidden states obtained while the current block is in its masked state, and let $\mathbf{h}^{(\ell,b)}_{\text{mask},j}$ denote the hidden state at position j.

Block-wise decoding. A dLLM begins generation with the prompt followed by N_G mask tokens. We refer to the state in which most positions in the current block $\mathcal{I}_b$ remain masked as the *masked state*. Block-wise decoding (Nie et al., 2026) iteratively resolves these positions and advances to block $b + 1$ after all B positions have been determined. We refer to the state in which every position in the current block has been resolved as the *denoised state*. Because these states are defined with respect to the current block, positions in $\mathcal{I}_{>b}$ remain masked even after block b reaches its denoised state.

For each block, after an initial denoising step, a full-sequence cache-refresh forward (Wu et al., 2026; Ma et al., 2026; Liu et al., 2026) constructs the external KV states and the masked-state representations used for selection. Although positions in $\mathcal{I}_{>b}$ remain masked, they already occupy their sequence positions and participate in the computation through bidirectional attention. Ask-dLLM evaluates the external candidates once using the representations obtained from this forward. The selected external KV entries remain fixed until the current block reaches its denoised state. For the next block, the cache is reconstructed and selection is repeated, allowing previously evicted positions to be reconsidered under the new masked-block condition. Pseudocode for offline teacher supervision collection and inference is provided in Appendix B.1.

3.2 DENOISED-STATE RELEVANCE

Existing eviction methods (Song et al., 2026) evaluate cache candidates using attention observed while the current block is in its masked state. However, this attention may not reflect the external KV entries that current-block queries will reference as they evolve throughout denoising. We therefore define the denoised-state relevance of each external candidate using the attention assigned to it when the same block reaches its denoised state. Because this quantity is unavailable in the masked state, we measure it during training using the same frozen dLLM as the teacher (Hinton et al., 2015).

Block-wise teacher measurement. For each block b, the teacher first performs a cache-refresh forward in the masked state with eviction disabled, retaining all external KV entries in $\mathcal{C}_b$. The resulting hidden states $\mathbf{H}^{(\ell,b)}_{\text{mask}}$ are used as inputs to the Ask-dLLM scorer. The teacher then denoises the current block using the full external cache until every position in $\mathcal{I}_b$ has been resolved.

Once block b reaches its denoised state, the teacher performs one additional block-only forward using the denoised block tokens as queries. The external candidate KV states remain those constructed during the masked-state cache-refresh forward. From this forward, we extract the attention assigned by the denoised block tokens to each candidate $j \in \mathcal{C}_b$.

Let $\widetilde{\alpha}^{(\ell,b)}_{h,i,j}$ denote the attention assigned by denoised block token i to external candidate j, aligned with KV head h. For GQA (Ainslie et al., 2023), attention is averaged over the query heads that share the same KV head; for MHA (Vaswani et al., 2017), this aggregation reduces to the identity. We define the denoised-state relevance of candidate j as

$$r^{(\ell,b)}_{h,j} = \max_{i\in\mathcal{I}_b} \widetilde{\alpha}^{(\ell,b)}_{h,i,j}, \qquad j \in \mathcal{C}_b. \tag{1}$$

The teacher repeats this measurement for every block, layer, and KV head following the generation order. The resulting $r_{h,j}^{(\ell,b)}$ serves as the supervision target for training the Ask-dLLM scorer.

3.3 Training the Ask-dLLM Scorer

The Ask-dLLM scorer learns to predict the ranking induced by the teacher-derived denoised-state relevance using hidden states available in the masked state. For each layer ℓ and block b, it takes $\mathbf{H}_{\text{mask}}^{(\ell,b)}$ as input and produces a score $s_{h,j}^{(\ell,b)}$ for each external candidate $j \in \mathcal{C}_b$ and head h. The current block $\mathcal{I}_b$ is excluded from the candidate set and instead provides the condition used to evaluate all external candidates.

Block-conditioned scoring. For each candidate j, we project its masked-state representation as $\mathbf{u}_j^{(\ell,b)} = \mathbf{W}_{\text{tok}}^{(\ell)} \mathbf{h}_{\text{mask},j}^{(\ell,b)}$. We obtain the current-block condition by mean-pooling its hidden states and projecting the result as $\mathbf{c}_b^{(\ell)} = \mathbf{W}_{\text{blk}}^{(\ell)}(B^{-1} \sum_{i \in \mathcal{I}_b} \mathbf{h}_{\text{mask},i}^{(\ell,b)})$. The scorer combines these representations as

$$\mathbf{s}_j^{(\ell,b)} = \text{MLP}^{(\ell)} \left(\left[\mathbf{u}_j^{(\ell,b)}; \mathbf{c}_b^{(\ell)}; \mathbf{u}_j^{(\ell,b)} \odot \mathbf{c}_b^{(\ell)} \right] \right) \in \mathbb{R}^{N_h}. \tag{2}$$

The h-th component $s_{h,j}^{(\ell,b)}$ is the ranking score for candidate j at head h. Each layer uses a separate scorer whose parameters are shared across generation blocks. We use the Base scorer in the main experiments. Appendix C.2 examines the trade-offs between scorer capacity, task performance, scoring latency, and memory usage.

Ranking objective. We train the scorer with a listwise term (Cao et al., 2007) and a pairwise term (Burges et al., 2005). For each layer ℓ, block b, and head h (superscripts omitted for brevity), we normalize the teacher-derived relevance over external candidates as $\pi_{h,j} = r_{h,j} / \sum_{k \in \mathcal{C}_b} r_{h,k}$ to obtain the target distribution $\boldsymbol{\pi}_h$. The predicted distribution is obtained by applying softmax to the scorer outputs: $\widehat{\boldsymbol{\pi}}_h = \text{softmax}_{j \in \mathcal{C}_b}(s_{h,j})$. For a sampled candidate pair (j, k) with unequal teacher relevance, let $y_{h,j,k} = \text{sign}(r_{h,j} - r_{h,k}) \in \{-1, +1\}$ and $\Delta s_{h,j,k} = s_{h,j} - s_{h,k}$. The two terms are

$$\mathcal{L}_{\text{list}} = \frac{1}{N_h} \sum_{h=1}^{N_h} D_{\text{KL}} \left(\boldsymbol{\pi}_h \,\|\, \widehat{\boldsymbol{\pi}}_h \right), \qquad \mathcal{L}_{\text{rank}} = \mathbb{E}_{(h,j,k)} \left[\text{softplus} \left(-y_{h,j,k} \Delta s_{h,j,k} \right) \right]. \tag{3}$$

Their combined objective is

$$\mathcal{L}^{(\ell,b)} = \mathcal{L}_{\text{list}}^{(\ell,b)} + \mathcal{L}_{\text{rank}}^{(\ell,b)}. \tag{4}$$

3.4 Block-wise Cache Eviction at Inference

At inference time, the teacher is removed and only the trained Ask-dLLM scorer is attached to the frozen dLLM. At the masked state of each block, the cache-refresh forward constructs $\mathbf{H}_{\text{mask}}^{(\ell,b)}$ and the corresponding KV states. Excluding the block currently being denoised, the scorer ranks all remaining KV entries as external candidates and retains the K highest-scoring candidates at each layer and head.

The scorer runs once per block using hidden states already computed by the cache-refresh forward. It requires neither attention weights nor attention-map materialization for eviction, preserving compatibility with fused attention kernels (Dao et al., 2022).

