RELIABLE PARALLEL DECODING IN MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models (MDLMs) can generate text efficiently by predicting multiple masked tokens in parallel, but predictions from the same forward pass are not necessarily reliable when committed together. We study when parallel commitment is reliable. Our diagnostics show that confidence alone does not determine a reliable commitment order: confident predictions near the end of the sequence can fix an answer before its supporting computations are established, and downstream predictions become less reliable as the uncertainty of their upstream context grows. At the same time, a single forward pass can already resolve several masked tokens, and predictions that remain stable across the final layers are more likely to be correct. Based on these findings, we propose Reliable Parallel Decoding (RPD), a training-free method that selects candidates by layerwise prediction stability and final confidence, and commits them under a cumulative entropy budget over their preceding masked positions. RPD defers predictions with uncertain upstream context while committing the remaining candidates in parallel, without relying on a fixed block schedule. Across mathematical reasoning and code generation benchmarks on LLaDA and Dream, RPD achieves the highest decoding throughput among the evaluated methods while maintaining or improving accuracy. Code is available at `https://anonymous.4open.science/r/RPD-B1EB`.

1 INTRODUCTION

Masked diffusion language models (MDLMs) offer the potential for efficient generation by predicting multiple masked tokens in parallel and progressively filling a sequence over successive decoding iterations (Nie et al., 2026; Gong et al., 2022). At each iteration, the decoder must choose which predictions to accept and use as context for the next iteration. Committing more tokens can reduce the number of decoding iterations, but predictions produced in the same forward pass are not necessarily reliable when committed together. These considerations motivate our central research question: ***When can masked diffusion language models safely commit multiple tokens in parallel during decoding?***

Figure 1 Left (b) shows a U-shaped confidence profile across the generation canvas, with high confidence near both ends of the sequence. High confidence at the end reflects strong formatting priors rather than a correctly computed answer. As a result, confidence-based decoding can lock in an incorrect answer before its supporting computations are established (Figure 1 Left). Imposing ordering constraints mitigates this problem (Figure 1 Left (a)): delaying the answer region until all other tokens are committed improves accuracy from 57.0% to 62.3%, and further requiring operands to precede their dependent results raises it to 72.0%. These results suggest that confidence alone does not determine a reliable commitment order. A reliable order should instead reflect dependencies: a prediction should wait while the information it depends on remains unresolved, and can otherwise be committed in parallel. Existing methods approximate this in different ways.

Block-wise decoding, as used in LLaDA (Nie et al., 2026), imposes order at a fixed granularity, decoding in parallel within each block while enforcing order across blocks, but its boundaries do not adapt to which dependencies remain unresolved. Other methods select tokens adaptively but judge each prediction without regard to its context. Fast-dLLM (Wu et al., 2026) and EB-Sampler (Ben-Hamu et al., 2026) use confidence thresholds or entropy budgets over the selected

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.

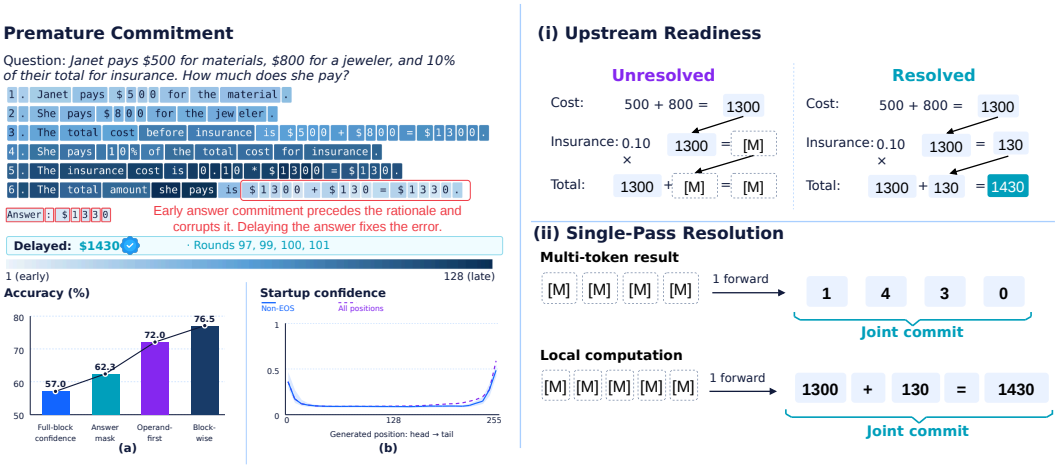


Figure 1: Left: Confidence-based decoding commits the answer early and fixes an incorrect value ($1330), whereas delaying the answer yields the correct $1430. (a) Accuracy improves under stricter ordering constraints. (b) Initial confidence peaks near the sequence boundaries. Right: (i) Downstream tokens should wait until their upstream values are resolved. (ii) Tokens resolved within one forward pass can be committed together.

tokens, without assessing whether the upstream information supporting each prediction has been resolved. DAPD (Kim et al., 2026) uses attention to avoid committing strongly coupled tokens together, but separating dependent predictions does not determine which should come first. LoPA (Xu et al., 2025) explores alternative filling orders to improve subsequent parallelism, at the cost of evaluating additional branches. What remains missing is a way to use the current forward pass to decide which predictions should wait and which can be committed together.

We address this question with two complementary insights. First, downstream predictions should wait while their upstream information remains uncertain. In Figure 1 (i), committing the total while the insurance amount is still masked risks locking in an unsupported answer. Second, a single forward pass can already resolve several masked tokens at once. In Figure 1 (ii), one forward pass produces all digits of 1430 and even the local computation $1300+130=1430$. Section 2 shows that predictions remaining consistent across the final layers without losing probability are more likely to be correct, which allows such tokens to be identified and committed together.

We turn these insights into Reliable Parallel Decoding (RPD), a method that uses upstream uncertainty and prediction stability to decide which tokens to commit at each decoding step. RPD delays predictions whose upstream information remains too uncertain and uses layerwise consistency together with final confidence to identify candidates for joint commitment. Across four mathematical reasoning and code generation benchmarks on LLaDA and Dream, RPD achieves the highest decoding throughput in all eight model–task settings, running 2.4–6.1$\times$ faster than default decoding, while maintaining or improving accuracy.

Our main contributions are:

- We analyze parallel commitment in MDLMs and identify two properties that determine its reliability: downstream predictions are less reliable when their upstream context remains uncertain, and predictions that remain stable across layers are more likely to be correct.
- We propose Reliable Parallel Decoding (RPD), a training-free method that selects candidates by layerwise stability and commits them under a cumulative entropy budget, ordering commitment by upstream uncertainty rather than a fixed block schedule.
- We evaluate RPD on mathematical reasoning and code generation with LLaDA and Dream. RPD achieves the highest decoding throughput among the evaluated methods while maintaining or improving accuracy, and ablations show that layerwise stability and cumulative entropy contribute parallelism and reliability, respectively.

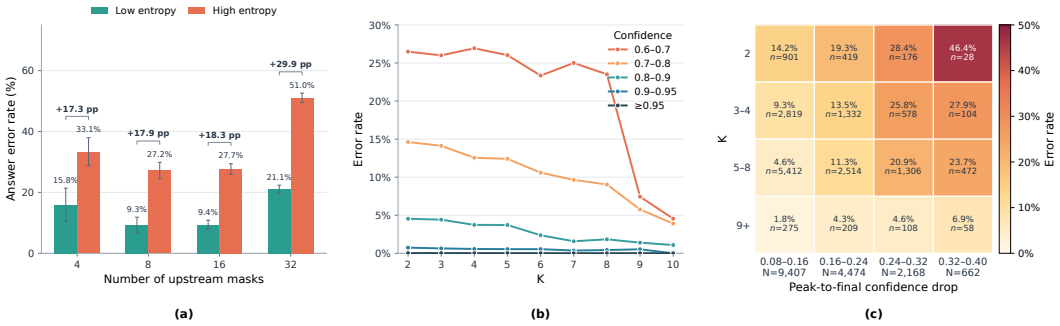


Figure 2: Diagnostics on Dream. (a) Answer error rates for low- and high-upstream-entropy masking patterns within the same problem and mask count. (b) Token error rates under increasing *prediction persistence* K. (c) Token error rates by *prediction persistence* K and *confidence drop* r_i for predictions with final confidence in $[0.6, 0.9)$.

