STEP-WISE ENDORSEMENT: EVALUATING APPROXIMATE KV CACHING IN DIFFUSION LLMS ONE DECISION AT A TIME

Anonymous authors
Paper under double-blind review

ABSTRACT

Most diffusion large language models (dLLMs) use bidirectional attention, so an exact KV cache is impossible: every newly decoded token changes the keys and values (KV) of all other tokens. Approximate KV caching methods such as Fast-dLLM reuse KV from an earlier step and recompute (*refresh*) them only occasionally. Such a method is good to the extent that the cached model approximates cache-free *exact decoding*, a property no metric measures directly. The refresh interval, however, orders these caches in advance: refreshing more often leaves less of the decoding on stale KV. Varying only this interval gives a *refresh-dose* scale of caches whose order is known, which any sound evaluation must follow. Accuracy, the evaluation used most, does not: in two of four configurations of Fast-dLLM, and on HumanEval and MBPP, it calls the cache that never refreshes the better one. Comparing the cached and the exact output token by token barely separates the settings. We propose *step-wise endorsement*: before each step of the cached model, the exact model receives the same state and decides there itself, and four metrics record how often and how far the two disagree on which `[MASK]` to decode and which token to write. All four follow the scale in every configuration on math, at both ends of it on code, and along the refresh knobs of dLLM-Cache, dKV-Cache, and Elastic-Cache; with a tenth of the questions, they order the settings far more reliably than accuracy does with all of them. They also show why Fast-dLLM works: stale KV change decisions, but rarely, and mostly where the exact model itself is uncertain.

1 INTRODUCTION

Diffusion large language models (dLLMs), such as LLaDA (Nie et al., 2025b), Dream (Ye et al., 2025), and the multimodal LLaDA-V (You et al., 2026), generate text by iterative unmasking. The response starts as a sequence of `[MASK]` tokens; at each step the model runs one forward pass with bidirectional attention over the whole sequence and writes the most probable token at the few masked positions with the highest *confidence* (the largest token probability at a position). Because attention is bidirectional, every newly written token changes the keys and values (KV) of all other tokens, so dLLMs cannot cache KV without error. Decoding without any cache, which we call *exact decoding*, recomputes the whole sequence at every step and is expensive.

Many approximate KV caching methods therefore reuse KV computed at an earlier step and recompute them only occasionally (Wu et al., 2026; Liu et al., 2026; Ma et al., 2025; Nguyen-Tri et al., 2026; Jiang et al., 2026); we call such a recomputation a *refresh*. Fast-dLLM (Wu et al., 2026), for example, decodes block by block, refreshes the KV of the whole sequence at the start of each block (FR, full refresh), and within a block recomputes only the current block; we call the schedule that never refreshes after the first forward pass NR (no refresh). These methods are designed to reproduce exact decoding at lower cost, so a caching method is good to the extent that the cached model approximates it. This is an abstract property, not a quantity that can be measured directly, so every evaluation reads something else instead; today that is task accuracy. Approximating exact decoding and maximizing accuracy are related but distinct goals; we evaluate the former.

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.

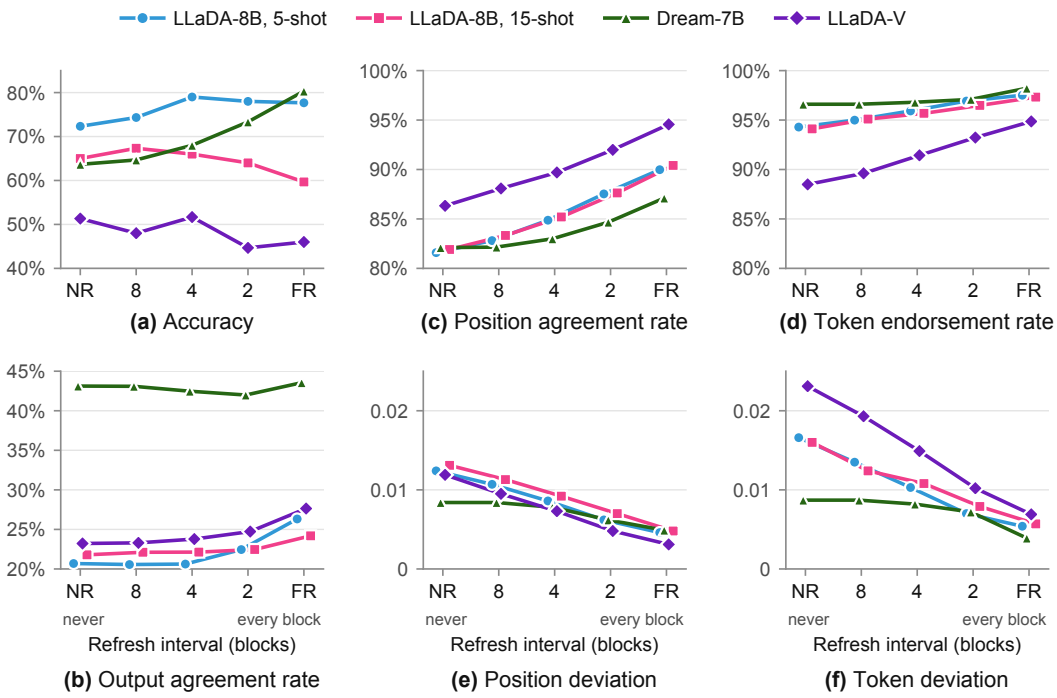


Figure 1: Six readings of the same runs along the refresh-dose scale: Fast-dLLM in four configurations that cover three models (Section 3), with the KV of the whole sequence refreshed at the start of every block (FR), of every second, fourth, or eighth block, or never (NR). More refreshes leave less of the decoding on stale KV, so each setting is a better cache than the one to its left. **(a, b)** Neither evaluation that reads a finished output—task accuracy and the share of positions at which the cached and the exact output carry the same token—follows the scale in every configuration. **(c–f)** The four metrics of step-wise endorsement do, in every configuration; higher rates and lower deviations mean a better cache. Appendix B lists every value.

How can one tell whether an evaluation gets this right? For one operation the answer is known before any measurement: a refresh recomputes the KV from the current state, and a denser schedule refreshes wherever a sparser one does and more, so it leaves less of the decoding on stale KV and is the better cache. This order comes from the refresh operation itself, so it can test any evaluation, including one that takes exact decoding as its reference. Varying only the refresh interval, with everything else fixed, thus gives a graded scale of caches whose order we know, from NR up to FR; we call a reading along it a *refresh-dose* experiment and read five settings in four configurations that cover three models (Figure 1). Following the scale is necessary for a sound evaluation, though not sufficient, so we also ask how many questions an evaluation needs to follow it reliably.

Accuracy does not follow the scale (Figure 1a): from NR to FR it rises in two configurations and falls in the other two, and on HumanEval and MBPP it falls as well. It reads the final output, which depends on much that the cache does not control: with the 15-shot prompt, changing only the rule that scores the output reverses its direction (Section 3). Comparing the cached and the exact output token by token, although it too takes exact decoding as the reference, does no better (Figure 1b): once the two runs write a different token, which happens early, the comparison mostly reads the drift that follows.

We propose to judge a cache where it acts instead: at a single step, on its *decisions*, each of which picks a masked position to decode and a token to write there. Without a cache, a decision depends only on the current *state*: the prompt, the tokens written so far, and the remaining masks. In *step-wise endorsement*, before every step of the cached model, the same model without any cache, the *exact model*, receives the same state and decides there under the same rule, and we record whether it endorses each decision of the cached model; its own decisions are never fed back, so the cached run is unchanged. Four metrics summarize the records: the *position agreement rate* and the *token*

endorsement rate say how often the exact model would make the same choice, and the *position deviation* and the *token deviation* how much probability it loses through the choice of the cached model (Section 4).

All four metrics follow the scale in every configuration, without a single exception, and between NR and FR on HumanEval and MBPP (Figure 1c–f, Section 5). With a tenth of the questions, they order adjacent settings far more reliably than accuracy does with all of them (Figure 4). Because they compare decisions only, they run unchanged along the refresh knobs of dLLM-Cache (Liu et al., 2026), dKV-Cache (Ma et al., 2025), and Elastic-Cache (Nguyen-Tri et al., 2026), and follow all three (Section 6). They also explain why Fast-dLLM works even under NR (Section 7). Our contributions are threefold.