4 Experiments

4.1 Experimental Setup

We evaluate LLaDA-8B-Instruct (Nie et al., 2026) and Dream-v0-Instruct-7B (Ye et al., 2025) on 16 LongBench datasets (Bai et al., 2024), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), ARC-Challenge (Clark et al.,

Table 1: **Long-context performance under aggressive eviction.** Results on 16 LongBench datasets under the default experimental configuration. Avg. denotes the mean across all datasets. The best eviction result at each keep ratio is shown in bold. Origin denotes the unmodified backbone without caching.

	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Synthetic		Code		
Method	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	MuSiQue	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRe	LCC	RB-P	**Avg.**
LLaDA-8B-Instruct																	
Keep ratio $= 1.0$																	
Origin	16.65	32.99	28.44	12.39	14.30	6.42	29.86	19.09	27.10	35.54	60.41	32.51	2.00	39.50	69.71	64.69	30.73
dLLM-Cache	16.28	31.27	27.47	12.76	13.59	6.74	22.93	18.89	25.26	33.94	57.36	32.72	1.00	40.00	69.03	64.91	29.63
Fast-dLLM	14.99	29.73	27.20	12.90	13.57	6.16	26.09	18.88	25.52	37.72	62.01	32.96	1.00	40.50	68.70	65.06	30.19
Keep ratio $= 0.5$																	
Sparse-dLLM	17.54	32.99	29.12	12.52	13.35	6.37	26.52	18.34	**25.64**	**43.42**	**64.30**	**32.73**	1.50	41.00	69.29	**64.44**	31.19
Ours	**17.69**	**34.24**	**29.36**	**12.72**	**13.98**	**6.56**	**27.06**	**18.57**	25.50	41.21	64.28	32.14	1.50	41.00	**69.69**	64.11	**31.23**
Keep ratio $= 0.1$																	
Sparse-dLLM	14.92	29.20	29.25	13.29	14.61	5.95	19.19	16.41	17.67	35.96	**67.99**	34.57	1.50	41.00	63.00	60.47	29.06
Ours	**17.64**	**35.42**	**29.46**	**13.86**	**15.20**	**6.49**	**25.47**	**18.69**	**25.57**	**45.12**	67.22	**35.97**	1.50	41.00	**69.75**	**64.35**	**32.04**
Dream-v0-Instruct-7B																	
Keep ratio $= 1.0$																	
Origin	22.47	29.29	34.58	34.14	32.12	15.83	20.23	20.11	18.36	34.62	89.11	37.13	0.00	24.50	60.07	58.83	33.21
dLLM-Cache	21.06	28.03	34.27	31.95	29.15	13.94	17.82	18.88	14.68	33.17	87.38	35.36	1.00	23.50	56.03	55.12	31.33
Fast-dLLM	20.67	27.39	32.87	29.03	26.44	13.63	21.49	16.59	24.60	30.79	86.60	35.08	2.00	26.50	62.19	60.37	32.27
Keep ratio $= 0.5$																	
Sparse-dLLM	20.25	27.06	33.95	31.21	**30.38**	12.98	20.78	17.65	24.53	26.39	**87.25**	34.64	**4.50**	25.50	**63.13**	60.15	32.52
Ours	**21.65**	**29.83**	**34.02**	**33.95**	29.91	**14.11**	**21.31**	**19.39**	**24.90**	**33.92**	86.03	**39.11**	2.00	**26.50**	60.17	**60.42**	**33.58**
Keep ratio $= 0.1$																	
Sparse-dLLM	18.03	20.70	33.06	33.39	**32.61**	12.26	15.44	17.57	19.70	**42.92**	**87.15**	34.67	**2.50**	26.00	57.42	**56.86**	31.89
Ours	**21.48**	**28.31**	**34.02**	**35.63**	30.53	**13.90**	**19.27**	**19.49**	**23.17**	41.75	86.14	**40.05**	2.00	**26.50**	**58.11**	55.89	**33.52**

2018), PIQA (Bisk et al., 2020), and GPQA (Rein et al., 2023). Following Sparse-dLLM (Song et al., 2026), the multiple-choice tasks are evaluated generatively with OpenCompass (OpenCompass Contributors, 2023), while the remaining tasks use lm-eval (Gao et al., 2024). We report pass@1 for HumanEval and MBPP, accuracy for the multiple-choice tasks, math-verify for MATH, and strict-match for GSM8K to ensure a strict evaluation. We compare Ask-dLLM with the unmodified backbone without caching (*Origin*), dLLM-Cache (Liu et al., 2026), Fast-dLLM (Wu et al., 2026), and Sparse-dLLM (Song et al., 2026). Eviction methods are evaluated at keep ratios $\rho \in \{0.5, 0.1\}$. The backbone remains frozen and only the Ask-dLLM scorer is trained. We use a block length of 32 and set the number of denoising steps to the generation length. The total context length is 4096 for LLaDA and 2048 for Dream, except in Section 4.3, which varies it from 2K to 8K. All methods use identical prompts, seeds, decoding settings, and evaluation harnesses within each task. Throughput and peak memory are measured on a single NVIDIA RTX A6000(48GB), except for the scorer-size analysis in Table 8, which uses an NVIDIA A100-SXM4-40GB. Per-task benchmark and generation settings are provided in Appendix A.1, and scorer training data and configuration are detailed in Appendix A.2.

4.2 Main Results

Long-context performance. Table 1 compares results across 16 LongBench datasets. At $\rho = 0.1$, Ask-dLLM achieves average scores of 32.04 on LLaDA and 33.52 on Dream, exceeding the corresponding Origin scores of 30.73 and 33.21. Under the same budget, Sparse-dLLM scores 29.06 and 31.89, respectively. As the keep ratio decreases from $\rho = 0.5$ to $\rho = 0.1$, the average margin

Table 2: **General benchmark performance under aggressive eviction.** Results on code generation, multiple-choice reasoning, and mathematical reasoning under the default experimental configuration. GSM8K is reported with strict-match. Avg. denotes the mean across the seven tasks. The best eviction result at each keep ratio is shown in bold. Origin denotes the unmodified backbone without caching.

	Code		Multiple-choice			Math		
Method	HEval	MBPP	ARC-C	PIQA	GPQA	GSM8K	MATH	Avg.
LLaDA-8B-Instruct								
Keep ratio $= 1.0$								
Origin	37.20	36.20	86.70	84.66	36.36	54.28	38.20	53.37
dLLM-Cache	39.63	39.20	86.70	83.73	33.84	51.18	37.78	53.15
Fast-dLLM	36.59	36.80	86.35	83.73	33.33	47.69	36.16	51.52
Keep ratio $= 0.5$								
Sparse-dLLM	35.98	**36.60**	86.01	84.55	**37.88**	**51.33**	36.36	**52.67**
Ours	35.98	35.60	**86.44**	**84.82**	35.35	50.72	**37.00**	52.27
Keep ratio $= 0.1$								
Sparse-dLLM	12.80	24.40	74.16	66.05	33.84	12.66	15.90	34.26
Ours	**39.02**	**35.60**	**85.58**	**83.46**	**34.85**	**55.88**	**35.84**	**52.89**
Dream-v0-Instruct-7B								
Keep ratio $= 1.0$								
Origin	51.83	60.00	89.44	87.70	35.35	70.66	40.56	62.22
dLLM-Cache	50.61	56.40	89.44	87.98	32.83	69.45	39.64	60.91
Fast-dLLM	50.61	51.40	89.36	87.16	31.82	66.72	44.42	60.21
Keep ratio $= 0.5$								
Sparse-dLLM	**46.34**	52.80	**89.10**	86.45	32.32	67.93	**43.94**	**59.84**
Ours	42.07	52.80	88.42	**86.56**	**33.84**	**68.31**	41.68	59.10
Keep ratio $= 0.1$								
Sparse-dLLM	34.76	50.00	87.64	85.85	31.31	52.69	38.18	54.35
Ours	**47.56**	**52.60**	**87.98**	**86.56**	**33.84**	**60.27**	**40.70**	**58.50**

over Sparse-dLLM increases from 0.04 to 2.98 points on LLaDA and from 1.06 to 1.63 points on Dream. At $\rho = 0.1$, Ask-dLLM outperforms Sparse-dLLM on 13 of the 16 datasets for LLaDA and 11 of the 16 datasets for Dream.