2 Understanding Parallel Commitment

The two insights above raise two questions: how unresolved upstream uncertainty relates to downstream errors, and how to identify predictions that can be reliably committed together. We investigate these questions through controlled masking experiments and analysis of layerwise predictions.

We use 2,000 gold arithmetic traces from GSM8K-Aug (Deng et al., 2024), masking both selected upstream digits and all digits of the final answer. For each problem, we sample 16 masking patterns at each upstream mask count $m \in \{4, 8, 16, 32\}$ and evaluate each state with one forward pass. Data construction and complete results, including LLaDA, are given in A.2 and A.3

2.1 Upstream Uncertainty and Downstream Errors

Figure 1 Left (b) shows how high-confidence predictions near the end of the sequence can be committed before their supporting calculations are complete. LLaDA's block-wise decoding prevents commitment in later blocks, while Dream's default configuration, with temperature 0.1, top-p 0.9, and lowest-entropy-first selection, produces an approximately left-to-right order in our observations (Appendix A.4). Both give earlier computations time to be established before downstream predictions are committed. This raises a question: *are downstream errors associated with uncertainty in the upstream information that remains unresolved?*

We define *cumulative entropy* as

$$E = \sum_{j \in \mathcal{U}} H(p_j), \qquad H(p_j) = -\sum_{v \in \mathcal{V}} p_j(v) \log p_j(v). \tag{1}$$

where $\mathcal{U}$ contains the masked upstream digit positions, p_j is the predictive distribution at position j, and $\mathcal{V}$ is the vocabulary. Within each problem and mask count, we compare the five lowest-entropy masking patterns with the five highest-entropy patterns, retaining problems with both correct and incorrect answers. Figure 2a shows that high-entropy patterns have answer error rates 17.3–29.9 percentage points higher, with all paired 95% confidence intervals above zero. These results support our first insight: downstream predictions are less reliable when their upstream information remains uncertain.

2.2 Layerwise Stability and Prediction Reliability

Respecting upstream dependencies does not mean that every token must wait for a separate decoding iteration. Such a restriction would sacrifice the parallelism that distinguishes MDLMs from autoregressive models. Our second insight is that a single forward pass can already resolve several masked tokens. We therefore examine whether the evolution of predictions across layers helps identify these opportunities.

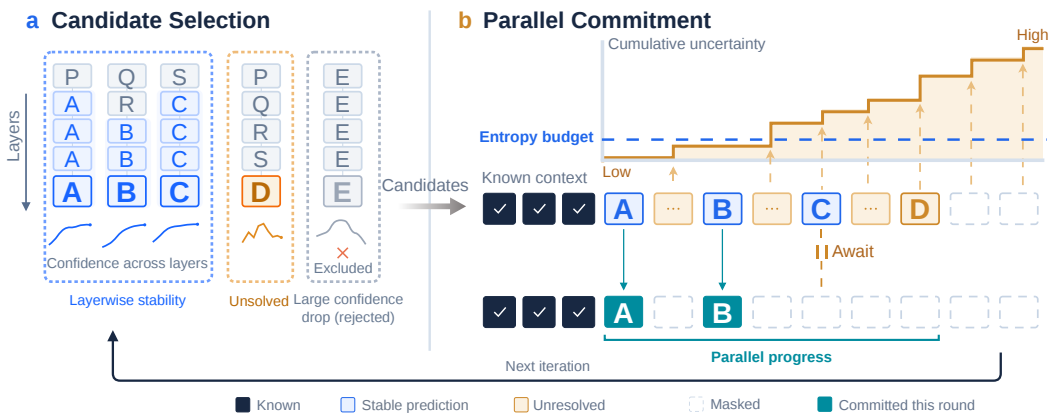


Figure 3: Overview of RPD. (a) Layerwise stability and confidence identify candidate predictions. Prediction persistence favors consistent predictions, while confidence drop penalizes predictions that lose probability. (b) A cumulative entropy budget determines which candidates can be committed together. A and B are committed in parallel, while C waits because the cumulative entropy of its preceding masked positions exceeds the budget.

Using the same masked states, we project intermediate hidden states into the vocabulary space and analyze predictions at the target-answer positions over the final half of the network. Let y_i be the final-layer prediction. We define *prediction persistence* as

$$K_i = \max\left\{k \in \{1, \ldots, L - \ell_0 + 1\} : \arg\max_v p_i^{(\ell)}(v) = y_i \ \text{ for all } \ell = L - k + 1, \ldots, L\right\}. \tag{2}$$

where $p_i^{(\ell)}$ is the predictive distribution at layer ℓ, L is the final layer, and ℓ_0 is the first layer in the analyzed half. Thus, K_i counts consecutive layers ending at the final layer that predict the same token. Figure 2b plots token error rates under increasing persistence requirements $K_i \geq k$, grouped by final confidence. Predictions that persist across more layers generally have lower error rates, even within the same confidence range.

Beyond how long a prediction persists, we also examine how its probability changes during this period. Even when the predicted token stays the same, its probability can fall before the final layer. We quantify this change as *confidence drop*,

$$r_i = \max_{\ell \in [s_i, L]} p_i^{(\ell)}(y_i) - p_i^{(L)}(y_i). \tag{3}$$

where $p_i^{(\ell)}$ is the predictive distribution at layer ℓ, s_i starts the final consistent suffix, and L is the final layer. Figure 2c shows that larger confidence drops correspond to more errors, while longer persistence corresponds to fewer errors. For $K_i = 5$–8, the error rate rises from 4.6% to 23.7% as confidence drop increases. These findings guide the design of our Reliable Parallel Decoding (RPD) method in Section 3.

3 Reliable Parallel Decoding

Section 2 identifies two conditions for committing a prediction reliably: the prediction should be stable across layers (Section 2.2), and the upstream information it depends on should be sufficiently resolved (Section 2.1). Reliable Parallel Decoding (RPD) turns these conditions into two stages that operate on a single forward pass (Figure 3). Candidate selection (Section 3.1) keeps predictions that are individually reliable, and parallel commitment (Section 3.2) determines which candidates can be committed together.

3.1 Candidate Selection with Layerwise Stability

Final confidence alone does not determine whether a prediction is reliable. Within the same confidence range, token error rates decrease as prediction persistence grows (Figure 2b) and increase

with confidence drop (Figure 2c). We therefore combine the two layerwise quantities into a stability score:

$$S_i = \min(K_i, K_{\max}) - w\, r_i, \tag{4}$$

where $K_{\max}$ caps the contribution of prediction persistence and $w > 0$ weights the penalty for confidence drop. Under a fixed threshold on S_i, longer persistence tolerates a larger drop, up to the cap. This follows Figure 2c: as confidence drop increases, the error rate rises from 14.2% to 46.4% for $K_i = 2$, but only from 1.8% to 6.9% for $K_i \geq 9$.

The stability test matters most at intermediate confidence. Predictions with very high confidence are already reliable (Figure 2b), whereas a low-confidence prediction can keep the same token across layers without being correct. Let $\mathcal{M}_t$ denote the masked positions at iteration t and $c_i = p_i^{(L)}(y_i)$ the final confidence. We define the candidate set as

$$\mathcal{C}_t = \left\{\, i \in \mathcal{M}_t : c_i \geq \theta_h \text{ or } \left(\theta_c \leq c_i < \theta_h \text{ and } S_i \geq \theta_s\right) \right\}, \tag{5}$$

where θ_h is the high-confidence threshold, $\theta_c < \theta_h$ is the minimum confidence for stability-based admission, and θ_s is the stability threshold. Hyperparameter values are given in Appendix A.6. High-confidence predictions bypass only the stability test and remain subject to the cumulative entropy budget in Section 3.2. Figure 3a illustrates this selection. A, B, and C remain consistent across the final layers and become candidates. D keeps changing, and E is excluded by a large confidence drop despite its persistent prediction.

