d^{2}Cache: Accelerating Diffusion-Based LLMs via Dual Adaptive Caching

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, Xu Yang
Southeast University, Nanjing University of Information Science and Technology, Qiyuan Tech

ICLR 2026 Poster | Diffusion LLMs

Core message: Diffusion LLMs cannot use standard KV cache because bidirectional attention makes one token update change the whole sequence context. d^{2}Cache solves this with a training-free, token-level adaptive cache that refreshes only the KV states that matter at each step.

Method:
Dual Adaptive Cache

Stage 1 selects masked tokens that are likely to change soon. Stage 2 keeps the prompt and decoded tokens with the highest attention influence.

3.5×
Average speedup
Across LLaDA-Inst and Dream-Inst over six benchmarks.

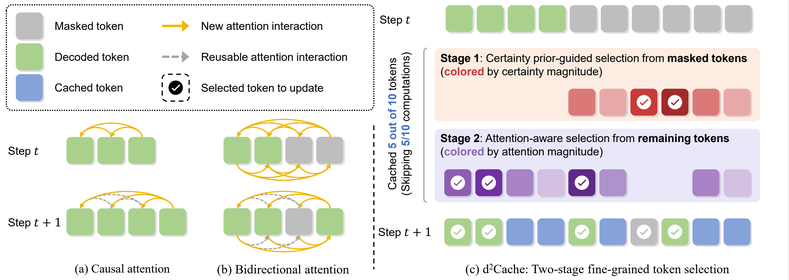
4.7×
Best-case speedup
Dream-Inst on GSM8K improves from 2.62 to 12.25 tok/s.

+2.0 pts
Quality gain
Average score rises from 51.4 to 53.4 on Dream-Inst.

Why dLLMs Are Hard to Cache

- Autoregressive models can reuse KV states because new tokens are appended under causal attention.
- Diffusion LLMs iteratively replace masked tokens inside a fixed-length sequence with **bidirectional attention**.
- Updating one token changes the context seen by all others, so naive KV reuse is invalid and full recomputation becomes expensive.
- Existing cache methods rely on coarse refresh windows or static/dynamic segments, which miss token-level dynamics.

From Full Recompute to Adaptive Refresh

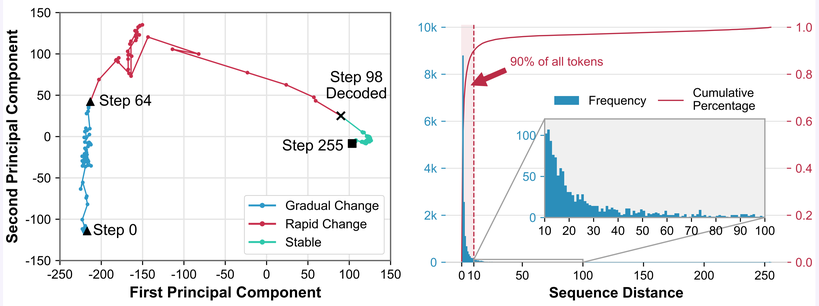


d^{2}Cache replaces coarse segment rules with token-level selection: refresh the small subset of tokens whose KV states are most valuable, and reuse the rest.

Three Empirical Observations

- **Masked tokens follow three phases:** gradual change → rapid change → stable. Only the rapid-change phase needs active refresh.
- **Decoding is local:** 90% of newly decoded tokens appear within distance 10 of the most recently decoded token.
- **Attention is sparse:** prompt and decoded tokens receive most of the attention, while masked tokens receive little.

Token Dynamics



Left: masked-token KV states change sharply only right before decoding. Right: consecutive decoded positions are usually close to each other, which motivates a local certainty prior.

d^{2}Cache Overview

d^{2}Cache splits tokens into **prompt**, **masked**, and **decoded** groups, then refreshes KV states with a two-stage strategy:

- **Stage 1:** select masked tokens that are likely to be decoded soon.
- **Stage 2:** among the remaining tokens, keep only those with the largest attention influence.
- Refresh KV states for the selected tokens and reuse cached KV states for everything else.