General benchmark performance. Table 2 compares results across seven reasoning and code-generation benchmarks. On LLaDA, reducing ρ from 0.5 to 0.1 changes the average score of Ask-dLLM from 52.27 to 52.89, whereas Sparse-dLLM drops from 52.67 to 34.26. At $\rho = 0.1$, the average margin is 18.63 points, with gains of 43.22, 26.22, and 19.94 points on GSM8K, HumanEval, and MATH, respectively. On Dream, the same budget reduction changes the average score of Ask-dLLM from 59.10 to 58.50 and that of Sparse-dLLM from 59.84 to 54.35, yielding a margin of 4.15 points at $\rho = 0.1$. At this keep ratio, Ask-dLLM scores 0.48 points below Origin on LLaDA and 3.72 points below Origin on Dream.

Together, these results show that current-attention-based eviction incurs greater performance degradation as the cache budget shrinks, whereas predicting denoised-state relevance from the masked state better preserves the original backbone's long-context, reasoning, and code-generation capabilities. This supports the effectiveness of a selection criterion that anticipates denoised-state attention patterns, even when retaining only 10% of the external KV entries. Additional results for the Denoised-State Reference are provided in Appendix C.1.

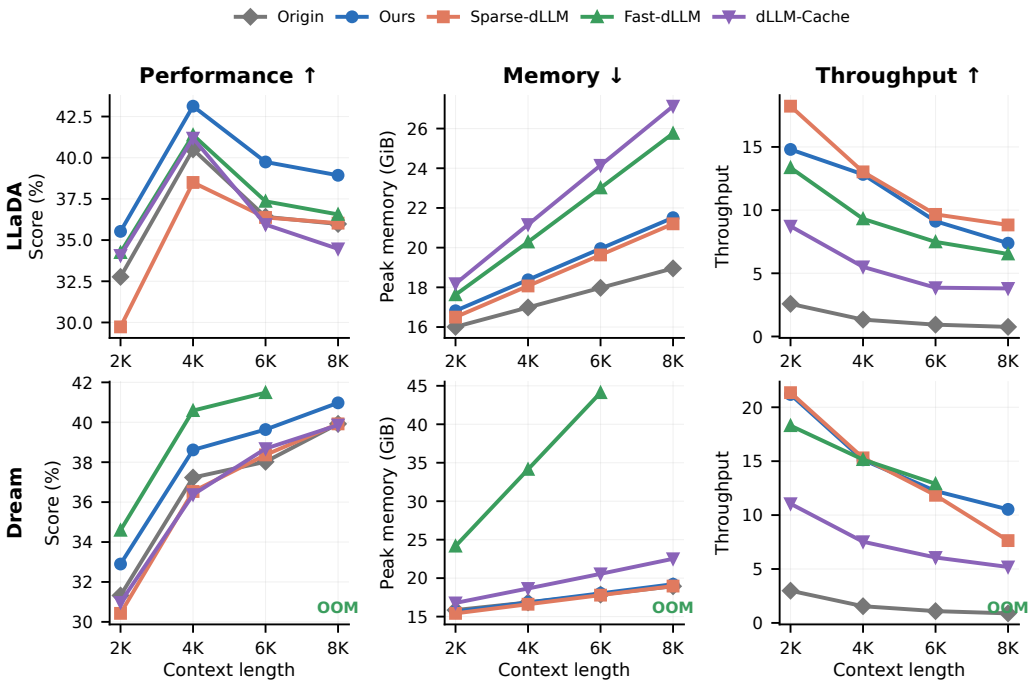

Figure 3: **Performance and efficiency across context lengths.** Mean task score, mean peak GPU memory, and throughput (generated tokens per second) over four LongBench datasets: GovReport, Qasper, LCC, and Passage Retrieval. The total context length increases from 2K to 8K. Results are shown for LLaDA-8B-Instruct (top) and Dream-v0-Instruct-7B (bottom). Origin denotes the unmodified backbone without caching. Ask-dLLM and Sparse-dLLM retain 10% of external KV entries, whereas Fast-dLLM and dLLM-Cache apply no eviction. OOM indicates an out-of-memory failure.

4.3 Scaling with Context Length

We vary the context length from 2K to 8K using the complete evaluation sets of GovReport, Qasper, LCC, and Passage Retrieval; all other settings follow Section 4.1. As shown in Figure 3, Ask-dLLM matches or exceeds Origin in mean task score while substantially increasing throughput, reaching approximately $9.7\times$ and $11.8\times$ that of Origin at 8K on LLaDA and Dream, respectively. Despite retaining only 10% of external KV entries, it exceeds the non-evicting dLLM-Cache baseline by 1.47–4.47 points on LLaDA and 0.96–2.24 points on Dream across the evaluated context lengths. It also outperforms Fast-dLLM on LLaDA, while remaining within approximately 2 points of Fast-dLLM on Dream at 2K–6K. At 8K, Fast-dLLM runs out of memory on Dream, whereas Ask-dLLM completes the evaluation. Compared with these non-evicting caching methods, Ask-dLLM uses less memory, with savings increasing at longer contexts. Under the same cache budget, it consistently outperforms Sparse-dLLM with similar memory usage. Per-dataset results are provided in Appendix C.3.

4.4 Fidelity to the Origin Model

Figure 4 examines whether predicting denoised-state relevance helps preserve the outputs and task performance of Origin under eviction. On both backbones, Ask-dLLM maintains higher Token F1 than Sparse-dLLM, with the gap widening as the keep ratio decreases. This divergence is accompanied by an increasing gap in task performance, particularly at $\rho \in \{0.1, 0.05\}$. These results support denoised-state relevance as an effective selection signal for preserving Origin's output content and task performance under aggressive eviction, compared with selection based on current-state attention.

4.5 Generalization Across Block Sizes

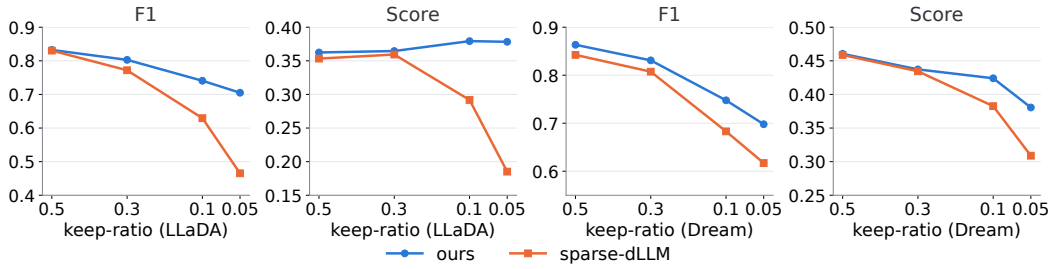

Figure 4: **Output fidelity and task performance under decreasing cache budgets.** Token F1 measures the similarity between outputs generated by each eviction method and those of Origin, while Score denotes the mean task performance over 100 examples each from Qasper, MuSiQue, SAMSum, and GSM8K. From left to right, the panels report Token F1 and task score for LLaDA-8B-Instruct and Dream-v0-Instruct-7B.