3.2 Parallel Commitment with Cumulative Entropy

Candidates in $\mathcal{C}_t$ have stable predictions, but stability alone does not ensure that a prediction is supported by resolved context. As shown in Figure 1 Left (b), a prediction near the end of the sequence can be confident and stable while the computations it depends on are still masked, and Section 2.1 shows that such downstream predictions become less reliable as upstream uncertainty grows. We therefore commit a candidate only when the uncertainty in its preceding masked positions is sufficiently low.

Whether this condition holds depends not only on the current masks, but also on which earlier candidates are committed alongside it in the same iteration. We therefore construct the commitment set sequentially. Starting from an empty set, we scan the candidates in $\mathcal{C}_t$ from left to right. When the scan reaches candidate i, let $\mathcal{A}_{t,<i}$ denote the candidates already selected. We measure the upstream uncertainty of i as the cumulative entropy of the preceding positions that will remain masked after this iteration:

$$E_i = \sum_{\substack{j<i \\ j \in \mathcal{M}_t \setminus \mathcal{A}_{t,<i}}} H\left(p_j^{(L)}\right), \tag{6}$$

which mirrors Eq. 1 using the final-layer distributions of the current forward pass. The sum covers every preceding masked position, including non-candidates and rejected candidates, but excludes $\mathcal{A}_{t,<i}$: positions committed together with i serve as its context, since related tokens can be resolved within a single forward pass (Section 2.2). Candidate i is selected if $E_i \leq \beta$ for a fixed entropy budget β, giving

$$\mathcal{A}_t = \{\, i \in \mathcal{C}_t : E_i \leq \beta \,\}. \tag{7}$$

If $\mathcal{A}_t$ is empty, we commit the single prediction with the highest final confidence among the first W masked positions from the left, which guarantees progress at every iteration. All positions in $\mathcal{A}_t$ are then filled with their predictions y_i from the same forward pass, and the remaining positions stay masked for the next iteration. In Figure 3b, A and B are committed together because little entropy separates them, while C waits despite its stable prediction because the entropy accumulated before it exceeds β.

Variants. Full RPD decodes over the full canvas without blocks, relying on the cumulative entropy budget to determine commitment order. We also consider RPD-block, which applies the candidate selection of Section 3.1 within fixed 32-token blocks and omits the cumulative entropy budget, relying on the block order alone. It isolates the contribution of layerwise stability and offers a faster operating point, while full RPD further delays candidates with uncertain upstream context.

Table 1: Main results across mathematical reasoning and code generation benchmarks. **Bold** indicates the best result among all methods, while underline indicates the better result between our two variants when neither is best overall.

Model	Benchmark	Metric	Default	Fast-dLLM	EB-Sampler	LoPA	DAPD	RPD-block	RPD
Dream-7B Instruct	HumanEval	pass@1 ↑	57.32	60.37	56.71	55.49	55.49	59.76	**60.98**
		NFE ↓	256.00	64.54	78.35	**35.79**	77.74	61.72	63.09
		TPS ↑	7.35	31.69	25.57	19.71	20.93	**31.71**	30.07
	MBPP	pass@1 ↑	56.80	58.20	56.80	51.20	56.40	58.00	**58.40**
		NFE ↓	256.00	57.63	62.41	**36.95**	65.05	55.49	55.89
		TPS ↑	4.09	16.82	15.30	11.78	13.82	**17.34**	15.14
	GSM8K	Acc. ↑	81.35	81.73	81.35	81.65	80.59	81.27	**81.80**
		NFE ↓	256.00	76.04	89.26	**37.20**	79.01	70.78	82.03
		TPS ↑	8.98	30.03	25.53	20.90	22.02	**30.82**	24.56
	MATH-500	Acc. ↑	41.80	**46.00**	45.00	44.00	41.80	44.20	45.80
		NFE ↓	256.00	114.34	124.60	**55.28**	112.64	108.93	116.17
		TPS ↑	12.57	29.74	27.09	20.43	24.07	**30.51**	27.71
LLaDA-8B Instruct	HumanEval	pass@1 ↑	43.29	43.29	43.90	40.24	41.46	41.46	**43.90**
		NFE ↓	256.00	44.48	59.06	**24.68**	49.09	41.23	40.59
		TPS ↑	7.70	43.13	32.10	29.32	36.46	**46.23**	45.29
	MBPP	pass@1 ↑	39.60	40.20	39.20	39.40	37.80	38.60	**40.60**
		NFE ↓	256.00	44.73	53.26	**23.94**	47.16	40.61	38.16
		TPS ↑	5.93	33.13	26.96	21.58	28.62	33.91	**35.92**
	GSM8K	Acc. ↑	76.50	76.35	**79.15**	76.50	76.19	75.74	78.85
		NFE ↓	256.00	77.09	88.79	**38.35**	75.77	69.08	71.76
		TPS ↑	10.50	35.63	30.72	25.83	34.34	**38.55**	35.30
	MATH-500	Acc. ↑	38.00	37.40	**39.00**	38.60	37.20	37.80	38.80
		NFE ↓	256.00	103.78	116.01	**50.39**	98.71	93.99	98.88
		TPS ↑	11.31	27.59	24.46	19.48	28.18	**30.00**	26.77

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate mathematical reasoning on the full test sets of GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), and code generation on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). We use fixed, task-specific zero-shot prompts, the models' native chat templates, and a maximum generation length of 256 tokens for every dataset. The same prompts are shared by all decoding methods. Their exact wording is given in Appendix A.1.

Models and Baselines. We evaluate LLaDA-8B-Instruct (Nie et al., 2026) and Dream-7B-Instruct (Ye et al., 2025). We compare with each model's default sampler and with Fast-dLLM (Wu et al., 2026), EB-Sampler (Ben-Hamu et al., 2026), LoPA (Xu et al., 2025), and DAPD (Kim et al., 2026). The default samplers are LLaDA's block-wise greedy decoder and Dream's full-canvas entropy sampler (temperature 0.1, top-$p = 0.9$). All accelerated baselines and RPD-block decode within 32-token blocks, whereas full RPD decodes over the full canvas. All methods share the same checkpoints, prompts, generation budget, and evaluators; method-specific settings are given in Appendix A.5.

Metrics. We report accuracy on GSM8K and MATH-500 and pass@1 on HumanEval and MBPP. For efficiency, we report the number of function evaluations (NFE) and throughput in tokens per second (TPS). NFE is the average number of backbone forward passes per example, where each lookahead or branch evaluation counts as a separate pass. TPS is the number of generated tokens divided by the wall-clock time of the decoding loop, measured with batch size 1 on a single NVIDIA A6000 with GPU synchronization before each timing. NFE reflects model evaluation cost, while TPS reflects practical decoding speed, including the overhead of each method's selection rule.

4.2 MAIN RESULTS

Table 1 compares RPD with existing decoding methods across two models and four benchmarks. RPD variants achieve the highest throughput in every setting, with 2.4–6.1× speedups over the default samplers.

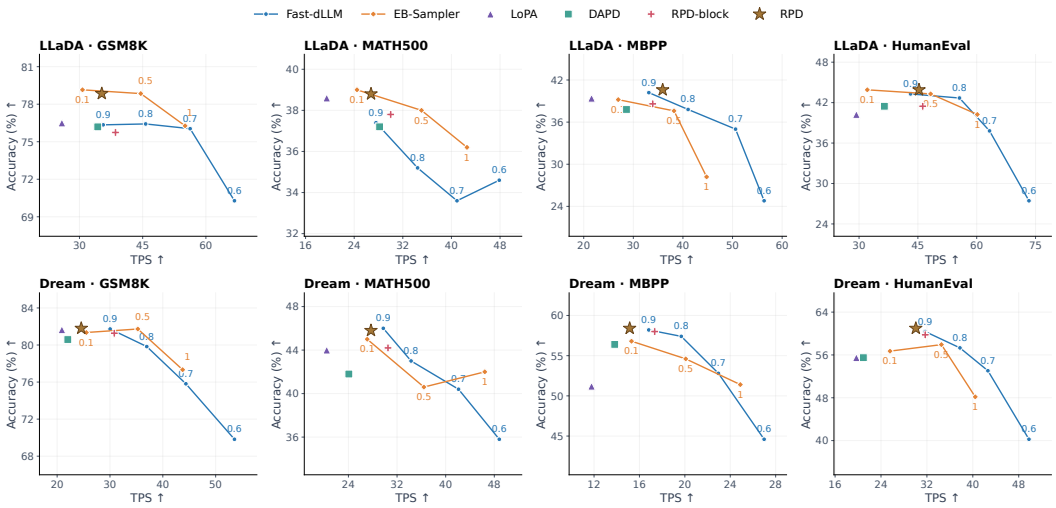


Figure 4: Accuracy–throughput trade-offs on LLaDA (top) and Dream (bottom). Fast-dLLM and EB-Sampler are swept over their confidence threshold and entropy budget, respectively, with each point labeled by its value. Other methods are shown at their default configurations.