- The refresh-dose experiment, which judges an evaluation of approximate KV caching by whether, and with how many questions, it follows a scale of caches whose order is known in advance. Neither accuracy nor the token-by-token agreement of the outputs does so reliably.
- Step-wise endorsement, a procedure and four metrics that judge a cache one decision at a time. They follow the scale in every configuration of Fast-dLLM, on math and, between NR and FR, on code, and along the knobs of three further caching methods, with a few dozen questions.
- An explanation of why Fast-dLLM works: stale KV change decisions, increasingly with their age but rarely, and mostly where the exact model itself is uncertain.

2 Related Work

Diffusion LLMs and approximate KV caching. Masked diffusion models (Austin et al., 2021; Campbell et al., 2022; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025; Nie et al., 2025a) generate text by iterative unmasking; LLaDA (Nie et al., 2025b) and Dream (Ye et al., 2025) scale them to 7–8B parameters, and LLaDA-V (You et al., 2026) and LaViDa (Li et al., 2025) add visual inputs. Long responses are decoded block by block. Fast-dLLM (Wu et al., 2026) refreshes the KV of all tokens at every block boundary and within a block runs the model on the current block only; its DualCache variant also caches the masked suffix. dLLM-Cache (Liu et al., 2026) also caches attention and feed-forward outputs, dKV-Cache (Ma et al., 2025) caches a token one step after it is decoded, FreeCache (Hu et al., 2026) freezes finished blocks, and Elastic-Cache (Nguyen-Tri et al., 2026), d^{2}Cache (Jiang et al., 2026), and the concurrent Polestar (Lee et al., 2026) select at each step what to recompute. All are validated by accuracy and throughput, and their reuse is justified by the similarity of KV between steps, a weaker argument than it looks (Appendix C).

Evaluating approximate inference. ParallelBench (Kang et al., 2026) shows that parallel decoding hurts tasks with strong dependencies between tokens; the concurrent Zhang et al. (2026) find that the accuracy ranking of dLLM decoding and caching methods changes with the prompt template, and Dou & Shu (2026) compare the text of accelerated and unaccelerated LLaDA-V. All of them read the final output. For autoregressive LLMs, compressed models, including those with a compressed KV cache (Zhang et al., 2023; Xiao et al., 2024), are likewise judged mostly by accuracy and perplexity; Dutta et al. (2024) show that accuracy can hide many changed answers and recommend distances between the two models on the same input, such as the KL divergence. Step-wise endorsement also compares the two models on the same input, but a dLLM decides two things at every step, and the position has no counterpart in an autoregressive model; we also judge the evaluations themselves, against a scale whose order is known in advance.

Judging a decision with the same model. In speculative decoding (Leviathan et al., 2023; Chen et al., 2023), a target model accepts or rejects the tokens of a draft model under the same prefix; in dLLMs, FreeDave (Wu & Zhang, 2026), the concurrent DualDiffusion (Goyal et al., 2026), and Pan et al. (2026) verify drafts of positions and tokens. All of them feed the verdict back into decoding; we never do, so every reading is due to the cache. The concurrent Archer (He et al., 2026) runs an extra forward pass from the same state at sampled steps to choose its refresh signal; we use the comparison to evaluate caches, make it at every step, separate the position from the token, and pair each rate with a deviation.

Table 1: The same runs of LLaDA-8B-Instruct on GSM8K with a 15-shot prompt (300 questions), read along the refresh-dose scale of Figure 1 and scored two ways; all entries in percent. *Continue*: the share of responses that set themselves a further question after answering. *Last answer*: the rule used everywhere else in this paper. *Before the continuation*: the same rule on the response truncated at that further question.

	Exact	NR	8	4	2	FR
Responses that continue	37.3	18.0	16.7	25.0	27.0	28.0
Accuracy, last answer	47.3	65.0	67.3	66.0	64.0	59.7
Accuracy, before the continuation	77.0	69.3	72.7	76.7	79.0	78.0

3 Accuracy and Output Agreement Do Not Follow the Scale

Setup. We read Fast-dLLM in four configurations, 300 questions each: LLaDA-8B-Instruct on GSM8K (Cobbe et al., 2021) with a 5-shot and with a 15-shot prompt, Dream-7B on GSM8K (5-shot), and LLaDA-V on MathVista (Lu et al., 2024), all with the DualCache variant. Responses have 256 tokens in blocks of 16 (LLaDA-V: 96 in blocks of 8). Decoding is deterministic: each step writes the most probable token at the k most confident masked positions of the current block, with $k = 2$ for the LLaDA models and $k = 1$ for Dream-7B. Only the refresh interval varies: every block (FR), every second, fourth, or eighth block, or never (NR). Appendix A gives every detail.

Accuracy. From NR to FR, accuracy rises by 5.3 points for LLaDA-8B-Instruct (5-shot) and by 16.7 for Dream-7B, but falls by 5.3 with the 15-shot prompt and for LLaDA-V (Figure 1a); along the five settings it is monotone only for Dream-7B. The final output answers to more than the cache. With the 15-shot prompt, the model often sets itself a further question after answering, and the standard rule, which reads the number after the last "####", scores that invented answer (Table 1). Exact decoding does this in 37.3% of the questions, and the cached runs come nearer to that habit as the refreshes grow denser, from 18.0% under NR to 28.0% under FR. Scored on the outputs truncated at the invented question, accuracy instead rises from 69.3 to 78.0%: the caches did not change, one line of the scorer did.

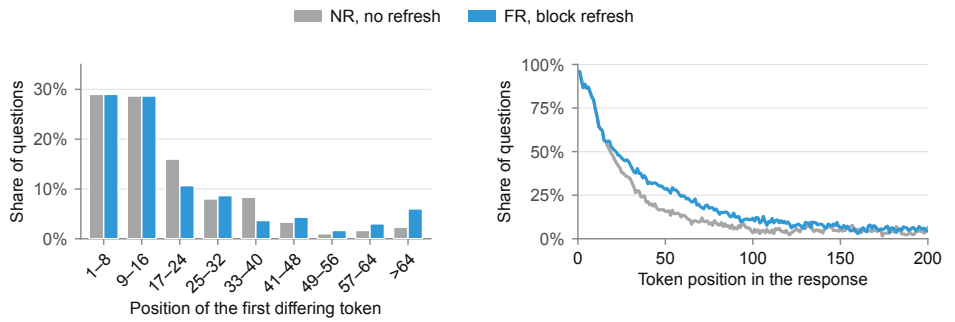
Output agreement. The share of positions at which the cached and the exact output carry the same token is monotone in two configurations (Figure 1b) and reads higher under NR than under three denser settings for Dream-7B. The two runs first differ early (Figure 2a), for LLaDA-8B-Instruct (5-shot) at a median position of 13 out of about 190, and from then on their agreement keeps falling (Figure 2b), so most of what this comparison reads is drift after they separated (Appendix D).

4 Step-Wise Endorsement

To compare a cache with exact decoding decision by decision, the exact model must decide in the same state as the cached model, make the same number of decisions there, and leave the cached run unchanged.