Table 3: **Generalization across inference block sizes.** Scorers trained at $B_{\text{train}} = 32$ are reused without fine-tuning at $\rho = 0.1$. Fast denotes Fast-dLLM without KV eviction; Sparse denotes Sparse-dLLM. The better of the two eviction methods is in bold.

	$B_{\text{test}} = 16$			$B_{\text{test}} = 64$		
Dataset	Fast	Sparse	Ours	Fast	Sparse	Ours
LLaDA-8B-Instruct						
Qasper	33.00	33.05	**35.22**	29.24	27.59	**32.96**
NarrativeQA	14.98	14.76	**16.10**	14.99	16.30	**18.17**
LCC	68.97	65.14	**69.88**	67.79	48.47	**58.97**
MBPP	37.80	26.00	**37.80**	27.20	18.00	**23.00**
Dream-v0-Instruct-7B						
Qasper	28.03	25.59	**28.32**	28.95	22.42	**28.33**
NarrativeQA	22.29	19.85	**21.34**	21.89	18.05	**21.32**
LCC	61.38	57.22	**58.90**	60.94	54.62	**55.75**
MBPP	54.60	47.40	**51.80**	55.00	53.40	**54.40**

We evaluate generalization across block sizes on Qasper, NarrativeQA, LCC, and MBPP, selected from datasets with generation lengths of at least 64 tokens. Scorers trained at $B_{\text{train}} = 32$ are reused at $B_{\text{test}} \in \{16, 64\}$ without fine-tuning. All methods use the same inference block size, with $\rho = 0.1$ for Ask-dLLM and Sparse-dLLM. Other settings follow the default experimental configuration, and results at $B = 32$ are reported in Tables 1 and 2.

As shown in Table 3, Ask-dLLM outperforms Sparse-dLLM across both backbones and block sizes on all four evaluated datasets. At $B_{\text{test}} = 64$, the gains reach 10.50 points on LCC with LLaDA and 5.91 points on Qasper with Dream. Despite retaining only 10% of external KV entries, Ask-dLLM also surpasses Fast-dLLM without eviction in several settings, including Qasper and NarrativeQA on LLaDA at both block sizes. These results show that the learned cache ranking transfers to block sizes unseen during training, preserving task performance under a limited cache budget.

5 Conclusion

We introduced Ask-dLLM, which learns from denoised-state attention to predict denoised-state relevance conditioned on the current masked block. Across 23 benchmarks, retaining only 10% of external KV entries improves the overall mean score over the prior eviction baseline by 7.75 points on LLaDA and 2.39 points on Dream, while remaining within 1 point of Origin on both backbones. On four LongBench datasets at an 8K context length, Ask-dLLM achieves approximately $9.7\times$ and $11.8\times$ the throughput of Origin, respectively, while matching or exceeding its mean task score. These results demonstrate that predicting denoised-state relevance enables aggressive cache eviction while preserving task performance and accelerating dLLM inference.

Limitations. Backbone-specific supervision requires retraining the scorer for each new backbone. However, training is limited to a lightweight scorer while the backbone remains frozen, keeping the additional training cost modest. Appendix A.3 reports the one-time cost of teacher target extraction and scorer training for each backbone.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing, including grammar correction and improving the clarity of expression, and to assist with code implementation. We have reviewed all AI-assisted work. All AI-assisted code was verified and tested for correctness by the authors, and all experimental results reported in this paper were produced and validated by the authors. The research ideas, method design, and experimental analysis are our own. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies inference-time cache eviction for publicly available diffusion language models using standard public benchmarks. It does not involve human subjects, new data collection, or personally identifiable information. We do not foresee direct negative societal impacts beyond those already associated with the underlying language models.

REPRODUCIBILITY STATEMENT

The method is described in Section 3, and the scorer architecture and training objective are given in Section 3.3. Per-task benchmark and generation settings are provided in Appendix A.1, and scorer training data and configuration are detailed in Appendix A.2. Per-dataset context-scaling results are reported in Appendix C.3. All backbones and benchmarks used are publicly available. Source code is included in the supplementary materials.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Stickland, Tomek Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *International Conference on Learning Representations*, volume 2024, pp. 18623–18642, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, 2005.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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. *arXiv preprint arXiv:1803.05457*, 2018.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in neural information processing systems*, 36:70293–70332, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

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. *NeurIPS*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed Abdelfattah, Jae-sun Seo, Zhiru Zhang, and Udit Gupta. Flashdlm: Accelerating diffusion language model inference via efficient kv caching and guided diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 153529–153544, 2026.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-cache: Accelerating diffusion large language models with adaptive caching, 2026. URL https://openreview.net/forum?id=DriG3hgh42.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *Advances in Neural Information Processing Systems*, 38:149009–149033, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

OpenCompass Contributors. OpenCompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33038–33046, 2026.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *Advances in Neural Information Processing Systems*, 38:138867–138907, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:56729–56762, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, 2026.

APPENDIX CONTENTS

A Experimental Setup . . . 14
A.1 Benchmarks and Generation Settings . . . 14
A.2 Scorer Training Data and Configuration . . . 14
A.3 Offline Adaptation Cost . . . 15
B Method Details . . . 16
B.1 Algorithm . . . 16
C Additional Results and Analyses . . . 16
C.1 Denoised-State Reference Analysis . . . 16
C.2 Scorer Size Trade-off . . . 16
C.3 Per-Dataset Context-Length Scaling . . . 18
D Qualitative Examples . . . 22
D.1 Full Decoding Trace . . . 22

Table 4: Benchmark configurations. *Gen.* is the generation length N_G, and *Blocks* is $M = N_G/32$, the number of decoding blocks per example. *Shots* counts the in-context exemplars added by the evaluation harness; the LongBench few-shot tasks (TREC, TriviaQA, SAMSum) already contain exemplars in their inputs. LongBench rows follow the column order of the main LongBench table, which abbreviates NarrativeQA as NrtvQA, MultiFieldQA-en as MF-en, PassageCount as PCount, PassageRetrieval-en as PRe, and RepoBench-P as RB-P.

Dataset	Task	Gen.	Blocks	Shots	Metric
LongBench					
NarrativeQA	long-narrative QA	128	4	0	QA-F1
Qasper	paper-grounded QA	128	4	0	QA-F1
MultiFieldQA-en	long-document QA	64	2	0	QA-F1
HotpotQA	multi-hop QA	32	1	0	QA-F1
2WikiMQA	multi-hop QA	32	1	0	QA-F1
MuSiQue	multi-hop QA	32	1	0	QA-F1
GovReport	report summarization	512	16	0	ROUGE-L
QMSum	meeting summarization	512	16	0	ROUGE-L
MultiNews	multi-doc summarization	512	16	0	ROUGE-L
TREC	question classification	64	2	0	class. acc.
TriviaQA	short-answer QA	32	1	0	QA-F1
SAMSum	dialogue summarization	128	4	0	ROUGE-L
PassageCount	passage counting	32	1	0	count acc.
PassageRetrieval-en	passage retrieval	32	1	0	retr. acc.
LCC	code completion	64	2	0	edit sim.
RepoBench-P	repo-level completion	64	2	0	edit sim.
Reasoning and code generation					
GSM8K	grade-school math	256	8	5	strict-match
MATH	competition math	256	8	4	math-verify
HumanEval	Python generation	512	16	0	pass@1
MBPP	Python programs	512	16	3	pass@1
Multiple choice (generated answer; option letter extracted)					
ARC-Challenge	grade-school science	256	8	0	acc.
PIQA	physical commonsense	256	8	0	acc.
GPQA	graduate science	256	8	5	acc.