Accuracy. These speedups come without a loss in accuracy. RPD ranks first or second in accuracy in all eight settings. Relative to Fast-dLLM, which commits tokens by confidence alone, RPD improves accuracy in seven settings and remains within 0.2 points in the remaining one. Only Fast-dLLM on Dream MATH-500 and EB-Sampler on the LLaDA math benchmarks exceed RPD, the latter at noticeably lower throughput. Combining layerwise stability with upstream uncertainty thus identifies reliable tokens more effectively than a single confidence or entropy criterion.

RPD versus RPD-block. The two variants offer complementary trade-offs. Full RPD removes the block structure and relies on the cumulative entropy budget to order commitment. It improves accuracy over RPD-block in all eight settings at a modest cost in throughput, indicating that adaptive ordering based on upstream uncertainty is more reliable than a fixed block schedule. This supports the finding in Section 2.1: deferring predictions with uncertain upstream context improves reliability, while layerwise stability alone already enables aggressive parallel commitment.

NFE and TPS. LoPA requires the fewest forward evaluations, but it has the lowest TPS among the accelerated methods, since its lookahead branches increase the cost of each evaluation. NFE therefore does not fully reflect decoding cost, and we treat TPS as the primary efficiency metric. RPD adds only layerwise projections to each forward pass, and its overhead remains small (Section 4.5).

4.3 Quality–Efficiency Trade-offs

Table 1 evaluates each method at a single configuration. To examine whether the baselines can reach a better trade-off by adjusting their thresholds, Figure 4 sweeps the confidence threshold of Fast-dLLM and the entropy budget of EB-Sampler.

Baseline trade-offs. The default configurations of both baselines lie at the conservative end of their curves. Relaxing the thresholds increases throughput but reduces accuracy, and the drop is steep on code generation: on LLaDA, Fast-dLLM falls from 40.2 to 24.8 on MBPP and from 43.29 to 27.44 on HumanEval as its threshold decreases from 0.9 to 0.6. A single confidence or entropy criterion cannot distinguish reliable predictions from unsupported ones, so admitting more tokens also admits more errors (Section 2).

Position of RPD. On LLaDA, RPD lies above the Fast-dLLM curve on all four benchmarks and on or above the EB-Sampler curve, with the clearest margin on MBPP. On Dream, RPD lies at the high-accuracy end of the frontier, matching the accuracy of the most conservative baseline configurations.

Table 2: Component ablation of RPD. Starting from confidence-based decoding, we add prediction persistence (PP), confidence drop (CD), and cumulative entropy (CE). Confidence (full canvas) commits all predictions with confidence of at least 0.9 without block structure. We report accuracy (Acc.), number of function evaluations (NFE), and throughput (TPS). On Dream, confidence decoding commits EOS tokens early and truncates the response, so few tokens count toward TPS.

Variant	LLaDA-8B-Instruct						Dream-7B-Instruct					
	GSM8K			HumanEval			GSM8K			HumanEval		
	Acc.↑	NFE↓	TPS↑	Acc.↑	NFE↓	TPS↑	Acc.↑	NFE↓	TPS↑	Acc.↑	NFE↓	TPS↑
Confidence	54.59	88.02	28.65	27.44	48.93	31.03	35.56	105.89	1.69	21.95	90.07	3.57
+ block32	76.35	77.09	35.63	43.29	44.48	43.13	81.73	76.04	30.03	60.37	64.54	31.69
+ PP	74.00	44.80	57.50	31.10	28.88	64.57	71.87	48.41	43.53	48.78	47.97	40.06
+ CE	78.92	80.84	33.23	43.29	45.23	42.46	81.88	88.03	24.53	60.98	65.62	29.75
+ PP + CD	75.74	69.08	38.55	41.46	41.23	46.23	81.27	70.78	30.82	59.76	61.72	31.71
Full RPD	78.85	71.76	35.30	43.90	40.59	45.29	81.80	82.03	24.56	60.98	63.09	30.07

Here the gains are smaller because confidence-based decoding already retains the accuracy of default decoding, leaving limited room for improvement. RPD-block shifts toward higher throughput at slightly lower accuracy, consistent with Section 4.2. LoPA and DAPD lie below or to the left of the frontiers in all settings.

4.4 Ablation Studies

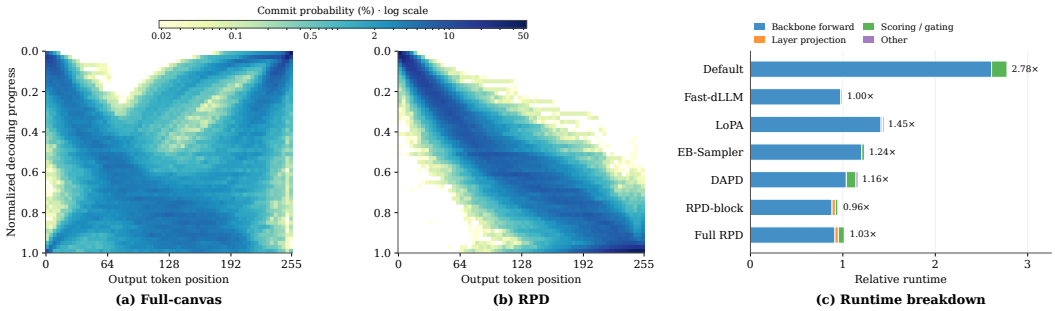
Table 2 starts from confidence-based decoding over the full canvas and adds ordering constraints and the components of RPD: prediction persistence (PP, Eq. 2), confidence drop (CD, Eq. 3), and cumulative entropy (CE, Eq. 1). Without any ordering constraint, confidence-based decoding degrades sharply: removing the block structure lowers accuracy from 76.35 to 54.59 on LLaDA GSM8K and from 81.73 to 35.56 on Dream GSM8K. Confident predictions near the end of the sequence are committed before their supporting computations (Figure 5a), and on Dream these early commitments are often EOS tokens that truncate the response. Block-wise decoding avoids this failure through a fixed order. CE also decodes over the full canvas, yet matches or exceeds the accuracy of block-wise decoding in all four settings, with the largest gain on LLaDA GSM8K (+2.6 points). Deferring candidates with uncertain upstream context thus provides an adaptive commitment order without a fixed block schedule, at the cost of more forward evaluations.

Layerwise stability recovers this parallelism. Within blocks, PP alone reduces NFE by 26–42% but causes large accuracy drops, for example from 43.29 to 31.10 on LLaDA HumanEval, because it admits predictions that keep the same token across layers while losing probability. Adding CD removes most of these predictions, recovering most of the lost accuracy while requiring fewer forward evaluations than block-wise confidence decoding in all four settings (Table 2). Combining the two, full RPD retains the accuracy of CE while reducing NFE by 4–11% and matching or increasing throughput in all four settings.

4.5 Decoding Behavior and Overhead

Commitment order. Figure 5a–b shows when each output position is committed on LLaDA GSM8K. Full-canvas commits both ends of the sequence early and fills the middle last, producing an X-shaped pattern. Positions near the end, where the final answer is usually written, are often committed before the reasoning that supports them, which is the premature commitment illustrated in Figure 1. RPD also decodes over the full canvas, yet commits positions in an approximately left-to-right order. The cumulative entropy budget thus recovers a dependency-respecting order without a fixed block schedule. Per-sample trajectories are provided in Appendix A.7.

