RIPPLE-PIVOT SEARCH: ACTIVE PARALLEL DECODING FOR DIFFUSION LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion Large Language Models (dLLMs) have emerged as a competitive alternative to autoregressive language models, offering the potential for substantially faster inference through parallel decoding. Existing parallel decoding schedulers typically commit positions only after they meet a per-position criterion, overlooking how early commitments may benefit subsequent decoding. We identify a ripple effect in dLLM decoding: proactively committing a *mid-entropy* pivot position can induce a pronounced reduction in uncertainty across the remaining masked positions. This uncertainty reduction allows subsequent steps to unmask more tokens in parallel, thereby accelerating the overall decoding process. To exploit the ripple effect, we propose Ripple-Pivot Search (RPS), a novel training-free decoding method that seeks mid-entropy positions as promising candidate pivots (*where to decode*), and determines their token assignment that yields the greatest downstream benefit via lookahead evaluation (*what to decode*). Across 3 dLLMs and 4 reasoning and code-generation benchmarks, RPS achieves 4–10$\times$ wall-clock speedup over the default decoder while preserving generation quality, and improves accuracy over the previous lookahead baseline by up to 5.49% while delivering higher throughput in most settings. When integrated with KV caching, RPS further achieves up to 18$\times$ wall-clock speedup over the default decoder.

1 INTRODUCTION

Diffusion large language models (dLLMs) (Nie et al., 2025; Zhu et al., 2026a; Ye et al., 2025; Cheng et al., 2026; Bie et al., 2025) have gained prominence as a viable alternative to autoregressive language models (Brown et al., 2020; Yang et al., 2025; Zhao et al., 2026), offering the potential for faster inference by decoding multiple tokens in parallel. In each denoising step, the model produces predictions at all masked positions simultaneously, and a decoding scheduler selects which positions to unmask and assigns tokens at those positions. In practice, however, committing more positions per step increases the risk of error accumulation (Li et al., 2026), making the balance between decoding speed and generation quality a central challenge in dLLM inference.

To navigate this speed-quality trade-off, a growing body of work on parallel decoding (Hong et al., 2025; Wu & Zhang, 2025; Ye et al., 2026; Zhu et al., 2026b) has focused on designing schedulers that commit multiple positions per step under reliability constraints, thereby amortizing the cost of each forward pass. The scheduler is thus responsible for two decisions: *where* to unmask and *what* token to assign. Most schedulers couple the two, committing each position to its greedy prediction once a per-position criterion is satisfied, such as sufficient confidence (Wu et al., 2025; Nie et al., 2025), low predictive entropy (Ye et al., 2025), cross-step stability (Kim et al., 2025; Wei et al., 2025), or bounded cumulative entropy (Ben-Hamu et al., 2025). More recent lookahead-based methods (Xu et al., 2025; Fu et al., 2025) further test whether committing an additional position can benefit subsequent decoding. However, these methods employ lookahead mainly to determine whether and where to commit. Once a position is selected, its token is typically fixed to the model's current top-1 prediction. As a result, the search explores different commitment positions but not alternative token assignments, potentially overlooking effective non-greedy decoding trajectories.

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.

Method family	Where to decode	What to decode
Confidence-based (Wu et al., 2025; Nie et al., 2025)	Confidence-qualified positions	Top-1 (greedy)
Entropy-based (Ye et al., 2025; Ben-Hamu et al., 2025)	Entropy-qualified positions	Top-1 (greedy)
Stability-based (Kim et al., 2025; Wei et al., 2025)	Cross-step stable positions	Top-1 (greedy)
Lookahead-based (Xu et al., 2025; Fu et al., 2025)	Lookahead-selected positions	Top-1 (greedy)
RPS (ours)	**Mid-entropy pivot**	**Lookahead selected (non-greedy)**

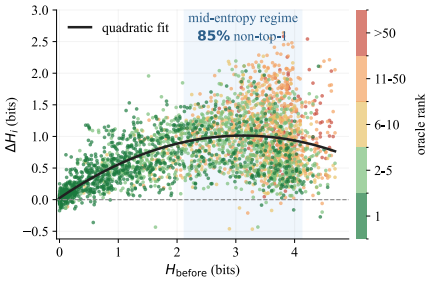

Figure 1: **Left:** Comparison of parallel decoding schedulers. **Right:** Starting from a partially decoded block, we commit each remaining masked position to its oracle token, i.e., the token eventually committed under the standard decoding procedure, and measure the resulting entropy reduction ΔH_i at other masked positions (y-axis) against the pre-commit entropy H_{before} of the resolved position (x-axis). Colors indicate the oracle token's rank under the model distribution, and the shaded band marks the mid-entropy regime.