A Experimental Setup

A.1 Benchmarks and Generation Settings

Table 4 lists every benchmark used in this work. For all generative tasks we set the number of denoising steps equal to the generation length N_G, so exactly one token is finalized per step, and we fix the block length to $B = 32$; the number of decoding blocks per example is therefore $M = N_G/32$, ranging from one block on the short-answer tasks to sixteen on the long summarization and code-generation tasks.

A.2 Scorer Training Data and Configuration

Adapting Ask-dLLM to a backbone uses prompts drawn from five sources, listed in Table 5, for a total of $1{,}621$ prompts. The same composition is used for both backbones. The sources span mathematical reasoning, program synthesis, long-document summarization, and multi-hop question answering, so that the teacher targets cover both short and long generation lengths. None of the prompts overlaps with the evaluation splits of Table 4.

Per-KV-head teacher targets are extracted once from these prompts, and the scorer is then trained for 10 epochs. The backbone weights remain frozen during both target extraction and scorer training. The Base scorer uses projection dimension 256 and MLP hidden dimension 512; its capacity variants are evaluated in Appendix C.2. Within each layer, the projections and MLP trunk are shared across KV heads, while the final readout produces a separate ranking score for each KV head.

Table 5: Prompt sources used for backbone adaptation. The same composition is used for LLaDA-8B-Instruct and Dream-v0-Instruct-7B.

Source	Domain	Prompts
MATH (five subjects)	mathematical reasoning	500
MBPP (full)	program synthesis	371
GovReport	report summarization	150
MultiNews	multi-document summarization	100
MuSiQue	multi-hop question answering	500
Total		**1,621**

Table 6: One-time backbone adaptation cost, measured on a single NVIDIA RTX A6000 with the prompt set of Table 5. Teacher extraction and scorer training are measured separately; total cost is their sum, and peak VRAM is the maximum across the two stages.

	Offline cost (GPU-h)			
Backbone	Teacher extraction	Scorer training	**Total**	Peak VRAM (GiB)
LLaDA-8B-Instruct	2.148	5.442	**7.590**	19.88
Dream-v0-Instruct-7B	1.941	3.885	**5.826**	16.07

Teacher target construction and the training objective are specified in Sections 3.2 and 3.3, respectively. The offline and inference procedures are summarized in Appendix B.1.

A.3 Offline Adaptation Cost

Adapting Ask-dLLM to a backbone is a one-time offline procedure: teacher targets are extracted once, the scorer is trained once, and inference afterwards adds only the scorer itself. Table 6 reports the wall-clock cost of that procedure for both backbones on a single NVIDIA RTX A6000, under the extraction and training settings of Appendix A.2. Adaptation takes under eight GPU-hours and peaks at 19.88 GiB, so it fits on one commodity 48 GiB card and need not be repeated per task or per keep ratio.

B METHOD DETAILS

B.1 ALGORITHM

Algorithms 1 and 2 summarize offline teacher supervision collection and inference with Ask-dLLM. Both procedures use the same frozen backbone and block-wise decoding configuration, and both index blocks as $b = 0, \ldots, M-1$, matching Figures 1 and 7.

INITIALSTEP performs the first denoising step using a full-sequence forward. The subsequent FULLFORWARD is the cache-refresh forward, returning layer-input hidden representations, KV states, and decoding logits $\mathbf{z}$. DECODEBLOCK first updates the current block using $\mathbf{z}$, then completes the remaining denoising steps using block-only forwards with the supplied external KV states held fixed. These routines follow the backbone-specific decoding schedule, including any required logit alignment.

DENOISEDATTENTION performs an additional block-only forward once the current block reaches its denoised state, retaining the external KV states from the cache-refresh forward. Attention is normalized over external and current-block positions before extracting the external-candidate columns. AGGREGATERELEVANCE averages over query heads sharing a KV head and then takes the maximum over current-block query positions. For MHA, the query-head aggregation reduces to the identity.

We apply Algorithm 1 to the training prompts and combine the resulting block-level records. Rather than storing hidden states, each record stores the token sequence entering the cache-refresh forward. During training, the frozen backbone replays this input to reconstruct the masked-state representations used by the scorer.

Each layer-specific scorer is trained separately using the listwise and pairwise objectives. Teacher relevance is normalized by its sum over candidates, while scorer outputs are normalized with softmax. The listwise KL is summed over candidates and averaged over valid heads; the pairwise loss is averaged over all valid sampled head–pair comparisons, excluding pairs with equal teacher relevance.

For readability, Algorithm 2 shows scoring and selection after FULLFORWARD; the implementation performs them within the cache-refresh forward as each layer constructs its cache. The current block is excluded from eviction, and retained entries preserve their original positional identities. The cache is reconstructed for every new block, allowing previously evicted positions to be reconsidered without accumulating scores across blocks.

C ADDITIONAL RESULTS AND ANALYSES

C.1 DENOISED-STATE REFERENCE ANALYSIS

The Denoised-State Reference (DSR) selects cache entries using attention measured after the current block reaches its denoised state. Although this information is unavailable at inference time, DSR isolates the value of selecting entries according to their use by the denoised block rather than their affinities in the masked state.

Table 7 reports four benchmarks for both backbones at $\rho \in \{0.5, 0.1\}$, without filtering tasks according to their outcomes. DSR is not expected to reproduce Origin exactly, because generation is repeated after most external KV entries have been removed. The table compares DSR and Ask-dLLM under the same cache budget. On LLaDA, Ask-dLLM exceeds DSR on all four benchmarks at both keep ratios. On Dream, the two are close: at $\rho = 0.1$, DSR is ahead on GSM8K and MuSiQue, Ask-dLLM is ahead on HotpotQA, and the two tie on HumanEval. Across all 16 comparisons, Ask-dLLM exceeds DSR in 11 and ties in one, using only masked-state representations. This confirms that DSR is an analysis reference rather than a strict performance upper bound.

C.2 SCORER SIZE TRADE-OFF

Table 8 examines the effect of scorer capacity while keeping the teacher data and training configuration fixed. Increasing the scorer size does not lead to a consistent improvement in task performance. The Base scorer achieves the highest average score of 43.50, whereas the Large scorer increases the