Runtime overhead. Figure 5c decomposes the runtime of each method. The backbone forward pass dominates in all cases. Layerwise projection and scoring add only a small cost, so RPD runs at $1.03\times$ the runtime of Fast-dLLM and RPD-block at $0.96\times$. In comparison, LoPA, EB-Sampler, and DAPD require $1.45\times$, $1.24\times$, and $1.16\times$, respectively.



(a) Full-canvas **(b) RPD** **(c) Runtime breakdown**

Figure 5: (a, b) Commitment order of Full-canvas and RPD on LLaDA GSM8K. Each cell shows the percentage of samples in which an output position is committed at a given normalized decoding progress. (c) Runtime of each method relative to Fast-dLLM, decomposed into backbone forward, layer projection, scoring and gating, and other operations.

5 Related Work

Parallel decoding in masked diffusion language models. MDLM (Sahoo et al., 2024) develops masked diffusion for language modeling, while LLaDA (Nie et al., 2026) and Dream (Ye et al., 2025) demonstrate its use in instruction following, mathematical reasoning, and code generation. Fast-dLLM (Wu et al., 2026) combines approximate KV caching with confidence thresholds for parallel decoding. EB-Sampler (Ben-Hamu et al., 2026) controls parallel unmasking through an entropy budget supported by an analysis of sampling error. Other methods search over decoding decisions. LoPA (Xu et al., 2025) uses parallel lookahead to find token-filling orders that allow more parallel decoding in later steps. Ripple-Pivot Search (Ye et al., 2026) searches for pivot positions and token assignments that reduce uncertainty at other masked positions. Unlike these search-based methods, we assess which tokens to commit using upstream uncertainty and layerwise predictions from the current forward pass, without exploring alternative continuations.

Dependencies and commitment order. DAPD (Kim et al., 2026) selects independent sets from an attention-based dependency graph to avoid jointly updating strongly coupled positions. DOS (Zhou et al., 2026) orders masked positions by their dependence on observed context. Yeom et al. (2026) show that committing answers too early can reduce reasoning accuracy and studies frontier gating to address this problem. We also study premature commitment, but distinguish dependencies that require further decoding steps from those that can be resolved within one forward pass. This distinction motivates separate checks for upstream uncertainty and prediction stability, allowing parallel commitment when further decoding steps are unnecessary.

Layerwise predictions and stability. The tuned lens (Belrose et al., 2023) uses learned affine probes to obtain vocabulary distributions from intermediate layers, while DoLa (Chuang et al., 2024) contrasts predictions across layers to improve factuality. Stability-Weighted Decoding (Wu & Huang, 2026) uses distributional changes between consecutive denoising steps to score tokens. We instead examine prediction consistency across layers within one forward pass and combine it with final confidence to select tokens for commitment. We use layerwise predictions to choose positions, without training probes or modifying token distributions through layer contrast.

6 Conclusion

We studied when masked diffusion language models can reliably commit multiple tokens in parallel. Our diagnostics show that confidence alone does not determine a reliable commitment order: downstream predictions are less reliable when their upstream information remains uncertain, and predictions that remain stable across layers are more likely to be correct. Based on these findings, we proposed Reliable Parallel Decoding (RPD), a training-free method that selects candidates by layerwise stability and orders their commitment by a cumulative entropy budget. Without a fixed block schedule, RPD commits tokens in a dependency-respecting order and achieves the highest throughput among the evaluated methods while maintaining or improving accuracy.

AI USE STATEMENT

We used generative AI tools in two ways: to aid and polish the writing of this paper, and to assist with retrieval and discovery of related work. All AI-assisted text was reviewed and revised by the authors, and every work suggested through AI-assisted search was read and verified by the authors before being cited. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

RPD is training-free and adds no parameters, so reproducing our results requires only the public checkpoints and the decoding code. The anonymous repository linked in the abstract contains the full decoder, together with the layerwise readout, the cumulative entropy gate, and a command-line entry point. Appendix A.5 specifies the evaluation protocol, metric definitions, and hardware; Appendix A.5.1 states the matched configuration of every baseline; Appendix A.1 reproduces the four zero-shot instructions and both chat templates verbatim. The hyperparameters of RPD, including the two model-specific values, are listed in Table 4, and the layer ranges and projection restrictions in Appendix A.6. For the diagnostics, Appendix A.2 gives the sampling seed, the masking construction, and the digit-tokenization procedure that makes the masked states identical across the two backbones, and Appendix A.3 reports the complete results with question-level bootstrap intervals. All decoding is greedy at temperature 0 unless a model's default sampler is explicitly used, in which case its settings are stated with the result.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *Advances in Neural Information Processing Systems*, 38:55981–56007, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54158–54183, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Bumjun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. Dapd: Dependency-aware parallel decoding via attention for diffusion llms. *arXiv preprint arXiv:2603.12996*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

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.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

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.

Yue Wu and Jian Huang. Stability-weighted decoding for diffusion language models. *arXiv preprint arXiv:2604.17068*, 2026.

Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, and Zhijie Deng. Lopa: Scaling dllm inference via lookahead parallel decoding, 2025. URL https://arxiv.org/abs/2512.16229.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Yushi Ye, Xu Chen, Haoyun Jiang, Jinsong Lan, Haihong Tang, Xiangtao Li, Mingming Gong, Ivor Tsang, Yanfeng Wang, and Jiangchao Yao. Ripple-pivot search: Active parallel decoding for diffusion large language models, 2026. URL https://arxiv.org/abs/2608.11742.

Jewon Yeom, Jaewon Sok, Seonghyeon Park, Jeongjae Park, Hwiyeong Lee, and Taesup Kim. Answer first, reason later: When commitment order costs accuracy in diffusion language models, 2026. URL https://arxiv.org/abs/2608.05687.

Xueyu Zhou, Yangrong Hu, and Jian Huang. Dos: Dependency-oriented sampler for masked diffusion language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17404–17419, 2026.

A Appendix

A.1 Prompts

All benchmarks use a single zero-shot instruction, filled with the raw benchmark field and wrapped in the model's native chat template with `add_generation_prompt=True`; the filled instruction is the only user turn. Figures 6–9 give the four instructions verbatim, and Figures 10–11 give the chat wrapping of each backbone. LLaDA receives no system turn; Dream's tokenizer template prepends its own default turn, `You are a helpful assistant.` We add no system message, demonstration, assistant prefill, or input truncation beyond this.

Prompt template for GSM8K

System instruction

None for LLaDA. For Dream the tokenizer chat template prepends its default turn: "You are a helpful assistant."

User template

```
Solve the following problem step by step.

Problem:
{question}

End your response with a line in the format:
#### <answer>
where <answer> is only the final numerical answer, without units.
```

Figure 6: GSM8K instruction.

Prompt template for MATH-500

System instruction

None for LLaDA. For Dream the tokenizer chat template prepends its default turn: "You are a helpful assistant."

User template

```
Solve the following mathematics problem step by step.

Problem:
{problem}

Put your final answer in \boxed{} on the last line.
```

Figure 7: MATH-500 instruction.

A.2 Diagnostic Data and Implementation

Data construction. We deterministically sample 2,000 examples from the GSM8K-Aug training set using seed 325. Each example contains a verified multi-step arithmetic trace and an integer final answer. We require at least 48 eligible upstream digit positions and exclude examples in which the final answer appears literally in the question, an earlier equation, or the left-hand side of the final equation.

The input consists of the question followed by the gold reasoning trace, with the right-hand side of the final equation serving as the target answer. For each example, we independently

Prompt template for MBPP

System instruction

None for LLaDA. For Dream the tokenizer chat template prepends its default turn: "You are a helpful assistant."

User template

```
Write a Python solution for the following task.

Task:
{text}

Your solution must pass these tests:
{test_list}

Return only the Python code, including any necessary imports.
Do not include Markdown fences, explanations, or test calls.
```

Figure 8: MBPP instruction. {test_list} is filled with the first three public tests.

Prompt template for HumanEval

System instruction

None for LLaDA. For Dream the tokenizer chat template prepends its default turn: "You are a helpful assistant."