The benefit of early commitment depends critically on which unresolved position is selected. To characterize this, we conduct an oracle analysis over candidate commitments in Fig. 1 (right). This analysis reveals a distinct pattern that we refer to as the *ripple effect*: proactively committing a pivot position in the **mid-entropy regime** induces the strongest downstream uncertainty reduction. Intuitively, such positions are not fully determined, but are already sufficiently tied to the current partial decoding state; resolving them can therefore influence other masked positions more strongly than positions that are either already certain or still weakly constrained. Moreover, the oracle token is not the model's top-1 prediction in 85% of mid-entropy cases, revealing a mismatch with existing lookahead-based schedulers. As shown in Fig. 1 (left), these methods use lookahead to decide *where* to commit while fixing *what* to the current top-1 prediction. They therefore search over commitment positions but not token assignments, potentially missing beneficial non-greedy trajectories.

Motivated by these findings, we propose **Ripple-Pivot Search (RPS)**, a training-free parallel decoding method that exploits beneficial early commitments in the mid-entropy regime. RPS first applies a two-stage pivot filter to exclude positions that are either overly certain or insufficiently supported by the current predictive distribution, thereby focusing in the mid-entropy regime. Given this pivot, RPS then constructs an adaptive set of plausible token assignments, evaluates their downstream benefits with a single lookahead forward pass, and selects the most promising token for early commitment. This allows RPS to choose the assignment expected to best amplify the ripple effect. Furthermore, to prevent premature commitments from accumulating errors, RPS commits a token only when it improves over leaving the pivot unmasked. In summary, our contributions are:

- We identify a *ripple effect* in dLLM decoding: proactively committing a position in the mid-entropy regime induces the strongest downstream uncertainty reduction, enabling more parallel commits in subsequent steps. Crucially, the oracle token in this regime is frequently not the current top-1 prediction, motivating token assignment beyond greedy decoding.
- We propose Ripple-Pivot Search (RPS), a novel training-free decoding method that seeks mid-entropy pivots with high potential for downstream uncertainty reduction, and determines its token assignment through lookahead-based evaluation on an adaptive candidate set, committing when the best candidate outperforms leaving the pivot unmasked.
- We conduct extensive experiments on three dLLMs across four reasoning and code-generation benchmarks. RPS achieves a **4–10**$\times$ inference speedup over the default one-token-per-step baseline while preserving generation quality, and improves accuracy over the previous lookahead baseline by up to **5.49%** while delivering higher throughput in most settings. When integrated with KV caching, RPS further achieves up to **18**$\times$ wall-clock speedup over the default decoder.

2 Related Work

Criterion-based Parallel Decoding. The most direct approach to accelerating dLLM inference commits multiple positions per decoding step based on per-position prediction statistics. Fast-dLLM (Wu et al., 2025) unmasks every position whose predictive confidence exceeds a fixed thresh-

old and introduces block-wise KV caching to reduce per-forward cost. KLASS (Kim et al., 2025) augments confidence with a KL divergence criterion that tracks prediction stability across consecutive steps, unmasking only positions that are both confident and temporally consistent. EB-Sampler (Ben-Hamu et al., 2025) controls the number of tokens unmasked per step by bounding the cumulative entropy of the newly committed positions. Learn2PD (Bao et al., 2025) further replaces fixed heuristics with a lightweight learned filter that predicts whether each current token prediction matches the final output, yielding an adaptive criterion for parallel unmasking.

Lookahead-based Parallel Decoding. Lookahead-based dLLM decoding accelerates inference by proposing token assignments and evaluating their downstream effects. WINO (Hong et al., 2025) drafts all positions admitted by a relaxed confidence threshold and evaluates the committed tokens under the enriched context, remasking those whose verification confidence falls below a stricter threshold. LoPA (Xu et al., 2025) forms lookahead branches from the highest-confidence positions that remain masked after the standard decoding update and selects the branch with the greatest future confidence. From an information-theoretic perspective, ETE (Fu et al., 2025) targets low-parallelism steps in which the standard decoding commits only a few tokens. At such steps, it explores high-information positions near a prescribed confidence level and selects the one that unlocks the most downstream high-confidence tokens. Yet these methods primarily use lookahead to decide *where* to commit with keeping greedy token assignments. By contrast, RPS focuses on mid-entropy regime and uses lookahead to jointly decide *where* to commit and *what* token to assign.

3 Preliminary

Notation. Consider a masked discrete diffusion language model with vocabulary $\mathcal{V}$ containing a special token [MASK]. Given a prompt y, let $x \in \mathcal{V}^L$ denote a length-L response in which each x_i is either decoded or masked. A forward pass on $[y\|x]$ yields a predictive distribution $p_i \triangleq p_\theta(\cdot \mid y, x)_i$ over $\mathcal{V}$ at each position i. We denote its entropy by $H(p_i)$ and its confidence (top-1 probability) by $P_i^{\max} \triangleq \max_{v\in\mathcal{V}} p_i(v)$.