Algorithm 1 Offline teacher supervision collection

Require: Frozen dLLM f; prompt p; generation length N_G; block length B
procedure COLLECTSUPERVISION(f, p, N_G, B)
  $M \leftarrow N_G/B$
  $x \leftarrow [p;\ \texttt{[MASK]}^{N_G}]$
  $\mathcal{D} \leftarrow \emptyset$
  for $b = 0, \ldots, M-1$ **do**
    $\mathcal{C}_b \leftarrow \mathcal{I}_P \cup \mathcal{I}_{<b} \cup \mathcal{I}_{>b}$
    $x \leftarrow$ INITIALSTEP($f, x, \mathcal{I}_b$)
    $x_{\text{mask}} \leftarrow$ COPY(x)
    $(\mathbf{H}, \mathbf{K}, \mathbf{V}, \mathbf{z}) \leftarrow$ FULLFORWARD($f, x, \mathcal{I}_b$)
    $(\mathbf{K}_{\text{ext}}, \mathbf{V}_{\text{ext}}) \leftarrow (\mathbf{K}[\mathcal{C}_b], \mathbf{V}[\mathcal{C}_b])$
    $x \leftarrow$ DECODEBLOCK($f, x, \mathcal{I}_b, \mathbf{z}, \mathbf{K}_{\text{ext}}, \mathbf{V}_{\text{ext}}$)
    $\boldsymbol{\alpha} \leftarrow$ DENOISEDATTENTION($f, x, \mathcal{I}_b, \mathbf{K}_{\text{ext}}, \mathbf{V}_{\text{ext}}$)
    $\mathbf{r} \leftarrow$ AGGREGATERELEVANCE($\boldsymbol{\alpha}$)
    Store $(x_{\text{mask}}, \mathcal{I}_b, \mathcal{C}_b, \mathbf{r})$ in $\mathcal{D}$
  end for
  return $\mathcal{D}$
end procedure

Algorithm 2 Inference with Ask-dLLM

Require: Frozen dLLM f; trained scorer S_θ; prompt p
Require: Generation length N_G; block length B; keep ratio ρ
procedure GENERATE($f, S_\theta, p, N_G, B, \rho$)
  $M \leftarrow N_G/B$
  $x \leftarrow [p;\ \texttt{[MASK]}^{N_G}]$
  for $b = 0, \ldots, M-1$ **do**
    $\mathcal{C}_b \leftarrow \mathcal{I}_P \cup \mathcal{I}_{<b} \cup \mathcal{I}_{>b}$
    $x \leftarrow$ INITIALSTEP($f, x, \mathcal{I}_b$)
    $(\mathbf{H}, \mathbf{K}, \mathbf{V}, \mathbf{z}) \leftarrow$ FULLFORWARD($f, x, \mathcal{I}_b$)
    for $\ell = 1, \ldots, L$ **do**
      $\mathbf{s}^{(\ell,b)} \leftarrow S_\theta^{(\ell)}(\mathbf{H}^{(\ell,b)}, \mathcal{I}_b, \mathcal{C}_b)$
      for $h = 1, \ldots, N_h$ **do**
        $\widehat{\mathcal{S}}_b^{(\ell,h)} \leftarrow \mathrm{TopK}_{j \in \mathcal{C}_b}\big(s_{h,j}^{(\ell,b)},\ \lfloor \rho |\mathcal{C}_b| \rfloor\big)$
        $(\widetilde{\mathbf{K}}_h^{(\ell)}, \widetilde{\mathbf{V}}_h^{(\ell)}) \leftarrow (\mathbf{K}_h^{(\ell)}[\widehat{\mathcal{S}}_b^{(\ell,h)}], \mathbf{V}_h^{(\ell)}[\widehat{\mathcal{S}}_b^{(\ell,h)}])$
      end for
    end for
    $x \leftarrow$ DECODEBLOCK($f, x, \mathcal{I}_b, \mathbf{z}, \widetilde{\mathbf{K}}, \widetilde{\mathbf{V}}$)
  end for
  return $x[\mathcal{I}_0 \cup \cdots \cup \mathcal{I}_{M-1}]$
end procedure

Table 7: **Comparison with the Denoised-State Reference.** We report four benchmarks for LLaDA and Dream at both keep ratios, without filtering by outcome. Origin denotes the unmodified backbone without caching. DSR selects entries using denoised-state attention and is available only for analysis, whereas Ask-dLLM uses masked-state representations at inference. HumanEval reports pass@1 and GSM8K uses strict-match.

		$\rho = 0.5$		$\rho = 0.1$	
Dataset	Origin	DSR	Ask-dLLM	DSR	Ask-dLLM
LLaDA-8B-Instruct					
HumanEval	37.20	34.76	35.98	35.98	39.02
GSM8K	54.28	48.29	50.72	55.65	55.88
HotpotQA	12.39	12.47	12.72	12.99	13.86
MuSiQue	6.42	6.50	6.56	6.15	6.49
Dream-v0-Instruct-7B					
HumanEval	51.83	42.68	42.07	47.56	47.56
GSM8K	70.66	69.22	68.31	62.93	60.27
HotpotQA	34.14	33.78	33.95	33.57	35.63
MuSiQue	15.83	13.61	14.11	14.11	13.90

Table 8: Scorer size trade-off on LLaDA-8B-Instruct at $\rho = 0.1$. All configurations use the same teacher data and training recipe. Latency measures one scorer pass over all 32 layers, and added VRAM includes scorer parameters and peak temporary memory. Avg. is the mean over the three tasks. GSM8K uses strict-match, and HEval (HumanEval) and MBPP report pass@1.

					Task performance			
Size	proj / mlp	Params (M)	Latency (ms)	VRAM (MiB)	GSM8K	HEval	MBPP	Avg.
Small	128 / 256	36.98	26.602	272.1	54.89	37.80	36.60	43.10
Base	256 / 512	80.25	40.102	455.1	55.88	39.02	35.60	**43.50**
Large	512 / 1024	185.66	79.606	845.2	55.50	34.76	37.60	42.62

parameter count by $2.31\times$, scoring latency by $1.99\times$, and added VRAM by $1.86\times$ while scoring 0.88 points lower on average.

The Small scorer provides a more aggressive efficiency option. Compared with Base, it reduces the parameter count by 53.9%, scoring latency by 33.7%, and added VRAM by 40.2%, while decreasing the average score by 0.40 points. These results show that increasing scorer capacity alone does not consistently improve task performance under the evaluated setting. We use the Base configuration in the main experiments as a balanced default and report Small as a lightweight alternative.

C.3 Per-Dataset Context-Length Scaling

Figure 5 provides the per-dataset results for LLaDA-8B-Instruct. Figure 6 provides the corresponding results for Dream-v0-Instruct-7B. These figures decompose the aggregate results in Figure 3. Across both backbones, Ask-dLLM and Sparse-dLLM exhibit closely matched peak GPU memory usage. Their relative task performance and throughput, however, vary with the dataset, backbone, and context length.

LLaDA. Ask-dLLM consistently improves over Sparse-dLLM on GovReport, LCC, and Qasper, and matches or exceeds it on Passage Retrieval. The throughput trade-off is dataset-dependent: on GovReport, which generates up to 512 tokens, Ask-dLLM has lower throughput at most context lengths while achieving substantially higher task scores. On LCC, which generates 64 tokens over two blocks, it achieves higher task scores with comparable or higher throughput. For Qasper and Passage Retrieval, throughput is generally close at 4K–8K, whereas Sparse-dLLM is faster at 2K. Thus, longer generation involves more scorer invocations, but generation length alone does not explain the observed throughput differences.

Dream. Ask-dLLM achieves higher task scores than Sparse-dLLM on GovReport and Qasper across all evaluated context lengths, while the two methods perform similarly on Passage Retrieval. On LCC, Ask-dLLM leads at 2K–4K, whereas Sparse-dLLM has a small advantage at 6K–8K. Throughput is generally comparable at shorter contexts, and Ask-dLLM is faster than Sparse-dLLM on all four datasets at 8K, without a substantial increase in peak memory. Fast-dLLM remains competitive in task performance, particularly on GovReport, LCC, and Qasper, but incurs substantially higher memory usage and runs out of memory at 8K on all four datasets. In contrast, Ask-dLLM completes all evaluated context lengths.