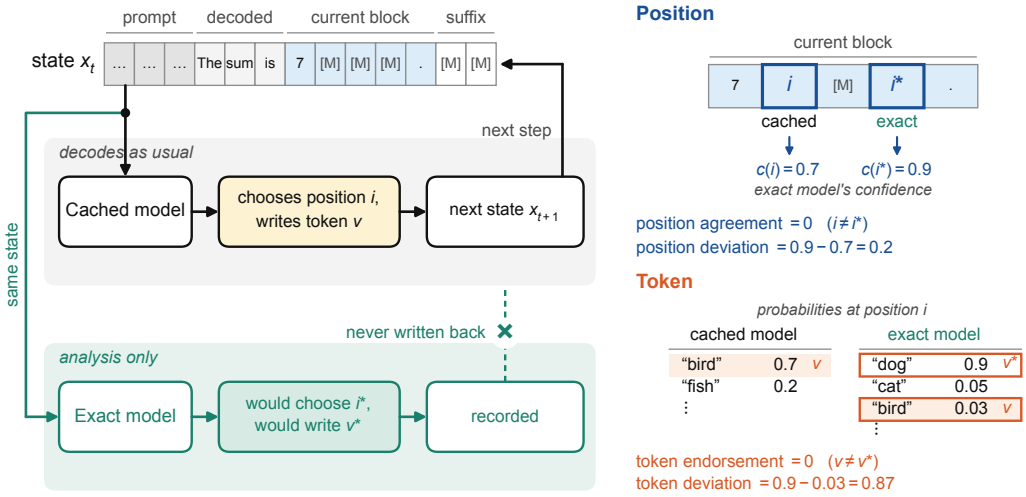
Procedure. Let x_t be the state before step t. Both models compute a token distribution at every masked position of the current block, decode the k positions with the highest confidence, and write the most probable token at each. We fix k instead of using the confidence threshold of Fast-dLLM, so that both models make the same number of decisions in every state. All our readings of Fast-dLLM are of this fixed-k variant. Before each step of the cached model, x_t is also given to the exact model, which runs one forward pass without any cache (Figure 3a). For each decision of the cached model, a position i decoded and a token v written there, we record what the exact model would do and its probabilities at i. Its decisions are never written into the sequence, so the cached output is unchanged and every step is judged in the state the cached model actually reached. The cost is one extra forward pass per step, paid only during analysis.

Four metrics. A decision has two choices: which position to decode, decided by comparing the confidences of all masked positions of the current block, and which token to write, decided by the



(a) Where the two outputs start to differ **(b)** Same token as exact decoding, by position

Figure 2: Token-by-token comparison of final outputs, for LLaDA-8B-Instruct on GSM8K (5-shot, 300 questions), under NR and FR. **(a)** Position of the first token at which the cached output differs from the exact output. Questions whose two outputs never differ (5 for NR, 13 for FR) are not shown. Within the first block, NR and FR are identical by construction, so the first two bins coincide. **(b)** Share of questions whose cached output contains the same token as the exact output, at each position. For most questions the two outputs differ within the first 16 tokens, and from then on the agreement keeps falling.



(a) One decoding step **(b)** The four metrics on one decision

Figure 3: Step-wise endorsement. **(a)** One decoding step. The cached model decodes as usual. The exact model receives the same state x_t and runs one forward pass; its decisions are recorded and never written into the sequence. **(b)** The four metrics on one illustrative decision ($k = 1$), read from the exact model's probabilities in that state: the cached model chose position i and wrote token v, whereas the exact model would have chosen i^* and, at i, written v^*.

distribution at that position. For a masked position j and a token u, let $p(u \mid j)$ be the exact model's probability of u at j and $c(j) = \max_u p(u \mid j)$ its confidence at j. Let $i^* = \arg\max_j c(j)$ be the position the exact model would decode, and $v^* = \arg\max_u p(u \mid i)$ the token it would write at i.[1]

[1]For LLaDA-8B-Instruct and LLaDA-V, $k = 2$: each model decodes two positions per step. We sort the two positions of each model by the exact model's confidence and pair them rank by rank; i^* is the position paired with i, and the position agreement asks whether i is one of the two positions the exact model would decode. The token metrics are unchanged. A position that agrees can then still carry a small position deviation, when the other position of the step does not agree.

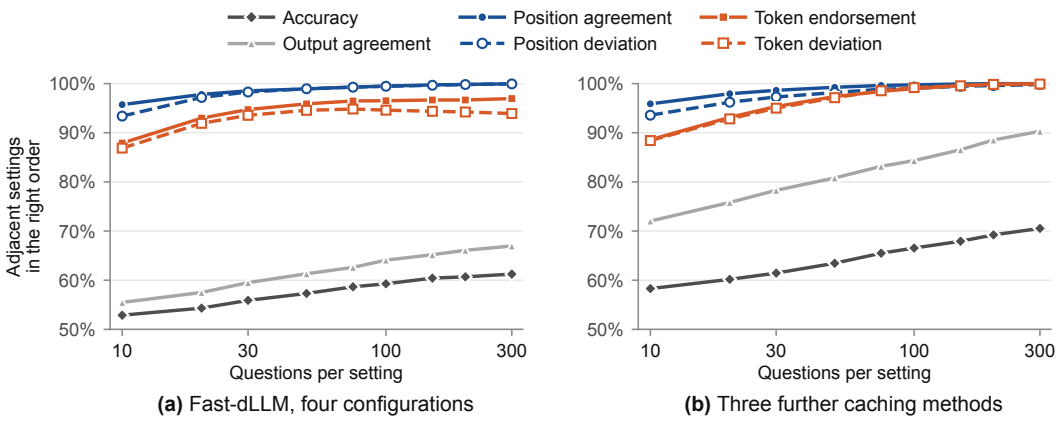


Figure 4: How reliably each evaluation orders the refresh settings, against the number of questions per setting: the share of adjacent settings that come out in the right order, over 2,000 resamples of the questions (the same questions for all settings of a scale; a tie counts as half, so a coin flip orders 50%). **(a)** Fast-dLLM, averaged over the four configurations of Figure 1. **(b)** The scales of the three further caching methods of Section 6, averaged. Appendix F gives each scale.

The four metrics are (Figure 3b):

$$\text{position agreement} = \mathbf{1}\,[\,i = i^*\,], \qquad \text{position deviation} = c(i^*) - c(i), \tag{1}$$

$$\text{token endorsement} = \mathbf{1}\,[\,v = v^*\,], \qquad \text{token deviation} = p(v^* \mid i) - p(v \mid i). \tag{2}$$

The two rates say how often the exact model would make the same choice; the two deviations say how much probability it loses through the choice of the cached model, and are zero when all choices of the step agree. A disagreement is not an error: the metrics say how closely a cache approximates exact decoding, not whether its output is correct. Each metric is averaged over the decisions of a question up to the one that writes the first EOS token, and then over questions.

5 Step-Wise Endorsement Follows the Scale

Every refresh reads as an improvement. On the same runs on which accuracy and output agreement failed, the two rates rise and the two deviations fall with more frequent refreshes in all sixteen curves, four configurations by four metrics (Figure 1c–f): no step toward more refreshes makes a metric worse. From NR to FR, the position agreement rate rises by 5.0 to 8.5 points and both deviations fall by a factor of 1.7 to 3.8 (Appendix B). The only flat step is Dream-7B's first, whose single refresh, at response position 129, comes after almost all counted decisions. The test is not circular: the order of the scale is fixed by the refresh operation before any metric is read, and taking exact decoding as the reference does not by itself pass it, since the output agreement rate also compares with exact decoding and fails (Section 3). Nor do the decisions made at a full refresh, which agree by construction and are more numerous the denser the refresh, carry the result: without them, all sixteen curves stay monotone (Appendix H). The four metrics come from the same forward pass and are correlated, so the sixteen curves are not sixteen independent tests; what separates them from accuracy is how few questions they need (Figure 4). In paired tests over questions (a bootstrap for the metrics, an exact McNemar test for accuracy), Holm-corrected, the four metrics separate every pair of adjacent settings significantly except Dream-7B's flat first pair; accuracy separates one pair of the sixteen.

A few dozen questions suffice. We resample the questions, the same for all settings of a scale (Figure 4). With 30 questions the four metrics order 94 to 99% of adjacent settings correctly; with all 300, accuracy orders 61%, output agreement 67%, and *flips* (Dutta et al., 2024), the questions whose correctness differs from exact decoding, 76% (Appendix F). Asked to order a whole scale, every adjacent pair right, position agreement succeeds in 92% of the resamples with 30 questions, averaged over the four configurations, while accuracy succeeds in 25% with all 300, and never with

Table 2: Fast-dLLM on two code tasks, LLaDA-8B-Instruct, 512-token responses. Rates in percent, deviations in probability.