Decoding. Standard practice (Nie et al., 2025; Bie et al., 2025; Wu et al., 2025) adopts a semi-autoregressive schedule: the length-L response is partitioned into contiguous blocks of size B and decoded left-to-right, fully unmasking each block before the next. At each step, let $\mathcal{M}$ denote the masked positions in the active block. Given the current $\{p_i\}_{i\in\mathcal{M}}$, a commit set $\mathcal{S} \subseteq \mathcal{M}$ is selected and each position in $\mathcal{S}$ is assigned its greedy (top-1) prediction,

$$x_i \leftarrow \arg\max_{v\in\mathcal{V}} p_i(v) \quad \text{for each } i \in \mathcal{S}. \tag{1}$$

Different schedulers are characterized by how they specify $\mathcal{S}$. The three most common rules are:

$$\mathcal{S} = \begin{cases} \underset{i\in\mathcal{M}}{\arg\text{top-k}}\, P_i^{\max}, & \text{highest-confidence decoding (Nie et al., 2025);} \\ \underset{i\in\mathcal{M}}{\arg\text{top-k}}\big(-H(p_i)\big), & \text{lowest-entropy decoding (Ye et al., 2025);} \\ \{i \in \mathcal{M} : P_i^{\max} \geq \tau\}, & \text{confidence-aware decoding (Wu et al., 2025).} \end{cases} \tag{2}$$

For confidence-aware decoding, an empty $\mathcal{S}$ triggers highest-confidence decoding with $k = 1$, ensuring progress. All three rules differ in how $\mathcal{S}$ is constructed but share the same token-assignment mechanism: each committed position receives its greedy prediction (Eq. 1). Positions not selected into $\mathcal{S}$ remain masked until they satisfy the criterion as decoding progresses.

4 Method

4.1 Ripple-Pivot Search

Our RPS introduces a per-step pivot search on top of the standard decoding process (instantiated with confidence-aware decoding in our experiments) as illustrated in Fig. 2, consisting of *pivot selection* (where to commit) and *lookahead scoring* (what to commit).

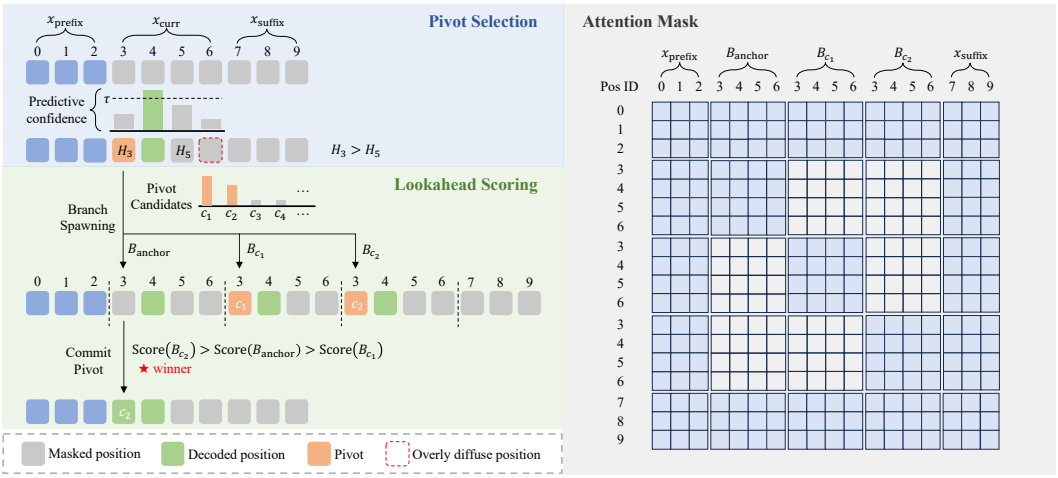

Figure 2: **Overview of RPS. Left**: One-step example of RPS decoding. The upper region illustrates pivot selection, while the lower region shows lookahead scoring. **Right**: Customized attention mask for lookahead forward pass. The packed sequence contains the shared context with candidate branches. Light-blue cells indicate allowed attention and blank cells indicate blocked attention.

Pivot selection. Pivot selection must balance benefit and cost: pivots should lie in the mid-entropy regime, where early resolution is most beneficial, while keeping lookahead token search compact. Although such positions are uncertain, the correct token typically remains among the top-ranked candidates. Motivated by this observation, RPS truncates each position's support to the top-$k_{\max}$ tokens, discarding most vocabulary items irrelevant to the decision, which is formulated as follows

$$i^\star = \underset{i \in \mathcal{M}:\, \mu_i \geq \tau_{\text{pivot}}}{\arg\max} \left\{ -\sum_{v \in \mathcal{T}_i} p_i(v) \log p_i(v) \right\}, \qquad \mu_i = \sum_{v \in \mathcal{T}_i} p_i(v), \tag{3}$$

where the truncated support $\mathcal{T}_i \subseteq \mathcal{V}$ contains the $k_{\max}$ tokens with the highest probabilities under predictive distribution p_i, and μ_i denotes the corresponding retained probability mass. With Eq. (3), the probability-mass constraint ($\mu_i \geq \tau_{\text{pivot}}$) excludes positions in the high-entropy regime, and maximizing truncated entropy over the retained positions avoids those that are already nearly determined. The resulting pivot therefore naturally falls in the mid-entropy regime to propagate useful information. If no position satisfies the probability-mass constraint, all remaining masked positions are still in a highly uncertain state where reliable intervention is infeasible. In this case, RPS skips pivot search at the current step and proceeds with standard decoding alone.