User template

```
Implement the Python function specified below.
Preserve the function name and signature, and follow the specification
and examples in the docstring.

{prompt}

Return the complete Python implementation, including any necessary imports.
Do not include Markdown fences, explanations, or test calls.
```

Figure 9: HumanEval instruction.

sample 16 upstream masking patterns for each mask count $m \in \{4, 8, 16, 32\}$. Selected upstream digits and all target-answer digits are masked simultaneously; all other positions retain their gold tokens. This produces 64 states per question and 128,000 states per model. LLaDA and Dream use identical questions and character-level masking patterns, mapped into each tokenizer through explicit digit tokenization. All experiments in this paper use the public instruction-tuned checkpoints `GSAI-ML/LLaDA-8B-Instruct` (revision `08b83a6`) and `Dream-org/Dream-v0-Instruct-7B` (revision `05334cb`), in bfloat16. Masking patterns are defined over characters so that both backbones see the same masks. To map them into each tokenizer we first verify that every digit `0–9` encodes to exactly one token in that vocabulary, then build the input by explicit segmentation: the reasoning prefix is encoded in runs, and each digit character is emitted as its own single-token encoding, recording the resulting token index for that character. Digits are masked only inside arithmetic content, never in step-number labels, and every target-answer digit is emitted the same way. A masking pattern therefore selects the same characters, and hence positionally corresponding tokens, in both models; we assert that all selected upstream positions precede the first target position.

Chat wrapping for LLaDA-8B-Instruct

Chat template

Applied with the tokenizer chat template and add_generation_prompt=True. The filled task template is the sole user turn.

```
<|startoftext|><|start_header_id|>user<|end_header_id|>

<FILLED TASK TEMPLATE><|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Figure 10: Chat wrapping for LLaDA-8B-Instruct.

Chat wrapping for Dream-v0-Instruct-7B

Chat template

Applied with the tokenizer chat template and add_generation_prompt=True. The filled task template is the sole user turn.