Task	Setting	pass@1	Output agr. ↑	Pos. agr. ↑	Pos. dev. ↓	Tok. endors. ↑	Tok. dev. ↓
HumanEval	exact	29.9	–	–	–	–	–
(164)	NR	28.0	17.2	80.9	0.0126	94.1	0.0187
	FR	22.0	17.0	89.6	0.0051	96.9	0.0072
MBPP	exact	30.0	–	–	–	–	–
(300)	NR	29.3	48.6	85.2	0.0106	95.1	0.0124
	FR	27.3	53.1	91.0	0.0055	96.8	0.0072

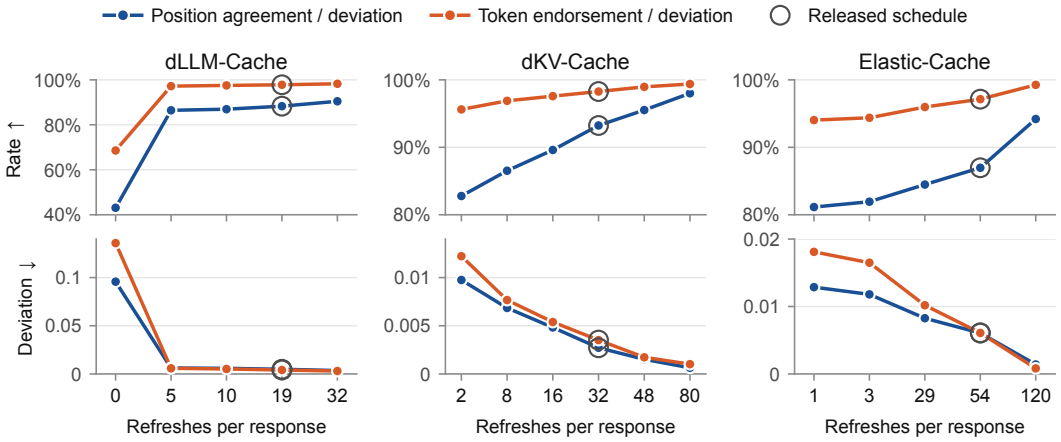


Figure 5: Three more caching methods, each read along a refresh-dose scale of its own, for LLaDA-8B-Instruct on GSM8K (5-shot, 300 questions). **(Top)** the two rates, **(bottom)** the two deviations. The horizontal axis counts a different operation for each method (Appendix B), so it orders the rungs of one scale and says nothing across the three; the rungs are drawn evenly spaced. Along every scale, all four metrics improve from each rung to the next.

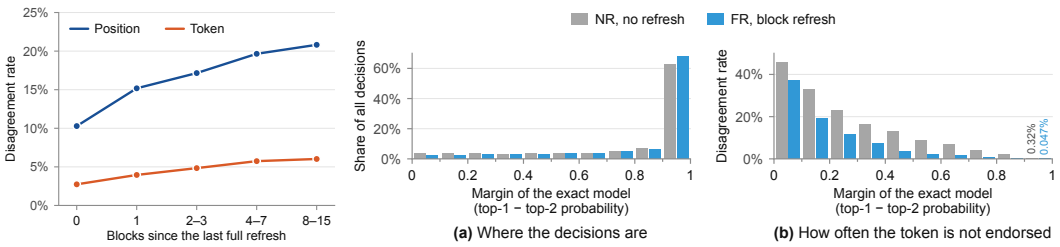
the 15-shot prompt or for LLaDA-V. The three further methods of Section 6 show the same picture (100 against 8%, with 300 questions).

Code generation. On HumanEval (164 problems) and MBPP (300), with 512-token responses from LLaDA-8B-Instruct under NR and FR (Appendix G), pass@1 falls from NR to FR, by 6.0 and by 2.0 points, and on HumanEval output agreement falls too. All four metrics improve on both tasks (Table 2), even as pass@1 falls: they measure how closely a cache tracks exact decoding, not whether its output passes the tests.

The metrics track the final output. A high endorsement rate could in principle coexist with a final output that the exact model would never write, if the few disagreements built on one another. They do not: the exact model finds the outputs under FR more probable than those under NR, and within one setting, the questions on which the cached model deviates most are those whose outputs it finds least probable and on which the cache loses accuracy (Appendix E).

6 Three More Caching Methods

Because the metrics compare decisions only, they do not depend on how a cache produces them. Three further methods cache different quantities and recompute them on triggers of their own, but each has a knob that sets how often, and each knob gives a scale of settings (*rungs*) whose order is known in advance. We read all three for LLaDA-8B-Instruct on GSM8K (5-shot), with the same 300 questions and decoding rule: dLLM-Cache over its two recomputation periods, dKV-Cache over its



(a) Where the decisions are **(b)** How often the token is not endorsed

Figure 6: Where Fast-dLLM's decisions disagree with exact decoding, for LLaDA-8B-Instruct on GSM8K (5-shot). **Left**: position and token disagreement rates against the number of blocks since the last full refresh, the decisions of the five refresh settings pooled. **(a)** Share of all decisions in ten equal-width bins of the exact model's margin, under NR and FR. **(b)** Token disagreement rate within each margin bin.

cache reset and within-block refresh, and Elastic-Cache over its attention-drift threshold γ, each from never refreshing to beyond its released setting (Appendix A).

Along all three, the two rates rise and the two deviations fall from each rung to the next (Figure 5): twelve curves without a single reversal, and all 52 comparisons between adjacent rungs, correlated across the four metrics, are significant in paired bootstrap tests, Holm-corrected. In all three the metrics keep improving above the released schedule, at a cost in computation (Appendix A). The methods recompute very different amounts per step, so each scale says how its own refresh reads, not how the methods rank against one another.

7 Why Does Fast-dLLM Work?

Under NR, fifteen of the sixteen blocks of Fast-dLLM are decoded on KV computed before the block began, yet for LLaDA-8B-Instruct on GSM8K (5-shot) accuracy drops only from 80.0 to 72.3%. The metrics show why.

Most decisions are endorsed. Under NR the exact model still agrees with 82% of the positions and endorses 94% of the tokens (under FR, 90 and 98%). The cached output leaves the exact one early (Section 3), yet its decisions do not drift: after the first token the exact model would not have written, it still endorses 94% of the tokens under NR and 98% under FR.

Disagreement grows with staleness, but slowly. Pooling the five settings and sorting each decision by the number of blocks since the last full refresh, the position disagreement rises from 10.3 to 20.8% and the token disagreement from 2.7 to 6.0% (Figure 6, left). This is not because later blocks are harder: comparing two settings on the same block, the staler one disagrees more often in 110 of 111 pairs, for positions and tokens alike (Appendix H). The KV themselves drift much further over the same span (Appendix C); the decisions follow only a little of that drift.

Disagreements concentrate where the exact model is uncertain. Let the *margin* be the difference between the exact model's two largest token probabilities at a position. About two thirds of all decisions fall in the highest of ten margin bins (Figure 6a), and the token disagreement rate falls from about 40% in the lowest bin to well under 1% in the highest (Figure 6b); decisions with a margin below 0.2 are fewer than one in ten but produce more than half of all token disagreements. Disagreements that override a token the exact model clearly prefers are rare, 0.4 per response under NR and 0.06 under FR, consistent with the modest drop in accuracy (Appendix E).

8 Conclusion

Approximate KV caches for dLLMs are designed to behave as exact decoding would, yet they are evaluated by what comes out at the end. Along the refresh-dose scale, whose order is known in advance, neither accuracy nor a token-by-token comparison of the outputs reliably tells which of two

caches is better, while the four metrics of step-wise endorsement, which judge each decision in the state the cached model reached, follow every scale we read with a few dozen questions and explain why Fast-dLLM works.

Using the metrics. To evaluate a caching method, one can decode a few dozen questions of the target task with step-wise endorsement, which needs no ground truth and costs about 40 seconds per GSM8K question on one RTX 3090 (33 for exact decoding), or 20 minutes for 30 questions. The four metrics then say how closely the cache approximates exact decoding, and read along the method's own refresh knob they show how much each additional recomputation buys; if one number is wanted, the position agreement rate orders the settings most reliably (Appendix F), and accuracy, reported next to it, says how useful the outputs are for the task.