Lookahead scoring. With the pivot $i^\star$ identified, RPS then constructs an adaptive candidate set to determine the token assignment, where the set is defined as $\mathcal{C} = \{v | p_{i^\star}(v) \geq r * P_{i^\star}^{\max},\ \forall v \in \mathcal{T}_{i^\star}\} \cup \{\texttt{[MASK]}\}$ by retaining tokens that satisfy the reachability ratio r relative to the top-1 probability. The reason that we include [MASK] here is to allow the pivot to remain masked. As shown in Fig. 2, we build a corresponding lookahead branch B_c for each candidate token $c \in \mathcal{C}$ by assigning c to $i^\star$ (the branch induced by assigning [MASK] is referred to as the anchor branch B_{anchor}), and all branches are evaluated jointly in a single lookahead forward pass with an isolated attention mask. Finally, RPS selects the token assignment for the pivot based on the following equation:

$$c^\star = \underset{c \in \mathcal{C}}{\arg\max} \left\{ -\frac{1}{|\mathcal{M}| - 1} \sum_{i \in \mathcal{M} \setminus \{i^\star\}} H(p_i^c) + \lambda \log p_{\text{anchor}}(c) \right\}, |\mathcal{M}| > 1 \tag{4}$$

where p_i^c denotes the predictive distribution at position i under the branch assigning c at the pivot, $p_{\text{anchor}}(c)$ denotes the anchor branch's probability for c at the pivot, and is set to $P_{\text{anchor}}^{\max}$ when $c =$ [MASK], $\lambda \geq 0$ is the plausibility weight balancing current plausibility against future benefit. When $|\mathcal{M}| = 1$, RPS bypasses pivot search and greedily commits the sole remaining masked position.

In Eq. (4), the first addition term inside the braces is the mean entropy over the remaining masked positions of each branch, reflecting how much entropy the pivot assignment reduces. Specifically,

lower is better, signaling a stronger ripple effect for more parallel commits in subsequent steps. The second addition term inside the braces acts as plausibility regularization to avoid trivial tokens—for instance, a premature end-of-sequence prediction may suppress downstream entropy simply by collapsing future uncertainty. Moreover, as the anchor branch keeps the pivot masked, $p_{\text{anchor}}(c)$ is evaluated without conditioning on the assignment, providing a useful independent quality signal.

Integration with standard decoding. At each decoding step, the standard commit rule first produces a commit set $\mathcal{S}$ and assigns greedy predictions to these positions. RPS then performs its pivot search on the remaining masked positions $\mathcal{M}$: it selects the pivot, constructs the anchor and candidate branches, and evaluates them jointly in a single forward pass using the branch-isolating attention mask in Fig. 2. Each branch is isolated from all others so that it faithfully simulates what the model would predict if only that particular token were assigned at the pivot. To avoid redundant computation, the predictive distributions from the selected branch are carried over to the next decoding step. The full per-step procedure is summarized in Appendix A.

4.2 Analysis of the Lookahead Objective

We next analyze the lookahead objective in Eq. (4). This analysis does not attempt to derive the empirically motivated pivot rule; instead, it characterizes how the two scoring terms relate to the speed–quality trade-off after a pivot has been selected. For brevity, write the downstream-position count $n = |\mathcal{M}| - 1$ and the mean downstream entropy $\bar{H}_c = \frac{1}{n}\sum_{i\in\mathcal{M}\setminus\{i^\star\}} H(p_i^c)$ for branch c.

Proposition 1 (Entropy-certified parallelism)**.** *Assume $n > 0$. For a confidence threshold $\tau \in [1/2, 1)$, define the eligible-commit count $N_\tau(c) = \sum_{i\in\mathcal{M}\setminus\{i^\star\}} \mathbb{I}[\max_v p_i^c(v) \geq \tau]$. With binary entropy $h(\tau) = -\tau\log\tau - (1-\tau)\log(1-\tau)$,*

$$N_\tau(c) \geq \max\left\{0,\ n - \left\lfloor \frac{n\bar{H}_c}{h(\tau)} \right\rfloor \right\}. \tag{5}$$

$N_\tau(c)$ counts the positions eligible for commitment by a confidence-aware decoder in the next step. Proposition 1 establishes that reducing the mean downstream entropy $\bar{H}_c$ monotonically tightens a certified lower bound on this number. The future-benefit term in Eq. (4) therefore has a direct speed interpretation rather than serving only as a generic uncertainty heuristic. The guarantee is deliberately conservative: it lower-bounds one-step commit opportunities but neither predicts the exact number of commits nor directly bounds end-to-end NFE.