Stage 1: Certainty Prior-Guided Selection

For each masked token at position i, estimate how likely it is to be decoded soon from the density of nearby known tokens:

$$D(i) = \sum_{j=0}^{L-1} \exp\left(-\frac{|i-j|^2}{2\sigma^2}\right) \mathbb{I}_{\{j \notin M\}}$$

Combine this structural prior with the model confidence s^i and select the top-k masked tokens:

$$M^* = \underset{i \in M}{\arg\operatorname{top_k}} D(i) \cdot s^i$$

- Higher $D(i)$ means stronger local support from prompt or already decoded tokens.
- This also yields a more reliable **quasi left-to-right decoding order**.

Stage 2: Attention-Aware Selection

For prompt and decoded tokens, d^{2}Cache computes attention rollout and derives an influence score

$$c_j = \sum_{i=1}^{L} C_{ij}^{(N)}$$

Then it selects the smallest set of tokens whose cumulative influence exceeds threshold p.

- High-influence prompt/decoded tokens are refreshed.
- Low-influence tokens safely keep their cached KV states.

Why Selective Updates Win

- More computation is not always better for dLLMs.
- Updating masked tokens during the gradual-change phase wastes compute without helping quality.
- d^{2}Cache refreshes KV states only when token dynamics or attention concentration justify it.
- The method is **training-free** and does not change model parameters.

What Changes in Decoding

- Confidence-only NAR decoding tends to show unstable ordering and premature end-of-sequence overconfidence.
- Certainty prior-guided decoding is more stable because it prefers tokens supported by nearby known context.
- This makes d^{2}Cache not only a cache mechanism, but also a more reliable decoding strategy.

Experimental Setup

- Models: LLaDA-Inst and Dream-Inst
- Tasks: GSM8K, MBPP, HumanEval, Math-500, GPQA, MMLU-Pro
- Baselines: Vanilla, dLLM-Cache, Fast-dLLM
- Default settings: $\sigma = 10$, top-$k = 32$, $p = 0.1$

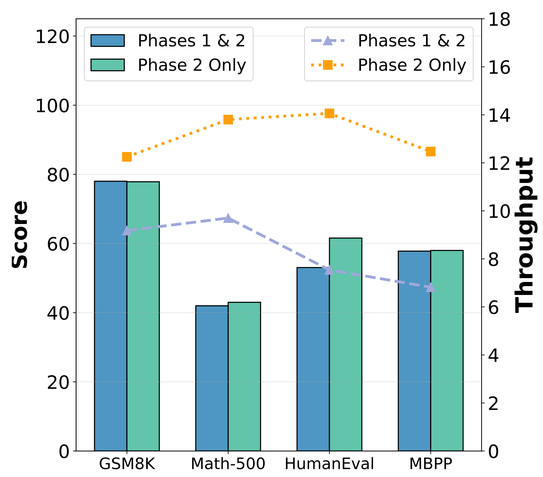
Main Results

Method	LLaDA Thr.	LLaDA Score	Dream Thr.	Dream Score
Vanilla	3.54	43.6	3.64	51.4
dLLM-Cache	8.16	43.8	7.64	50.9
Fast-dLLM	9.38	43.4	9.59	51.7
d^{2}Cache	**12.24**	**44.4**	**12.89**	**53.4**

d^{2}Cache delivers the best overall balance of **throughput**, **latency**, and **quality**.

- **Dream-Inst on GSM8K:** 2.62 → 12.25 tok/s, a **4.7**× speedup, while score improves from 76.7 to 78.2.
- **Against Fast-dLLM:** d^{2}Cache is usually both faster and more accurate.

Ablation: Update Only the Important Phase



Phase-2-only updates are faster than updating both phases, while keeping comparable or better quality.

Takeaways

- Diffusion LLM caching should be treated as a **fine-grained token selection problem**: d^{2}Cache uses **certainty prior** for masked tokens and **attention influence** for prompt/decoded tokens, reducing redundant computation while often improving decoding quality.