Limitations. Step-wise endorsement costs one extra forward pass per step, so it is a tool for evaluation and analysis, not for deployment. It needs the same number of decisions per step in both models; we fix k, decode deterministically, and have tested neither the confidence threshold of Fast-dLLM nor sampling. Our models have 7–8B parameters and one block length each, our responses at most 512 tokens, on the code tasks we read only the two ends of the scale, and we read each of the three further methods in one configuration. A scale orders the settings of one knob of one method, so it cannot check a comparison across methods, which also differ in what they recompute. Following the scale is necessary but not sufficient for a sound evaluation, and the metrics say how closely a cache approximates exact decoding, not how useful its outputs are.

AI USE STATEMENT

We used generative AI tools (large language models) to survey the literature, to translate and polish the text, and to assist in generating the experimental code. The core method, the design of the experiments, and the structure of the paper were developed by the authors. The authors reviewed all AI-assisted text and code and take responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

All experiments use publicly released model weights and public datasets. Appendix A gives the details needed to rerun them: the models and their numerical precision, the questions and prompts of GSM8K, MathVista, HumanEval, and MBPP, the decoding rule and its parameters for each model, the refresh settings of Fast-dLLM and the refresh settings of each of the three further caching methods, the implementation of step-wise endorsement and the checks made on it, the answer extraction, and the EOS rule under which the metrics are counted. Decoding is deterministic, and rerunning it on another GPU reproduced the archived outputs token by token (Appendix A). Appendix B lists every value plotted in Figures 1 and 5 and the 15-shot configuration under a second scoring rule, Appendix D the token-by-token comparison of the outputs, Appendix E the likelihood estimate, Appendix F the resampling of the questions, Appendix G the settings and values of the code tasks, and Appendix H the staleness analysis. We will release the source code upon publication: the instrumented model files of the three models, the four caching methods with their refresh settings, the recording of step-wise endorsement, and the scripts that compute every table and figure of the paper from the recorded decisions.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 17981–17993, 2021.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 28266–28279, 2022.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

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.

Yaoxuan Dou and Yang Shu. Faster but different: Diagnosing and controlling content drift in accelerated multimodal diffusion language models. *arXiv preprint arXiv:2607.29079*, 2026.

Abhinav Dutta, Sanjeev Krishnan, Nipun Kwatra, and Ramachandran Ramjee. Accuracy is not all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2407.09141.

Leo Gao, Jonathan Tow, Baber Abbasi, et al. The language model evaluation harness. Zenodo, version v0.4.3, doi:10.5281/zenodo.12608602, 2024. URL https://github.com/EleutherAI/lm-evaluation-harness.

Satyam Goyal, Kushal Patel, Tanush Mittal, and Arjun Laxman. DualDiffusion: A speculative decoding strategy for masked diffusion models. *arXiv preprint arXiv:2604.05250*, 2026.

Xuning He, Zinan Sheng, Yongding Tao, Huanyu Liu, Ge Li, Xue Jiang, and Yihong Dong. Archer: Adaptive reuse of cached hidden states for efficient rollback in diffusion language models. *arXiv preprint arXiv:2608.08086*, 2026.

Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S. 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 (ICLR)*, 2026. arXiv:2505.21467.

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, and Xu Yang. d^2Cache: Accelerating diffusion-based LLMs via dual adaptive caching. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.23094.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Il Koo, Nam Ik Cho, and Kangwook Lee. ParallelBench: Understanding the trade-offs of parallel decoding in diffusion LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.04767.

Mingyu Lee, Akshat Ramachandran, Souvik Kundu, and Tushar Krishna. Polestar: Drift-aware cache calibration and token commitment for efficient inference of diffusion LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. arXiv:2607.14107.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2211.17192.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. LaViDa: A large diffusion language model for multimodal understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.16839.

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. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2506.06295.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*, volume 235, pp. 32819–32848, 2024.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.02255.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 149009–149033, 2025. arXiv:2505.15781.

Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. Attention is all you need for KV cache in diffusion LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.14973.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *International Conference on Learning Representations (ICLR)*, 2025a.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. arXiv:2502.09992.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations (ICLR)*, 2025.

Tianxiang Pan, Baitao Gong, Mo Guang, Hongwei Yong, Tianpeng Jiang, Yaqian Li, Zheng Cao, and Kaiwen Long. Trajectory-level speculative decoding for diffusion language models. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2608.27514.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.07524.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.04329.

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 (ICLR)*, 2026. arXiv:2505.22618.

Shutong Wu and Jiawei Zhang. Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. arXiv:2510.00294.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.17453.

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.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. LLaDA-V: Large language diffusion models with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10093–10105, 2026. arXiv:2505.16933.

Hengxiang Zhang, Jiaxi Ren, Renchunzi Xie, and Hongxin Wei. Understanding evaluation illusion in diffusion large language models. *arXiv preprint arXiv:2606.29228*, 2026.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. LMMs-Eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025. arXiv:2407.12772.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.14048.

A Experimental Details

Models. We use the released weights of LLaDA-8B-Instruct (Nie et al., 2025b), Dream-v0-Instruct-7B (Ye et al., 2025), and LLaDA-V (You et al., 2026). LLaDA-8B-Instruct and LLaDA-V run in float16, Dream-7B in bfloat16. All runs use one NVIDIA RTX 3090 GPU per process and PyTorch's scaled-dot-product attention. Decoding is deterministic: the most probable token is always written, there is no sampling and no classifier-free guidance. On 15 questions per text model, rerunning exact decoding on another GPU of the same type reproduced the archived outputs token by token.

Tasks and prompts. GSM8K (Cobbe et al., 2021): the first 300 questions of the test split, the same questions for every model, cache, and refresh setting. The 5-shot prompt consists of one instruction sentence, the first five question–answer pairs of the training split (with the calculator annotations of the original data removed), and the question, ending with "Answer:". The model completes this text; no chat template is applied, for LLaDA-8B-Instruct and for Dream-7B alike. The 15-shot prompt uses the first fifteen pairs of the training split and is otherwise identical. MathVista (Lu et al., 2024): the first 300 questions of the testmini split that have an image, in the order of the dataset. The prompt is built exactly as in the official evaluation code of LLaDA-V (the reason-first format of `lmms-eval`, Zhang et al., 2025) and uses the model's "think" mode.

Decoding. The response has L positions, all `[MASK]` at the start, and is decoded in blocks of length b from left to right. At every step, the model runs one forward pass, and among the masked positions of the current block the k with the highest confidence are decoded with their most probable token. The number of steps is L/k. LLaDA-8B-Instruct: $L = 256$, $b = 16$, $k = 2$, 128 steps. Dream-7B: $L = 256$, $b = 16$, $k = 1$, 256 steps; one token per step is the released setting of this model, and two tokens per step cost it about 27 points of accuracy. LLaDA-V: $L = 96$, $b = 8$, $k = 2$, 48 steps. All three models, and the exact model in step-wise endorsement, run under the same rule; the only difference between the cached and the exact model is where the KV come from.

Fast-dLLM and the refresh settings. We use the DualCache variant of Fast-dLLM (Section 2). At a refresh, one forward pass over the whole sequence writes the KV of every token, including the masked suffix, into the cache. Within a block, every step runs the model on the b tokens of the current block only; their KV are computed fresh and concatenated with the cached ones of all other tokens. The refresh interval r means that the refresh happens at the first step of every r-th block. At the other block boundaries, the first step of the new block runs the model on the finished block and the new block together ($2b$ tokens), writes the KV of the finished block into the cache, and leaves all other entries unchanged. Not counting the first forward pass, the number of refreshes per response is $\lceil L/(br) \rceil - 1$: 15, 7, 3, 1, and 0 for the 16 blocks of LLaDA-8B-Instruct and Dream-7B, and 11, 5, 2, 1, and 0 for the 12 blocks of LLaDA-V. The settings $r = 1$ and $r = L/b$ are FR and NR themselves: their outputs and endorsement records are identical to those of the FR and NR runs, token by token.