Proposition 2 (Plausibility-adjusted selection margin)**.** *Let a_c denote the effective anchor plausibility used by the scoring rule, where $a_c = p_{anchor}(c)$ if $c \neq [\texttt{MASK}]$, otherwise $a_c = \max_v p_{anchor}(v)$ when $c = [\texttt{MASK}]$. Maximizing Eq. (4) is equivalent to*

$$c^\star = \arg\min_{c\in\mathcal{C}} \left\{ \bar{H}_c - \lambda \log a_c \right\}. \tag{6}$$

Moreover, for any $c, d \in \mathcal{C}$, candidate c scores at least as high as d if and only if

$$\bar{H}_d - \bar{H}_c \geq \lambda \log \frac{a_d}{a_c}. \tag{7}$$

Eq. (6) is a Lagrangian relaxation of minimizing downstream entropy under a candidate-surprisal budget. Proposition 2 makes the resulting safeguard explicit: relative to a more plausible candidate, a less plausible candidate must compensate for its plausibility deficit with a proportionally larger entropy reduction. When $d = [\texttt{MASK}]$, the same margin governs whether RPS commits or abstains. Together, Propositions 1 and 2 separate the two roles of the scoring function: the entropy term promotes certifiable next-step parallelism, while the anchor term controls the evidence required to take that acceleration opportunity. Appendix B provides both proofs.

4.3 Discussion

As compared in Fig. 1 (left), although RPS shares the high-level goal of accelerating dLLM decoding with lookahead evaluation, recent lookahead methods (Xu et al., 2025; Fu et al., 2025) mainly refine *where* to commit: LoPA searches high-confidence residual positions, whereas ETE targets positions

near a prescribed confidence level. RPS instead selects mid-entropy pivots via truncated entropy. More importantly, unlike LoPA and ETE, which retain greedy top-1 assignment, RPS also refines *what* to commit by lookahead evaluation over plausible token candidates, motivated by our finding that the correct token is often non-top-1 where the ripple effect is strongest.

5 EXPERIMENTS

Setup. We evaluate two representative dLLM families: LLaDA family, which is trained from scratch, and Dream family, which is adapted from an autoregressive language model. Our evaluation covers mathematical reasoning (GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2024)) and code generation (HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021)). We compare RPS with the standard one-token-per-step **Default** decoder and five parallel decoding or sampling baselines: **Confidence** (Wu et al., 2025), **KLASS** (Kim et al., 2025), **EB-Sampler** (Ben-Hamu et al., 2025), **WINO** (Hong et al., 2025), and **LoPA** (Xu et al., 2025).[1] For evaluation, we use lm-evaluation-harness (Gao et al., 2024) with its standard task configurations: 5-shot for GSM8K, 4-shot for MATH500, 0-shot for HumanEval, and 3-shot for MBPP. Unless stated otherwise, the generation length is 256 and the block length is 32. We measure generation quality by accuracy and efficiency by both the number of function evaluations (NFE), which captures sequential model evaluations, and tokens per second (TPS), which captures end-to-end throughput. Speedups are computed relative to Default under the same model, task, and generation length.

Implementation details. We build on the Fast-dLLM inference stack (Wu et al., 2025) and evaluate LLaDA-8B-Instruct, Dream-v0-Instruct-7B, and LLaDA-1.5, for 12 model–benchmark configurations in total. For brevity, we refer to LLaDA-8B-Instruct and Dream-v0-Instruct-7B as LLaDA and Dream, respectively, throughout this section. We report the main-text results on LLaDA and Dream, with additional results on LLaDA-1.5 provided in Appendix D. Unless otherwise noted, RPS uses $\tau = 0.9$, $k_{\max} = 10$, reachability ratio $r = 0.1$, and probability-mass threshold $\tau_{\text{pivot}} = 0.9$ for LLaDA and 0.95 for Dream. We select the plausibility weight λ from the interval $[0.1, 0.5]$ identified in §5.2. Appendix C reports baseline configurations and hardware details.

5.1 MAIN RESULTS

Comparison with parallel decoding baselines. We compare RPS with the default one-token-per-step decoder and several state-of-the-art parallel decoding methods across two model families and four benchmarks, with results reported in Table 1. RPS remains in the highest-throughput regime among parallel decoders, delivering 4.24–9.80× TPS speedup over Default while largely preserving generation quality. In several settings, acceleration even comes with improved accuracy: on LLaDA, RPS exceeds Default by 2.2% on MBPP, and on HumanEval it improves accuracy by 1.22 % for both LLaDA and Dream. These results demonstrate that jointly selecting a mid-entropy pivot and evaluating plausible token assignments with an explicit plausibility safeguard enables aggressive parallel commitment while preserving or improving generation quality in most settings. Among the parallel baselines, LoPA is closest to RPS in decoding speed, as it likewise uses lookahead to guide early commitment. However, LoPA consistently loses accuracy relative to Default on the code-generation benchmarks. The gap is especially clear on HumanEval, where RPS outperforms LoPA by 4.27 % on LLaDA and 5.49 % on Dream at comparable throughput. Section 5.3 provides a dedicated failure-mode analysis of this behavior.