```
<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
<FILLED TASK TEMPLATE><|im_end|>
<|im_start|>assistant
```

Figure 11: Chat wrapping for Dream-v0-Instruct-7B.

Prediction and scoring. Every masked state is evaluated with one independent forward pass. Diagnostic probabilities are taken before temperature scaling, sampling noise, or top-p truncation. For the upstream-entropy analysis, a state is correct only if the complete decoded numerical answer matches the gold answer. For the layerwise analysis, a target token is correct if its final-layer argmax matches the gold token. There are 531,264 target-token observations per model.

Intermediate-layer readout. We apply the model's final normalization and language-model head to intermediate hidden states. At the final layer, we use native logits without applying final normalization twice. Dream predictions follow its native one-position logit alignment. The trajectory analysis uses the final half of the transformer layers. Concretely, this is layers 17–32 of LLaDA's 32 layers and layers 15–28 of Dream's 28 layers, in both cases the final layer together with the posterior half of the stack, giving 16 and 14 trajectory points respectively.

Let y_i denote the final-layer argmax at position i. Scanning backward from the final layer, we count consecutive layers predicting y_i until the first disagreement or the beginning of the analysis window. This count is K_i, and the suffix starts at $s_i = L - K_i + 1$. Earlier matches separated by a disagreement are excluded. The peak-to-final drop r_i is computed from the probability of the same token y_i throughout this suffix.

A.3 Complete Diagnostic Results

A.3.1 Upstream Entropy

For each model and mask count, we retain questions whose 16 masking patterns contain both correct and incorrect answers. Within each retained question, patterns are ranked by cumulative upstream entropy. We average answer-error labels over the five lowest-entropy patterns and separately over the five highest-entropy patterns. The reported rates average these question-level values with equal

Table 3: Complete upstream-entropy results. n is the number of retained mixed-outcome questions. Error rates are percentages; Δ is the paired high-minus-low difference in percentage points, computed before rounding.

Model	Masks	n	Low entropy	High entropy	Δ
LLaDA	4	40	22.5	51.0	28.5
	8	81	14.6	38.8	24.2
	16	182	12.7	37.6	24.8
	32	813	10.5	39.0	28.6
Dream	4	128	15.8	33.1	17.3
	8	322	9.3	27.2	17.9
	16	762	9.4	27.7	18.3
	32	1513	21.1	51.0	29.9

weight. We compute 95% confidence intervals using 2,000 question-level bootstrap resamples, preserving the low–high pairing within each question. The comparison does not match configurations by target confidence or mask distance.

Table 3 reports all groups. High-entropy configurations have higher error rates for both models at every mask count, and all paired differences have 95% confidence intervals above zero.

Paired differences with 95% question-level bootstrap intervals, in percentage points, are 28.5 $[18.0, 38.5]$, 24.2 $[18.0, 29.9]$, 24.8 $[21.0, 28.9]$ and 28.6 $[26.5, 30.5]$ for LLaDA, and 17.3 $[13.9, 20.9]$, 17.9 $[15.5, 20.2]$, 18.3 $[16.4, 20.2]$ and 29.9 $[28.4, 31.5]$ for Dream, at 4, 8, 16 and 32 masks respectively. Every interval lies above zero.

A.3.2 Layerwise Agreement and Probability Drop

The agreement curves group target tokens by final confidence into $[0.6, 0.7)$, $[0.7, 0.8)$, $[0.8, 0.9)$, $[0.9, 0.95)$, and $[0.95, 1]$. For each interval and threshold $k \in \{2, \ldots, 10\}$, we report the token error rate among observations satisfying $K_i \geq k$. These selections are nested, and empty selections are omitted. The curves are token weighted.

The Dream heatmap includes observations with final confidence in $[0.6, 0.9)$, $K_i \geq 2$, and $r_i \geq 0.08$. Rows group agreement lengths into 2, 3–4, 5–8, and ≥ 9. Columns use drop intervals $[0.08, 0.16)$, $[0.16, 0.24)$, $[0.24, 0.32)$, and $[0.32, 0.40]$. Each cell reports the token-weighted error rate and observation count.

Two patterns follow. First, within a fixed confidence interval, requiring a longer agreement suffix lowers the error rate, and the effect is largest exactly where the decision is hardest: in the $[0.6, 0.7)$ interval Dream's error rate falls from 26.5% at $K_i \geq 2$ to 4.6% at $K_i \geq 10$, while in $[0.95, 1]$ the curve is already flat, because such predictions are reliable whether or not they persist across layers. This is what motivates applying the stability test only below θ_h. Second, the two quantities interact: a large confidence drop is tolerable when the prediction has persisted for many layers but not when it has just appeared, which is why the score combines them as $\min(K_i, K_{\max}) - w\, r_i$ rather than thresholding either alone.

A.4 Commitment Order under Model-Specific Decoding

All traces in this section commit exactly one position per forward pass on a 256-position canvas, for the first GSM8K document, so the panels differ only in the setting named in their title.

LLaDA: with and without blocks. LLaDA's block-wise decoder processes blocks sequentially, preventing commitment in later blocks before the active block is completed. Figure 12 contrasts this with the same greedy maximum-confidence rule applied to the full canvas. With blocks of 32 the commitment order is a clean left-to-right diagonal. Without blocks the order is still broadly left-to-right, but the second half of the canvas receives scattered late commits that interleave with positions committed much earlier: whole spans of the arithmetic are still dark while the text around them is

already light. The block schedule, not the confidence rule, is what enforces the order. This trace reproduces the archived default run token for token.

Dream: the role of the default sampler. Dream has no block structure; its default sampler draws from the distribution obtained after temperature 0.1 and top-p 0.9, and commits the position whose transformed distribution has the lowest entropy. Under this configuration we observe an approximately left-to-right commitment order (Figure 13, top).

The temperature is not an incidental sampling detail. Setting it to zero, with no top-p truncation, and keeping the same lowest-entropy rule reverses the order (Figure 13, bottom): the trailing terminal positions have the lowest entropy on the raw distribution, so they are committed first and the canvas fills from right to left: in Figure 13 the terminal tokens are the lightest of the panel and the only content left is the three tokens of a truncated answer. The ordering score must therefore be read from the same transformed distribution the token is drawn from.

A.5 Detailed Experimental Configuration

Prompts and evaluation. All experiments use dataset-specific zero-shot instructions and the model's native chat template, without demonstrations, a system message, assistant prefill, or input truncation. The GSM8K prompt asks for step-by-step reasoning and requires a final line of the form `#### <answer>`. The MATH-500 prompt similarly requests a derivation followed by a `\boxed{}` answer. For MBPP, the prompt supplies the task and its public tests and requests Python code only. For HumanEval, it supplies the canonical function signature and docstring and requests a complete implementation without Markdown fences, explanations, or test calls.

GSM8K responses are scored by extracting the final answer and performing exact numeric comparison, treating equivalent decimal representations identically. MATH-500 uses symbolic answer equivalence. HumanEval and MBPP are evaluated by executing the generated Python implementation against their tests; we apply the same formatting-only code-fence removal to every method. NFE counts actual model forward calls, including LoPA branch evaluations. TPS excludes model loading, tokenization, and task scoring. Generation is synchronized before timing, and each throughput run occupies one NVIDIA A6000 GPU.

Default decoding. Every default run commits one token per forward evaluation. LLaDA uses greedy maximum-confidence decoding with blocks of 32. Dream uses its official full-canvas entropy sampler with temperature 0.1 and top-$p = 0.9$. Both models generate on a 256-position canvas.

A.5.1 Matched baseline configuration.

For a controlled comparison, accelerated baselines use blocks of 32 and the same 256-token budget.

Fast-dLLM commits, in every forward pass, all masked positions in the active block whose final confidence exceeds 0.9, and otherwise the single most confident position. The threshold is applied per position and independently of how many positions are already committed in the same pass. Our evaluation disables KV caching so that NFE and token-selection effects can be compared directly.