The three further caching methods. All three run on LLaDA-8B-Instruct, GSM8K (5-shot), under the decoding rule above ($b = 16$, $k = 2$, $L = 256$, 128 steps), on the same questions and through the same pipeline as Fast-dLLM; only the source of the KV differs.

dLLM-Cache (Liu et al., 2026). The cached prompt features are recomputed every $K_p = 50$ steps and the cached response features every $K_r = 7$ steps, while at every other step the $\rho = 25\%$ of the response tokens whose value vectors changed the most are recomputed and the rest are reused. The rungs of its scale scale the two periods together at a fixed $\rho = 25\%$: $(K_p, K_r) = (200, 28)$, $(100, 14)$, the released $(50, 7)$, and $(25, 4)$. At the lowest rung both intervals are set to infinity and $\rho = 0$, so that only the first forward pass computes the cached features. The released code recomputes the features of the first transformer layer at every step; this is an implementation constant and is kept at every rung.

dKV-Cache (Ma et al., 2025), the dKV-Cache-Decode variant. Only decoded tokens enter the cache, and they enter it one step after they are decoded, so that a token is cached with the KV it has once it is a real token; masked positions are never cached and are recomputed at every step. The cache is reset at every block boundary and, within a block, the whole sequence is refreshed every `cache_reloading_step` $= 8$ steps; under our block length of 16 a block lasts 8 steps, so that this periodic refresh coincides with the block boundary and all refreshes come from the boundaries

(two full forward passes per block: the reset and the one that builds the delayed cache). The rungs of its scale are: no reset at all, so that only the first two forward passes are full; a reset every fourth block; every second; the released every block; and then, above the released setting, a within-block refresh every fourth and every second step on top of the reset at every block.

Elastic-Cache (Nguyen-Tri et al., 2026). The query of a step consists of the most attended token of each layer, the positions decoded at the previous step, and the masked positions of the window; hidden states, queries, keys, and values are cached per layer. The refresh is triggered by attention drift rather than by a period: on a reused layer, the attention distribution of the most attended token is compared with its distribution at the previous step, and if the cosine similarity falls below γ, every layer from the next one to the last is recomputed. The released setting is $\gamma = 0.9$ with one tracked token, and the rungs of its scale are $\gamma = -1$, 0.5, 0.8, 0.9, and 0.99; one tracked token throughout. At $\gamma = -1$ the trigger is unsatisfiable, so after the first step each layer only writes its query rows back into the cache, and at $\gamma = 0.99$ it fires at about 120 of the 128 steps. Two adaptations were needed: the window is taken to be the still-masked positions of the current block, which at $b = 16$ selects the same positions as the sliding window of length 16 of the released code; and, because the trigger needs the attention weights themselves, which scaled-dot-product attention does not return, we keep the released code's explicit attention ($\mathrm{softmax}(QK^\top/\sqrt{d})V$ in the model's dtype).

Porting checks. Each of the ported arms passed the following gates. The exact model's forward pass agrees with a forward pass of the uninstrumented model logit by logit ($\max|\Delta| = 0$); the decoded tokens are identical with and without the extra forward pass; rerunning an arm on the same question reproduces the decoded tokens and every recorded decision; every step makes exactly one reference forward pass; and the attention of a cached arm reads far fewer query rows than exact decoding. Against the released implementations, on three questions each: our exact decoding matches the official cache-free decoding token by token; our dKV-Cache matches the official implementation token by token, with the same first differing position and the same agreement with cache-free decoding; our Elastic-Cache matches it token by token at $\gamma = 0.9$, and on two of the three questions with the trigger switched off, where the third differs from one token at position 37 onward. Explicit attention was checked separately: an arm that recomputes every layer at every step with explicit attention reproduces the archived exact outputs token by token, and in the same state its response logits differ from those of scaled-dot-product attention by at most 0.29, with the same most probable token at 98.4% of the positions. This numerical floor changes no decoded token, so the Elastic-Cache readings can be read as cache effects.

What each method recomputes. Per response, the number of forward passes over the whole sequence is 1.0 (Fast-dLLM with the refresh off), 16.0 (Fast-dLLM, released), 1.0 and 1.0 (dLLM-Cache, off and released), 2.0 and 32.0 (dKV-Cache), and 1.0 and 53.8 (Elastic-Cache, where the number counts the steps at which the layer-wise refresh fired, not full recomputations). The mean number of attention query rows per step and layer, against about 900 for exact decoding, is 25 and 126 (Fast-dLLM), 35 and 132 (dLLM-Cache), 141 and 321 (dKV-Cache), and 21 and 207 (Elastic-Cache). Along the three scales of Section 6 it rises with the refresh, from 35 to 166 (dLLM-Cache), 141 to 609 (dKV-Cache), and 21 to 620 (Elastic-Cache), so the rungs above the released schedule are bought with computation. The methods therefore differ in cost as well as in how good they are under this block structure, which is one more reason not to read the three scales of Section 6 as a ranking.

Implementation of step-wise endorsement. Before every step of the cached model, its current sequence (prompt, written tokens, and `[MASK]` tokens) is passed to a second forward pass with the same weights and no cache. From this pass we record, for each decision of the cached model, the rank of its position among the masked positions of the current block ordered by the exact model's confidence, the confidence at that position and at the position the exact model would decode instead, and the exact model's probabilities of the token written and of its own most probable token. At the first step of every block under FR, where the cache has just been refreshed, the position agreement is 100%, as it must be. For LLaDA-8B-Instruct on GSM8K, one instrumented run takes about 40 seconds per question on one RTX 3090, against about 33 seconds for exact decoding of the same question.

Answer extraction. GSM8K: the number after the last "####" in the response; if there is none, the last number in the response (the flexible extraction of `lm-eval-harness`, Gao et al., 2024); a

prediction is correct if it equals the reference within 10^{-4}. No stop string is applied, so a response that continues after its answer is scored by its last answer; Appendix B reads the 15-shot configuration under a rule that does truncate such a response. MathVista: the rule-based extraction and normalization of the official evaluation code (the text after the last "Answer:", matched to the choices of a multiple-choice question or parsed as a number), without any model-based extraction.

The EOS rule. The EOS tokens are `<|endoftext|>` and `<|eot_id|>` for LLaDA-8B-Instruct and LLaDA-V, and `<|endoftext|>` for Dream-7B. The *first EOS* of a response is the first EOS token that the cached model writes, in the order of decoding. Every reading in this paper is bounded by it: the four metrics and the margin bins count the decisions up to and including the one that writes it, while the token-by-token comparison of the outputs (Appendix D) and the likelihood estimate (Appendix E) run over the response positions up to and including the position it occupies. Because decoding takes a fixed number of steps, the positions after the first EOS are decoded as well, almost always with EOS tokens, and these decisions are not counted; counting all decisions of the response instead changes no comparison in this paper. Per response, the numbers of counted decisions under NR and FR are 181.8 and 189.5 (LLaDA-8B-Instruct, 5-shot), 196.5 and 216.2 (15-shot), 70.8 and 83.2 (Dream-7B), and 90.0 for both settings of LLaDA-V, whose every response writes its first EOS as its ninetieth decision, at the last position of the response; for the three further methods with the refresh off and released they are 136.4 and 188.6 (dLLM-Cache), 185.5 and 186.1 (dKV-Cache), and 183.0 and 186.9 (Elastic-Cache).

B Full Numbers of the Refresh-Dose Scales

Table 3 lists every value read along a refresh-dose scale in this paper. Its first four blocks are the five refresh settings of Fast-dLLM in the four configurations of Section 3, the points of Figure 1, each with the accuracy of exact decoding on the same questions for reference; the last three are the scales of Section 6, one per further caching method, all in the configuration of the first block and thus against the same 80.0% of exact decoding. The column *Refreshes* counts a different operation for each method—full recomputations of the KV for Fast-dLLM and for dKV-Cache, steps at which the cached response features are recomputed for dLLM-Cache, and steps at which the layer-wise refresh fired for Elastic-Cache (the first step included)—so it orders the rungs within a block and is not comparable across blocks. In all seven blocks, each of the four metrics changes monotonically from top to bottom. The output agreement rate does so in five of them, and accuracy in two, the Dream-7B and the Elastic-Cache blocks.

Table 3: Every value read along a refresh-dose scale in this paper: accuracy, the output agreement rate, and the four metrics, 300 questions per row. Within each block the refresh grows denser from top to bottom; *Refreshes* counts a different operation for each method (see the text above), and for Fast-dLLM it leaves out the first forward pass; a dagger marks the released schedule of a method. Rates in percent, deviations in probability.