Robustness to generation length. The preceding experiments establish a strong quality–efficiency trade-off for RPS at the default generation length of 256. We next examine whether this advantage persists across different generation lengths. As shown in Table 2, RPS maintains the strongest quality–efficiency trade-off among the compared baselines across all evaluated lengths. A model-specific exception is HumanEval at $L = 128$, where all LLaDA decoders exhibit lower accuracy than at longer lengths, while Dream does not show the same degradation. Inspection of the generated programs indicates that the LLaDA outputs are generally syntactically complete but often implement overly simplified or incomplete logic, suggesting that the restricted budget affects solution formation rather than merely truncating the code. Under this constraint, LoPA's aggressive early

[1] ETE is not included because no official implementation was publicly available at the time of evaluation.

Table 1: Results on LLaDA and Dream. **Bold** indicates the best result.

Benchmark	Method	LLaDA-8B-Instruct					Dream-v0-Instruct-7B				
		Acc	NFE	NFE Sp.	TPS	TPS Sp.	Acc	NFE	NFE Sp.	TPS	TPS Sp.
GSM8K (5-shot)	Default	79.00	256.0	1.00×	4.02	1.00×	78.70	256.0	1.00×	5.14	1.00×
	Confidence	**79.45**	77.80	3.29×	13.21	3.29×	77.33	66.05	3.88×	19.16	3.73×
	KLASS	79.30	131.93	1.94×	7.88	1.96×	**79.38**	120.87	2.12×	10.47	2.04×
	EB-Sampler	**79.45**	87.73	2.92×	11.88	2.96×	79.30	108.15	2.37×	11.86	2.31×
	WINO	78.24	57.37	4.46×	17.70	4.40×	76.12	61.90	4.14×	19.43	3.78×
	LoPA	78.70	41.95	6.10×	22.98	5.68×	75.66	**34.62**	**7.40**×	**31.31**	**6.09**×
	RPS	79.15	**35.73**	**7.17**×	**26.66**	**6.63**×	78.62	41.74	6.13×	28.21	5.49×
HumanEval (0-shot)	Default	41.46	256.0	1.00×	14.24	1.00×	57.32	256.0	1.00×	9.13	1.00×
	Confidence	42.07	78.83	3.25×	45.93	3.23×	58.54	82.98	3.08×	28.51	3.12×
	KLASS	41.46	128.82	1.99×	28.75	2.02×	**60.98**	125.45	2.04×	19.26	2.11×
	EB-Sampler	41.46	91.46	2.80×	40.16	2.82×	60.37	110.01	2.33×	21.98	2.41×
	WINO	40.24	73.67	3.47×	47.83	3.36×	57.93	79.37	3.23×	26.68	2.92×
	LoPA	38.41	40.25	6.36×	76.42	5.37×	53.05	**40.36**	**6.34**×	**43.02**	**4.71**×
	RPS	**42.68**	**37.22**	**6.88**×	**82.98**	**5.83**×	58.54	48.33	5.30×	41.32	4.53×
MATH500 (4-shot)	Default	38.40	256.0	1.00×	6.02	1.00×	44.60	256.0	1.00×	7.84	1.00×
	Confidence	38.60	98.85	2.59×	15.47	2.57×	**45.60**	93.90	2.73×	21.17	2.70×
	KLASS	38.20	157.05	1.63×	9.92	1.65×	44.20	142.56	1.80×	14.28	1.82×
	EB-Sampler	38.60	210.50	1.22×	7.45	1.24×	44.20	100.29	2.55×	20.36	2.60×
	WINO	**38.80**	73.15	3.50×	20.75	3.45×	44.80	82.60	3.10×	22.31	2.84×
	LoPA	38.20	70.36	3.64×	21.21	3.53×	44.20	64.39	3.98×	27.62	3.52×
	RPS	38.20	**48.32**	**5.30**×	**29.09**	**4.83**×	44.80	**55.61**	**4.60**×	**33.27**	**4.24**×
MBPP (3-shot)	Default	30.60	256.0	1.00×	3.99	1.00×	**58.80**	256.0	1.00×	2.40	1.00×
	Confidence	30.60	69.82	3.67×	14.72	3.69×	56.20	39.61	6.46×	14.71	6.13×
	KLASS	30.60	111.32	2.30×	9.41	2.36×	55.20	61.58	4.16×	8.87	3.70×
	EB-Sampler	30.80	79.64	3.22×	13.07	3.28×	55.00	38.34	6.68×	14.17	5.90×
	WINO	30.00	67.63	3.79×	15.06	3.77×	56.40	42.03	6.09×	13.11	5.46×
	LoPA	29.20	37.83	6.77×	23.25	5.83×	55.60	25.19	10.16×	21.75	9.06×
	RPS	**32.80**	**29.92**	**8.56**×	**25.10**	**6.29**×	56.80	**23.81**	**10.76**×	**23.52**	**9.80**×

Table 2: Generation-length robustness on LLaDA and Dream.