EB-Sampler sorts the masked positions of the active block by an error proxy (confidence, in our runs) and commits the longest prefix of that order whose marginal entropies satisfy $\sum_{i \in U} H_i - \max_{i \in U} H_i \leq \gamma$, with $\gamma = 0.1$ nats. The budget is shared by the whole prefix, so the least certain admitted position is exempt and every additional position consumes budget. A single position is therefore always affordable.

LoPA keeps a confidence anchor that commits all positions above 0.9, and additionally spawns three lookahead branches, each extending the anchor by one greedy token at a distinct high-scoring position. All four branches are verified in one batched forward pass and scored by the mean negative entropy of the remaining masked positions of the block; the winning branch is retained and its logits are reused in the next iteration. Every branch microbatch contributes to NFE.

DAPD uses the direct variant, which derives a dependency graph over the masked positions from self-attention and commits an independent set of that graph, with the official task-specific depen-



Figure 12: LLaDA commitment order with and without block structure, one commit per forward pass. Each box is one generated token, shaded by the step at which it was committed.

dency thresholds. Unless explicitly specified above for the native Dream sampler, matched accelerated baselines use greedy sampling without top-p truncation.

A.6 Implementation Details of RPD

Hyperparameters. Table 4 lists the hyperparameters of RPD. All values are shared across benchmarks. The stability threshold θ_s and the confidence-drop weight w are the only model-specific values; they were set once per backbone and then frozen for every benchmark.

Choice of confidence thresholds. The thresholds θ_c and θ_h follow the diagnostics in Figure 2b. Predictions with final confidence of at least 0.9 have near-zero error rates regardless of prediction persistence, so a stability test adds little information for them. We therefore set $\theta_h = 0.9$ and admit

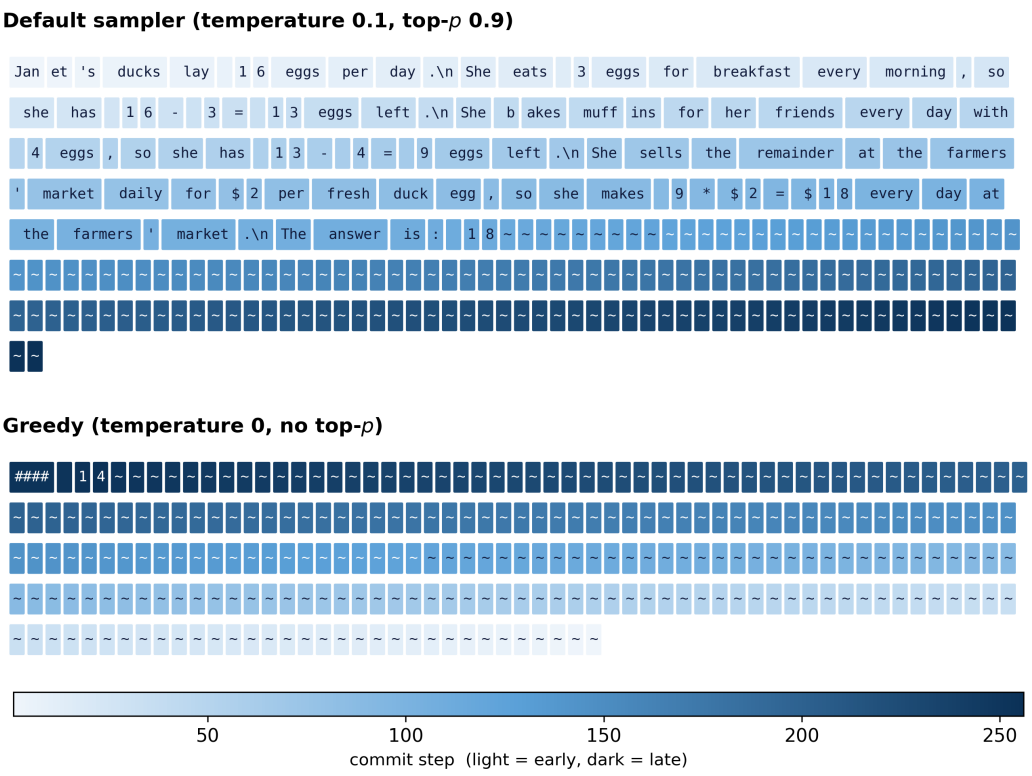


Figure 13: Dream commitment order under its default sampler and under greedy decoding, one commit per forward pass and lowest-entropy-first selection in both panels. Each box is one generated token, shaded by the step at which it was committed; ˜ marks a terminal token.

such predictions directly. For lower confidence, persistence becomes informative. In the lowest analyzed range, $[0.6, 0.7)$, Dream's token error rate decreases from 26.5% at $K_i \geq 2$ to 4.6% at $K_i \geq 10$, reaching levels comparable to those of higher-confidence predictions. We therefore set $\theta_c = 0.6$ and allow predictions in $[0.6, 0.9)$ to become candidates when they pass the stability test. Predictions below θ_c are never admitted in the current iteration.

Choice of the stability threshold and drop weight. Unlike θ_c and θ_h, which are read directly off the confidence axis of the diagnostics, θ_s and w describe how the two layerwise quantities should be traded against each other, and we set them from the same analysis in Section 2.2. Two observations fix their form. First, the benefit of persistence saturates: beyond a handful of agreeing layers the error rate is already flat, so the score caps persistence at $K_{\max} = 6$ and θ_s only has to separate predictions that have settled from those that have not. Second, the tolerable confidence drop depends on how long the prediction has persisted; for $K_i = 5$–8 the error rate rises from 4.6% to 23.7% across the drop range, while for $K_i \geq 9$ the same range costs only a few points. A single linear penalty $w\, r_i$ reproduces this coupling: under a fixed threshold, a longer suffix buys tolerance for a larger drop, up to the cap. We chose w so that the implied ceiling $(K_{\max} - \theta_s)/w$ sits just above the drop region that Figure 2c still shows as low risk, giving 0.167 for LLaDA and 0.175 for Dream. The two backbones need slightly different values because their trajectories saturate at different depths, which is also why ℓ_0 differs. The resulting pair was then frozen for both models, all four benchmarks and both variants; no benchmark-specific or task-specific tuning is performed anywhere in the paper. Figure 15 confirms that the committed predictions sit well inside the implied ceiling.

Layerwise projection. We compute K_i and r_i by projecting the hidden states of layers $\ell_0, \ldots, L$ through the model's final normalization and output head, following the procedure in Section 2.2. All projections reuse the hidden states of the current forward pass and require no additional forward evaluation. Two restrictions keep this inexpensive. First, only the currently masked positions are

Table 4: Hyperparameters of RPD.

Symbol	Description	Value
θ_h	High-confidence threshold (bypasses the stability test)	0.9
θ_c	Minimum confidence for stability-based admission	0.6
θ_s	Stability threshold	3.5 (LLaDA) / 2.5 (Dream)
$K_{\max}$	Cap on prediction persistence	6
w	Weight of the confidence-drop penalty	15 (LLaDA) / 20 (Dream)
β	Cumulative entropy budget	4 nats
ℓ_0	First layer used for layerwise projection	17 (LLaDA) / 15 (Dream)
W	Fallback window when no candidate is committable	32

projected, never the already committed prefix. Second, only the posterior half of the stack is projected, and the final layer reuses the raw logits of the forward pass rather than being normalized a second time. The projection is further evaluated lazily, so that layers are materialized only until the agreement suffix of a position is determined. Figure 5c reports the resulting runtime overhead.

What the stability test actually admits. Because the admission rule is $S_i = \min(K_i, K_{\max}) - w\, r_i \geq \theta_s$ with $K_{\max} = 6$, the thresholds impose a hard ceiling on the tolerated confidence drop: $r_i \leq (6 - \theta_s)/w$, which is $1/6 \approx 0.167$ for LLaDA and 0.175 for Dream. The distributions below are therefore selection-induced and are not unconditional model statistics. Figures 14–15 show them over all committed stability-admitted tokens of the full runs. On LLaDA GSM8K, RPD-block admits 29,957 tokens through the stability route against 273,423 through the high-confidence route and 34,284 by fallback; the admitted tokens have median confidence 0.858 $[0.774, 0.892]$ at the 10th and 90th percentiles, median persistence $K_i = 7$, and median drop 0.117 with a 90th percentile of 0.156. Full RPD on the same cell is nearly identical (27,991 stability commits, median confidence 0.861, median drop 0.116), and Dream behaves the same way at its own ceiling (median drop 0.125, 90th percentile 0.165). No committed token exceeds a drop of 0.25 in any cell. The stability route therefore contributes roughly 8–10% of all commits, concentrated in the intermediate confidence band it was designed for.

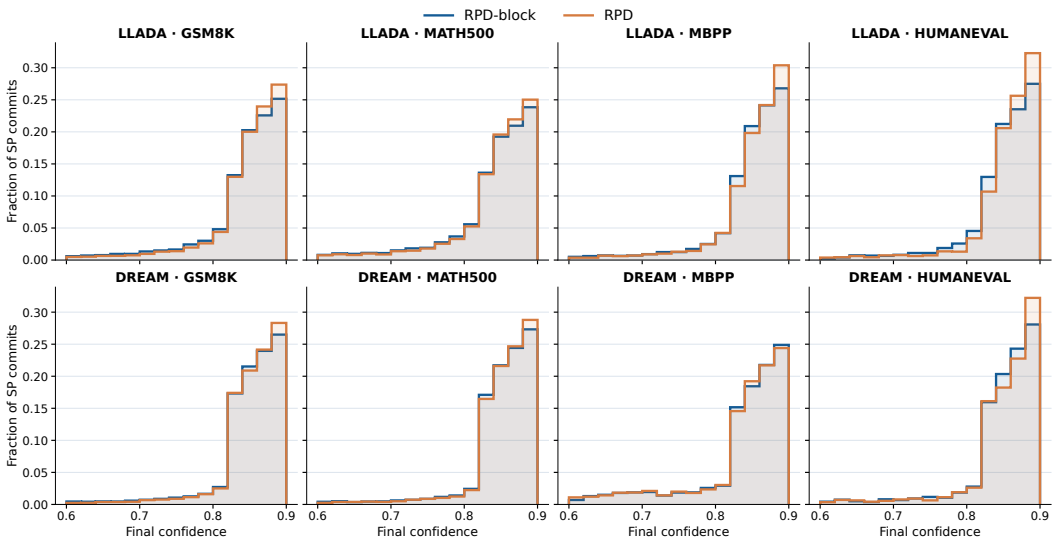


Figure 14: Final confidence of tokens admitted by the stability test.

RPD-block. RPD-block applies the candidate selection of Section 3.1 within fixed decoding blocks of 32 tokens and omits the cumulative entropy budget, relying on the block order alone. It uses the same θ_h, θ_c, θ_s, $K_{\max}$, w and ℓ_0 as Full RPD; only the commitment rule differs. When no candidate passes, it falls back to the most confident masked position of the active block.

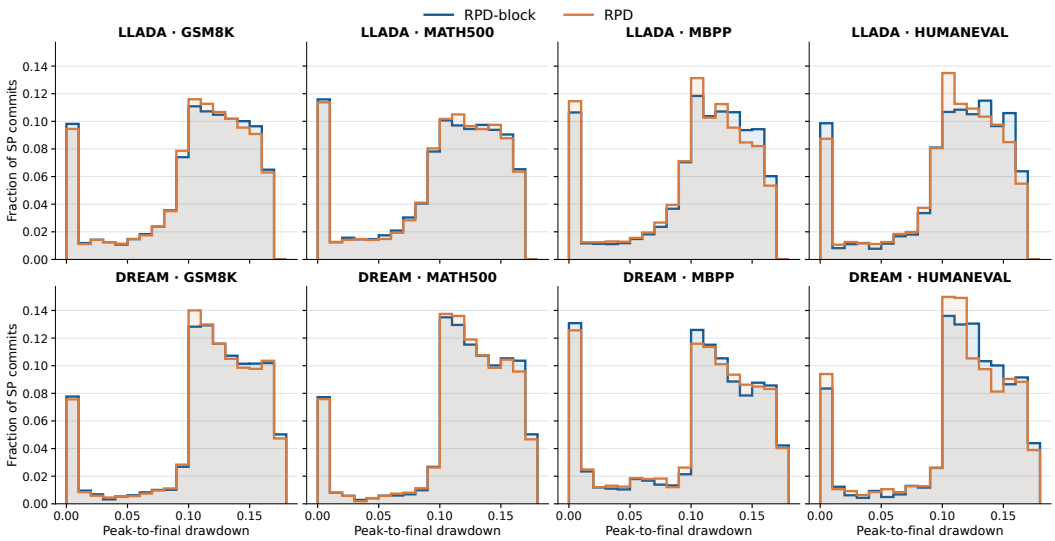


Figure 15: Confidence drop r_i of tokens admitted by the stability test. The upper limit in each panel is the selection ceiling $(K_{\max} - \theta_s)/w$.

A.7 Trajectories Visualization

Figure 16 places the three decoders side by side on the same GSM8K document, plotting the step at which each output position is committed. Committing one token per forward pass over the full canvas produces a broadly left-to-right order, but the second half of the canvas receives scattered late commits. Fast-dLLM's block structure removes that scatter and yields a strict block-by-block staircase, at the cost of never committing outside the active block. RPD recovers the same left-to-right progression on the full canvas without any block schedule, and does so in fewer steps: 95 forward passes against 114 for Fast-dLLM and 256 for one-token-per-forward decoding. The cumulative entropy budget is therefore doing the work that the block schedule does for Fast-dLLM, while remaining free to commit anywhere on the canvas when the upstream context allows it.

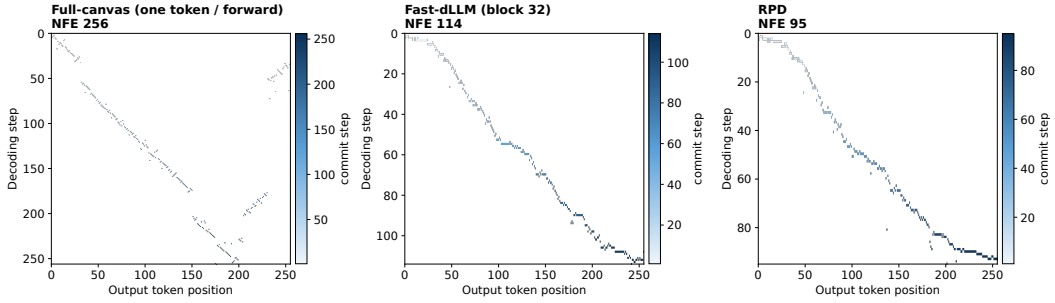


Figure 16: Commit step of each output position under three decoders, LLaDA on GSM8K (first document). Lower is earlier.