Overall, the per-dataset results show that Ask-dLLM improves cache selection quality over Sparse-dLLM on a broad range of tasks while preserving a similar memory footprint. The throughput trade-off is not uniform across workloads: LLaDA exhibits a quality–throughput trade-off on some datasets, whereas Dream achieves higher throughput than Sparse-dLLM at the longest evaluated context length.

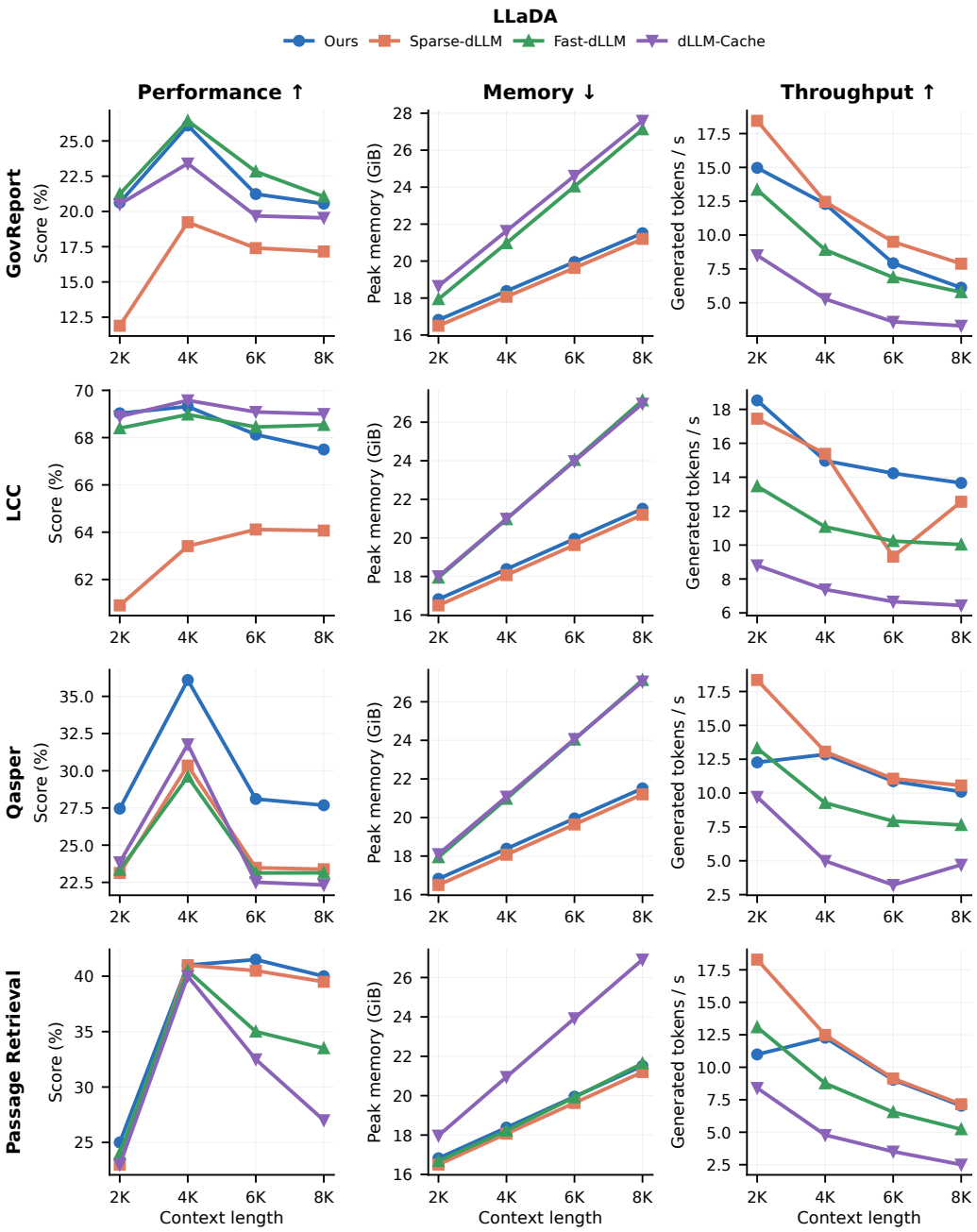

Figure 5: Per-dataset context-length scaling on LLaDA-8B-Instruct. Rows show GovReport, LCC, Qasper, and Passage Retrieval; columns report task score, peak GPU memory (GiB), and throughput (generated tokens per second) as the total context length increases from 2K to 8K. Ask-dLLM and Sparse-dLLM use an external-cache keep ratio of $\rho = 0.1$, whereas Fast-dLLM and dLLM-Cache run without KV eviction. Higher task score and throughput are better; lower peak memory is better.