Model	Benchmark	Method	$L = 128$			$L = 256$			$L = 512$		
			Acc	NFE Sp.	TPS Sp.	Acc	NFE Sp.	TPS Sp.	Acc	NFE Sp.	TPS Sp.
LLaDA-8B-Instruct	HumanEval (0-shot)	Default	**30.49**	1.00×	1.00×	41.46	1.00×	1.00×	**43.29**	1.00×	1.00×
		Confidence	29.27	3.59×	3.49×	42.07	3.25×	3.23×	41.46	3.15×	3.15×
		LoPA	22.56	4.79×	4.39×	38.41	6.36×	5.37×	40.85	6.32×	5.19×
		RPS	28.66	**6.49**×	**5.26**×	**42.68**	**6.88**×	**5.83**×	42.07	**6.48**×	**5.45**×
	MATH500 (4-shot)	Default	34.40	1.00×	1.00×	38.40	1.00×	1.00×	42.20	1.00×	1.00×
		Confidence	34.20	2.13×	2.13×	**38.60**	2.59×	2.57×	**42.40**	3.42×	3.38×
		LoPA	34.40	3.00×	2.91×	38.20	3.64×	3.53×	40.40	6.84×	6.21×
		RPS	**35.60**	**4.22**×	**3.86**×	38.20	**5.30**×	**4.83**×	**42.40**	**6.97**×	**6.36**×
Dream-v0-Instruct-7B	HumanEval (0-shot)	Default	58.54	1.00×	1.00×	57.32	1.00×	1.00×	53.66	1.00×	1.00×
		Confidence	**59.76**	2.71×	2.56×	**58.54**	3.08×	3.12×	57.32	5.72×	5.63×
		LoPA	53.66	4.14×	3.30×	53.05	**6.34**×	**4.71**×	53.66	9.43×	6.90×
		RPS	58.54	**4.20**×	**3.73**×	**58.54**	5.30×	4.53×	**58.54**	**9.77**×	**8.19**×
	MATH500 (4-shot)	Default	42.00	1.00×	1.00×	44.60	1.00×	1.00×	45.60	1.00×	1.00×
		Confidence	**42.80**	2.08×	2.09×	**45.60**	2.73×	2.70×	47.80	4.31×	4.30×
		LoPA	40.60	3.02×	2.65×	44.20	3.98×	3.52×	46.20	6.54×	5.79×
		RPS	41.40	**3.39**×	**2.99**×	44.80	**4.60**×	**4.24**×	**49.20**	**7.15**×	**6.62**×

commitments further amplify the quality loss, reducing accuracy from 30.49 for Default to 22.56. In contrast, RPS retains 28.66 accuracy while achieving 6.49× NFE and 5.26× TPS speedups, limiting the additional degradation caused by parallel decoding.

5.2 Hyperparameter Ablations

Plausibility-weight sensitivity. We isolate the contribution of the plausibility safeguard by sweeping $\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ on LLaDA GSM8K, as shown in Fig. 3 (left). Enabling the

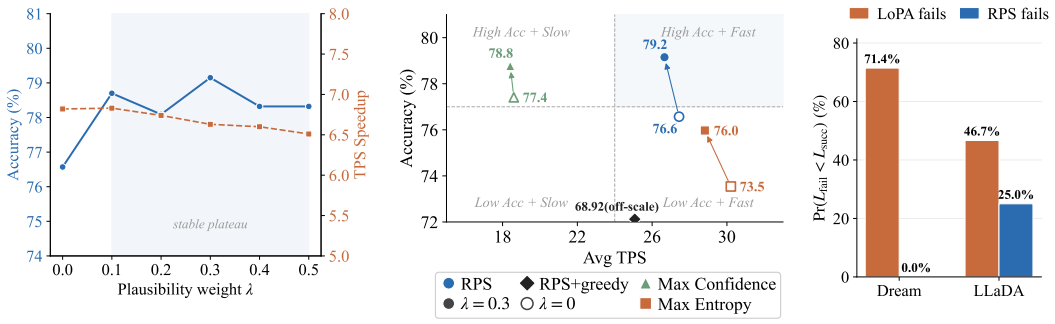

Figure 3: **Left:** Accuracy and TPS speedup of RPS on GSM8K with LLaDA under different plausibility weights λ. **Middle:** Accuracy and TPS trade-off of different pivot-selection and scoring strategies on GSM8K with LLaDA. **Right:** Failure-length comparison on HumanEval for Dream and LLaDA. For cases where exactly one of LoPA and RPS succeeds, we report the fraction for which the failing completion contains fewer non-empty source lines than the successful completion.

safeguard with any positive λ substantially improves accuracy over entropy-only scoring ($\lambda = 0$); in particular, $\lambda = 0.1$ yields a 2.1% gain. This consistent improvement demonstrates that branch-external plausibility is necessary to prevent lookahead from favoring decisive but incorrect token assignments. Meanwhile, accuracy varies by only 1.1% across $\lambda \in [0.1, 0.5]$, with little change in TPS speedup. The scoring function is therefore robust once the safeguard is enabled and does not require extensive scenario-specific hyperparameter tuning.