Refresh setting	Refreshes	Output		Position		Token	
		accuracy	agreement ↑	agreement ↑	deviation ↓	endorsement ↑	deviation ↓
Fast-dLLM, LLaDA-8B-Instruct, GSM8K (5-shot)							
exact decoding	–	80.0	–	–	–	–	–
never (NR)	0	72.3	20.7	81.6	0.0124	94.3	0.0166
every 8 blocks	1	74.3	20.6	82.8	0.0107	95.0	0.0135
every 4 blocks	3	79.0	20.6	84.9	0.0086	95.9	0.0103
every 2 blocks	7	78.0	22.4	87.5	0.0062	96.9	0.0070
every block (FR)	15	77.7	26.3	90.0	0.0046	97.5	0.0054
Fast-dLLM, LLaDA-8B-Instruct, GSM8K (15-shot)							
exact decoding	–	47.3	–	–	–	–	–
never (NR)	0	65.0	21.8	81.9	0.0131	94.1	0.0160
every 8 blocks	1	67.3	22.1	83.3	0.0113	95.1	0.0124
every 4 blocks	3	66.0	22.1	85.2	0.0092	95.7	0.0108
every 2 blocks	7	64.0	22.5	87.6	0.0070	96.5	0.0079
every block (FR)	15	59.7	24.2	90.4	0.0048	97.3	0.0057
Fast-dLLM, Dream-7B, GSM8K (5-shot)							
exact decoding	–	82.7	–	–	–	–	–
never (NR)	0	63.7	43.1	82.1	0.0084	96.6	0.0087
every 8 blocks	1	64.7	43.1	82.2	0.0084	96.6	0.0087
every 4 blocks	3	68.0	42.5	83.0	0.0078	96.8	0.0082
every 2 blocks	7	73.3	42.0	84.7	0.0062	97.1	0.0072
every block (FR)	15	80.3	43.5	87.1	0.0049	98.2	0.0039
Fast-dLLM, LLaDA-V, MathVista							
exact decoding	–	45.3	–	–	–	–	–
never (NR)	0	51.3	23.2	86.3	0.0119	88.5	0.0231
every 8 blocks	1	48.0	23.3	88.1	0.0095	89.6	0.0193
every 4 blocks	2	51.7	23.8	89.7	0.0073	91.4	0.0149
every 2 blocks	5	44.7	24.7	92.0	0.0048	93.2	0.0102
every block (FR)	11	46.0	27.6	94.6	0.0031	94.9	0.0069
dLLM-Cache, recomputation periods $(K_p,\ K_r)$							
off	0	1.7	15.0	43.1	0.0958	68.6	0.1358
$(200,\ 28)$	5	77.7	26.0	86.5	0.0062	97.2	0.0059
$(100,\ 14)$	10	79.0	26.1	87.0	0.0058	97.5	0.0052
$(50,\ 7)^{\dagger}$	19	77.0	26.6	88.3	0.0049	97.8	0.0041
$(25,\ 4)$	32	81.7	28.0	90.5	0.0035	98.3	0.0031
dKV-Cache, cache reset (and, above the released rung, a within-block refresh)							
never	2	75.7	25.6	82.8	0.0098	95.6	0.0122
every 4 blocks	8	75.7	26.5	86.5	0.0069	96.9	0.0077
every 2 blocks	16	77.0	28.6	89.6	0.0048	97.6	0.0054
every block†	32	79.7	35.7	93.2	0.0027	98.3	0.0035
+ every 4 steps	48	77.0	44.6	95.5	0.0015	99.0	0.0017
+ every 2 steps	80	80.3	60.4	98.0	0.0006	99.4	0.0010
Elastic-Cache, attention-drift threshold γ							
$\gamma = -1$	1	70.7	22.2	81.1	0.0129	94.0	0.0181
$\gamma = 0.5$	3	73.3	22.5	81.9	0.0118	94.4	0.0165
$\gamma = 0.8$	29	78.7	23.5	84.5	0.0083	96.0	0.0102
$\gamma = 0.9^{\dagger}$	54	78.7	24.1	87.0	0.0061	97.1	0.0061
$\gamma = 0.99$	120	79.3	40.8	94.2	0.0014	99.3	0.0008

The 15-shot configuration under two scoring rules. Table 1 gives the readings behind the discussion of accuracy in Section 3. A response *continues* when it sets itself a further question after its

own answer; the standard rule of Appendix A reads the number after the last "####" of the response, which on such a response is the answer to the question the model invented. The second accuracy applies the same rule to the response truncated at that further question. Changing the rule moves exact decoding by 89 of the 300 questions, Fast-dLLM under FR by 55, and under NR by 13—the same order as the share of responses that continue. Along the scale, the Spearman correlation of accuracy with the number of refreshes is -0.700 under the first rule and $+0.900$ under the second, and that of the share of responses that continue is $+0.900$. The second rule does not rescue accuracy elsewhere: with the 5-shot prompt fewer than 1% of the responses continue and the two rules agree to within 0.4 points, no response of Dream-7B continues, and MathVista has no such rule. The four metrics are untouched by either rule, because they never read the final output.

C What the Similarity of KV Between Steps Does Not Say

The caching methods of Section 2 are motivated in the same way. The KV of a token, or the features computed from them, change very little from one decoding step to the next, so a step can read them from an earlier one instead of recomputing them. The evidence offered is a similarity between the cached quantities themselves, usually measured between adjacent steps.

It is taken at a distance of one step, while a cache reuses across the whole distance from its last refresh. Under Fast-dLLM with a block length of 16 and two tokens decoded per step, that distance is up to seven steps within a block; with the refresh switched off it is up to 127. A quantity that barely moves between neighbours may still have moved a long way over that span, and a similarity measured at distance one does not say how far. We measured it in the main configuration (LLaDA-8B-Instruct, GSM8K 5-shot, exact decoding, 100 questions): averaged over the 32 layers, the cosine similarity of the values of prompt tokens is 0.995 between adjacent steps but 0.891 between the first forward pass and step 127, the distance over which NR reuses them, and 0.767 in the last ten layers. Under NR the exact model nevertheless still endorses 94% of the tokens (Table 3).

More to the point, a high similarity between KV does not bound the difference between two decisions. A decision compares probabilities across the masked positions of the current block, and those probabilities are computed from the queries of those positions against the concatenation of the cached KV with the KV recomputed at this step. The cache supplies only the first part of that concatenation. As soon as the cached run writes a token that exact decoding would not have written, the current block itself differs, so its queries and its freshly computed keys and values differ too—and they differ in a way that a similarity between the cached parts does not describe. From then on the two runs are ranking different candidates in different states, and how close their cached KV are says nothing about which position wins that ranking or which token is written there.

Step-wise endorsement closes this gap: it does not ask how alike two sets of KV are, but whether, in the state the cached model actually reached, the model without any cache would have made the same two choices.

D Token-by-Token Comparison of the Final Outputs

Section 3 rejected the output agreement rate because most of what it reads is drift accumulated after the two runs first differ. This appendix gives the comparison in full, for LLaDA-8B-Instruct on GSM8K (5-shot, 300 questions). Each cached output is aligned with the output of exact decoding position by position, up to and including the first EOS token of the cached output (Appendix A).

The two outputs first differ at a median position of 13 under NR and 12 under FR, out of 186 and 192 positions compared on average (Figure 2a); only 5 and 13 of the 300 questions never differ. Over all positions the cached output carries the same token as the exact output at 20.7% of them under NR and 26.3% under FR, and over the positions after the first difference at 9.9% and 13.7% (Figure 2b). In the other three configurations the picture is the same: the first difference falls at a median position of 15 (15-shot), 11 (Dream-7B), and 8 (LLaDA-V), and the overall agreement is 21.8, 43.1, and 23.2% under NR against 24.2, 43.5, and 27.6% under FR. A refresh at every block thus removes 7.1% of the disagreement in the main configuration, 3.1% with the 15-shot prompt, 5.8% for LLaDA-V, and 0.8% for Dream-7B. Compared with what the same refreshes do to the four metrics (Figure 1), the reading is both weak and, along the five refresh settings, ordered in only two of the four configurations.