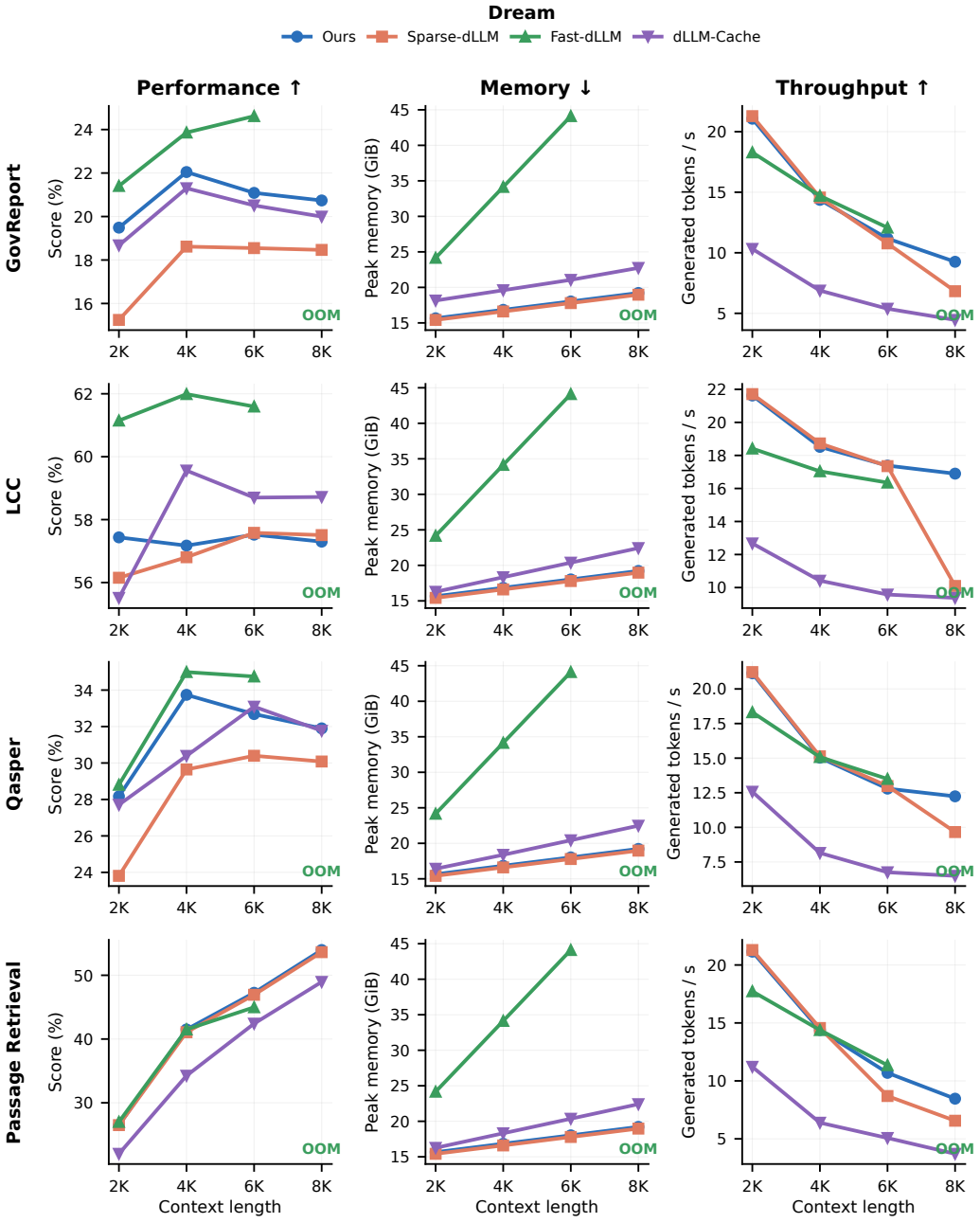

Figure 6: Per-dataset context-length scaling on Dream-v0-Instruct-7B. Rows show GovReport, LCC, Qasper, and Passage Retrieval; columns report task score, peak GPU memory (GiB), and throughput (generated tokens per second) as the total context length increases from 2K to 8K. Ask-dLLM and Sparse-dLLM use an external-cache keep ratio of $\rho = 0.1$, whereas Fast-dLLM and dLLM-Cache run without KV eviction. OOM denotes an out-of-memory failure: Fast-dLLM cannot complete the 8K evaluation on any of the four datasets. Higher task score and throughput are better; lower peak memory is better.

D QUALITATIVE EXAMPLES

D.1 FULL DECODING TRACE

Figure 7 follows the GSM8K question of Figure 1 through the seven blocks in which the answer is produced (blocks 0–6; blocks are indexed from 0 as in Algorithms 1 and 2) under the three eviction methods, so that the retained cache and the text it produces can be read side by side. Each block is decoded over 32 denoising steps against a cache that is pruned once, after the second step; the heatmap for a block therefore shows the cache that block sees for the remainder of its decoding. Rows are the 32 query positions of the block, columns are retained cache entries, and values are attention at layer 26 as in Figure 1. Column labels are the retained tokens themselves, greyed when DSR did not retain them under the same budget.

Two things are visible across the seven blocks. First, the retained set changes from block to block for every method: entries that dominate one block's attention are absent from the next, illustrating the changing cache requirements across generation blocks. Second, the methods diverge in what kind of entry they keep. Sparse-dLLM scores an entry from the block's early masked state, and its retained columns are largely grey from block 3 onward: the quantities of the question drop out while entries not selected by DSR are kept. All three methods open by restating the problem (blocks 0–2) and eventually compute $60 - 15 = 45$; DSR and Ask-dLLM complete this step in block 3, whereas Sparse-dLLM completes it only in block 4. From block 3, however, Sparse-dLLM no longer holds both operands. In block 5 it sets out to subtract 15 a second time instead of 20, and in block 6 it answers `30`. DSR and Ask-dLLM keep both quantities, continue with $45 - 20$ in block 4, and answer `25`.

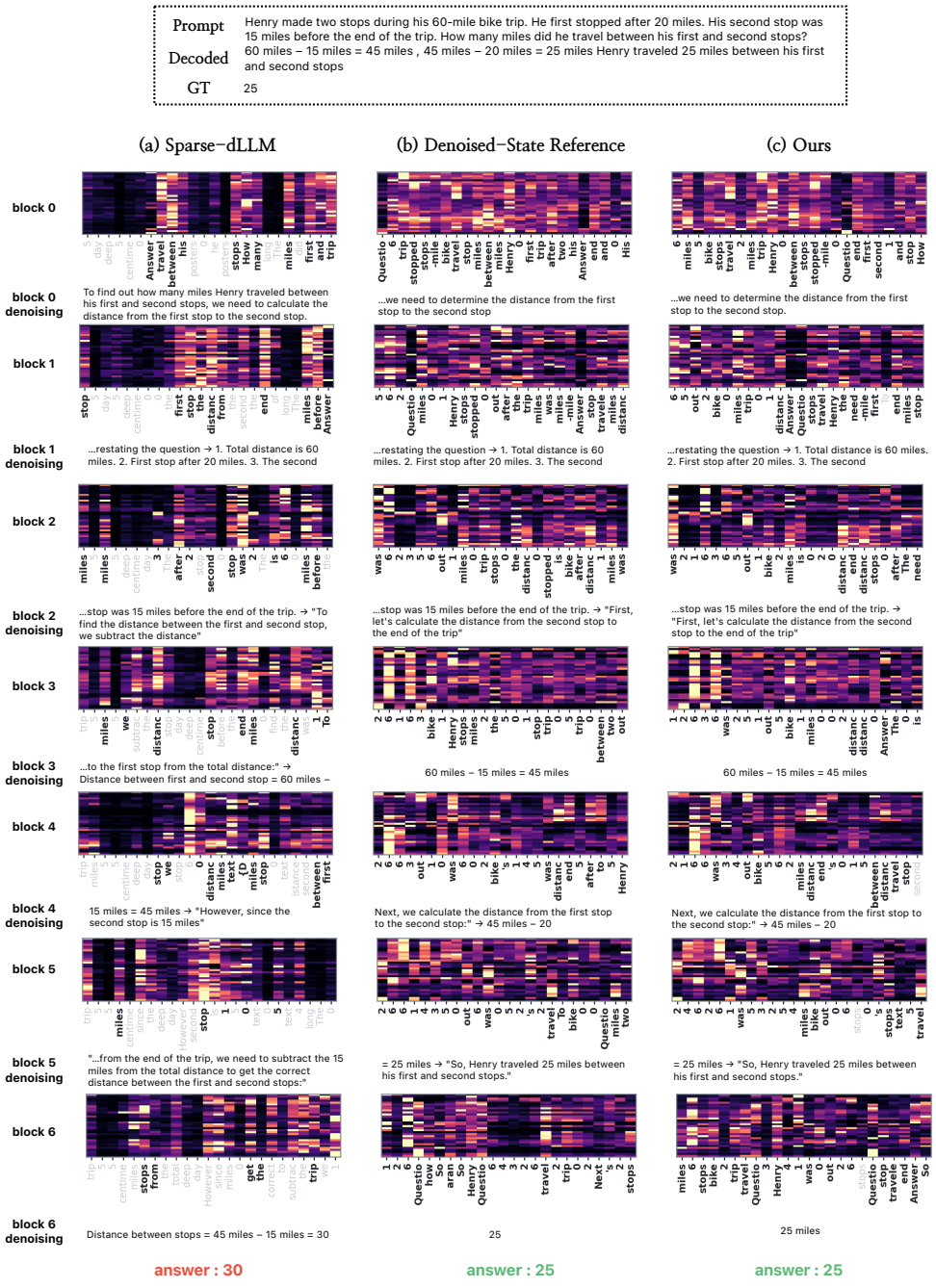

Figure 7: Block-by-block decoding trace for the GSM8K question of Figure 1 under Sparse-dLLM (a), the Denoised-State Reference (b), and Ask-dLLM (c). Blocks are indexed from 0. Each heatmap shows attention at layer 26 from the 32 query positions of a block to the cache entries retained for that block at $\rho = 0.1$; below it is the text decoded so far. Column labels are the retained tokens, greyed when (b) did not retain them. The prompt, the full-cache generation, and the ground truth are boxed at the top.