Table 4: Questions of LLaDA-8B-Instruct on GSM8K (5-shot) in quartiles of the per-question token deviation of each cached run (Q1: the 75 questions with the smallest deviation, Q4: the 75 with the largest). For each quartile: the mean token deviation, the mean NLL difference between the cached output and the exact output (nats per token; positive means that the exact model finds the cached output less probable than its own), and the accuracy of the cached run and of exact decoding on the same questions.

Cache	Quartile	Token deviation	NLL difference	Accuracy (cached)	Accuracy (exact)
NR	Q1 (smallest)	0.0047	-0.021	93.3	86.7
	Q2	0.0129	$+0.054$	81.3	82.7
	Q3	0.0189	$+0.048$	69.3	81.3
	Q4 (largest)	0.0299	$+0.210$	45.3	69.3
FR	Q1 (smallest)	0.0007	-0.038	90.7	90.7
	Q2	0.0032	-0.025	82.7	86.7
	Q3	0.0055	$+0.023$	73.3	73.3
	Q4 (largest)	0.0121	$+0.101$	64.0	69.3

E Endorsed Decisions and the Final Output

This appendix checks that endorsed decisions and the final output are related (Section 5), for LLaDA-8B-Instruct on GSM8K (5-shot, 300 questions), the configuration of Section 7. We hand each final output, of exact decoding and of Fast-dLLM under NR and under FR, to the exact model and estimate the negative log-likelihood (NLL) it assigns to that output, in nats per token, with the standard likelihood bound of masked diffusion models: a random number of response positions is masked, the model predicts the tokens there, and the negative log-probabilities of the true tokens are averaged. We use 32 random masks per output, the same masks for the three outputs of a question, and no cache, and we average over the positions up to and including the first EOS token of that output, the scope used everywhere else in the paper. A lower NLL means that the exact model finds the output more probable, that is, closer to what it would write itself.

Across settings. The exact model assigns its own outputs an NLL of 0.390, the outputs under FR 0.405, and the outputs under NR 0.463. The order is the order of the four metrics (Table 3): the cache whose decisions the exact model endorses more often produces outputs that the exact model finds more probable.

Across questions. Within one setting, the questions on which the cached model deviates more are the ones whose outputs the exact model finds less probable. Table 4 splits the questions into quartiles of the per-question token deviation. From the first to the last quartile, the NLL difference between the cached output and the exact output rises from -0.021 to $+0.210$ under NR and from -0.038 to $+0.101$ under FR, and the accuracy of the cached model falls below that of exact decoding on the same questions, by 24 points in the last quartile under NR. The two position metrics are related more weakly, which is expected: a different position changes the order in which tokens are written, not necessarily the tokens themselves, and the NLL reads only the final tokens.

The two readings are therefore related, but they are not the same. Step-wise endorsement reads each step in the state the cached model reached; the final output also depends on factors outside the cache's control, such as where the response stops, which decides which number is scored (Appendix A).

F How Many Questions Each Evaluation Needs

Figure 4 averages over scales; Table 5 gives each. For a number n of questions, we draw n questions with replacement from the 300, the same questions for every setting of a scale, compute each reading on every setting, and count the share of adjacent settings that come out in the right order: higher with more refreshes for the rates, accuracy, and the output agreement rate, lower for the deviations, with a tie counting as half; we also record whether every adjacent pair of the scale is in the right order. We repeat this 2,000 times with a fixed random seed; with all 300 questions drawn once each, every reading equals the value in Table 3. Accuracy orders the settings of Dream-7B well, where it rises

by 16.7 points from NR to FR, but with the 15-shot prompt and for LLaDA-V it orders fewer pairs correctly the more questions it is given, since over all 300 questions its difference already points the wrong way. The only readings of the four metrics that stay below 90% are the token metrics of Dream-7B, whose first step, from NR to a refresh every eight blocks, is flat (Section 5); resampling splits that pair evenly. Flips, the share of questions whose correctness differs from exact decoding (Dutta et al., 2024), read 23.0, 21.0, 19.7, 17.3, and 17.0% from NR to FR for LLaDA-8B-Instruct (5-shot), 42.3, 40.0, 39.3, 34.0, and 35.7% with the 15-shot prompt, 29.0, 28.0, 24.7, 19.3, and 11.7% for Dream-7B, and 29.3, 30.7, 29.0, 26.0, and 26.0% for LLaDA-V; they order 62, 69, and 76% of the adjacent settings of Fast-dLLM with 30, 100, and 300 questions, and are monotone along none of the three further scales. Read as whole scales, with all 300 questions, position agreement orders every adjacent pair right in 99 to 100% of the resamples on all seven scales; accuracy does so in 7, 0, 94, and 0% for the four configurations of Fast-dLLM and in 8, 1, and 16% for dLLM-Cache, dKV-Cache, and Elastic-Cache.

Table 5: Share of adjacent settings in the right order (%), with 30 / 100 / 300 questions per setting, over 2,000 resamples of the questions. The first four rows are the configurations of Fast-dLLM in Figure 1, the last three the scales of Section 6.

Scale	Accuracy	Output agr.	Position agr.	Position dev.	Token endors.	Token dev.
LLaDA-8B, 5-shot	58 / 63 / 66	66 / 70 / 70	100 / 100 / 100	100 / 100 / 100	99 / 100 / 100	99 / 100 / 100
LLaDA-8B, 15-shot	46 / 41 / 38	63 / 70 / 78	100 / 100 / 100	100 / 100 / 100	97 / 100 / 100	97 / 100 / 100
Dream-7B	76 / 90 / 99	43 / 39 / 35	94 / 98 / 100	94 / 98 / 100	83 / 86 / 88	79 / 79 / 76
LLaDA-V	45 / 42 / 43	66 / 76 / 85	100 / 100 / 100	100 / 100 / 100	99 / 100 / 100	99 / 100 / 100
dLLM-Cache	68 / 72 / 74	71 / 77 / 83	97 / 99 / 100	93 / 97 / 99	92 / 98 / 100	91 / 98 / 100
dKV-Cache	55 / 59 / 64	92 / 98 / 100	100 / 100 / 100	100 / 100 / 100	99 / 100 / 100	97 / 100 / 100
Elastic-Cache	61 / 68 / 74	72 / 78 / 88	99 / 100 / 100	99 / 100 / 100	95 / 99 / 100	97 / 100 / 100

G Code Generation

We read Fast-dLLM on two code tasks for LLaDA-8B-Instruct, under NR and FR: HumanEval (all 164 problems, 0-shot: the model completes the function body after the signature and docstring, without a chat template) and MBPP (the first 300 problems of the test split, with the 3-shot prompt of `lm-eval-harness`). Responses have 512 tokens, decoded in 32 blocks of 16 with $k = 2$ (256 steps); everything else is as in Appendix A. A solution is scored by executing its tests (pass@1), and the four metrics count the decisions up to the first EOS. Table 2 gives every value. The responses to MBPP are short, 63 counted decisions on average; those to HumanEval are long, 282 under NR and 345 under FR.

H Disagreement and the Staleness of the KV

Section 7 relates disagreement to how long ago the KV were last refreshed. We pool the decisions of the five refresh settings of Fast-dLLM for LLaDA-8B-Instruct on GSM8K (5-shot, 300 questions), up to the first EOS, and sort each by the number of blocks since the last full refresh of the whole sequence: zero for the block in which that refresh took place, and up to fifteen under NR, whose only full forward pass is the first. At the very step of a full refresh the cached and the exact model see identical KV, and all 14,699 such decisions agree in position and in token. Because such decisions agree by construction and are more numerous the denser the refresh (under FR, 7 to 13% of the counted decisions of the text models and 27% of LLaDA-V), we recomputed the four metrics along the four scales of Figure 1 without them: all sixteen curves remain monotone, and only Dream-7B's first step stays flat, in the two token metrics. Both disagreement rates rise monotonically with the number of blocks (Figure 6, left).

Pooling mixes settings, and later blocks may be harder, so we also compare two settings on the same block of the same questions, where only the staleness of the KV differs: of the 111 such pairs (blocks 1 to 15), the staler setting disagrees more often in 110, for the position and for the token alike. We count staleness in blocks, not steps, because within a block the disagreement also depends on the step: the last step must decode the two positions left, and the hardest tokens tend to be left to the end.