Search-space hyperparameters. We further analyze the three hyperparameters governing the pivot-search space (Table 3). The candidate budget $k_{\max}$ caps the support: overly large values admit weak candidates and hurt accuracy, while overly small values limit useful alternatives and slow decoding. The reachability ratio r further prunes tokens relative to the top-1 probability and is robust over a broad range, though aggressive pruning may remove plausible assignments. The threshold τ_{pivot} controls pivot reliability: low values admit overly diffuse positions, whereas high values make selection too conservative and reduce speed. Since these parameters impose structural search constraints rather than task-specific preferences, we fix $k_{\max}$ and r globally and use a single τ_{pivot} across all tasks within each model family, without any tuning.

Table 3: Hyperparameter sensitivity on LLaDA GSM8K. One parameter is varied at a time; **bold** marks the setting used in the main experiments.

Hyperparameter	Value	Acc (%)	Avg. NFE	TPS	Avg. cand.
$k_{\max}$	5	78.70	42.09	23.24	3.00
	10	**79.15**	**35.73**	**26.66**	**3.87**
	20	74.91	32.02	28.30	5.16
r	0.05	78.62	35.15	26.31	4.77
	0.10	**79.15**	**35.73**	**26.66**	**3.87**
	0.20	78.70	37.51	26.06	3.00
	0.30	77.41	39.93	24.90	2.55
τ_{pivot}	0.60	78.17	31.12	29.26	5.05
	0.70	77.33	31.38	28.93	5.02
	0.80	78.62	32.26	28.54	4.70
	0.90	**79.15**	**35.73**	**26.66**	**3.87**
	0.95	77.86	40.62	23.90	3.20

5.3 Further Analysis

Pivot strategy and scoring ablation. Fig. 3 (middle) disentangles the effects of pivot selection and token scoring. For pivot selection, with scoring fixed, Max Confidence is accurate but slow because it selects nearly determined positions, whereas Max Entropy is faster but less accurate because its pivots are overly uncertain. Between these two extremes, RPS targets tractable mid-entropy positions and achieves the strongest quality–speed trade-off. Turning to token scoring, we fix the RPS pivot rule and replace lookahead scoring with greedy top-1 assignment, which reduces accuracy to 68.92% at comparable speed. This result indicates that top-1 predictions are unreliable in the mid-entropy regime. Finally, comparing $\lambda = 0$ with $\lambda = 0.3$ shows that plausibility-aware scoring filters implausible assignments with little effect on throughput.

Failure-mode analysis on HumanEval. To better understand why RPS attains higher accuracy than LoPA on zero-shot HumanEval, we examine the programs generated by the two methods. We focus on cases where exactly one method succeeds and compare the number of non-empty source lines in the failing and successful completions. Fig. 3 (right) reveals a clear asymmetry: a shorter failing program is substantially more common when LoPA fails than when RPS fails, suggesting that

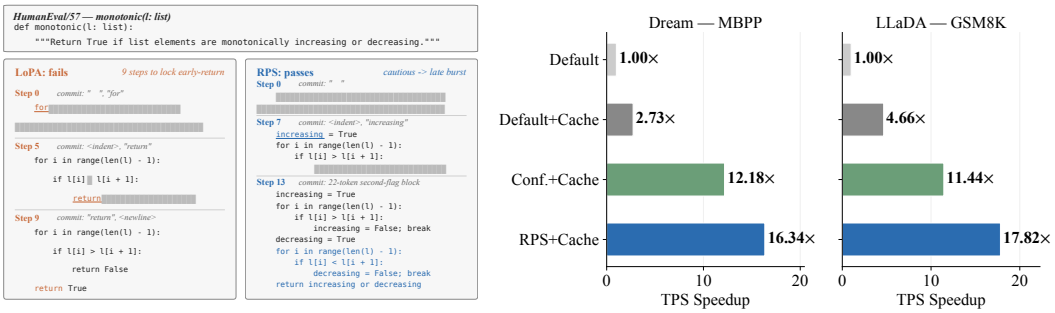

Figure 4: **Left:** Decoding snapshots of LoPA and RPS on HumanEval/57 (`monotonic`). Coloured tokens denote newly committed tokens, underlined tokens denote lookahead commits, and "■" denotes still-masked positions. **Right:** TPS speedup relative to Default and Confidence with and without Fast-dLLM prefix caching, evaluated on MBPP with Dream and GSM8K with LLaDA.

LoPA is more prone to terminating a plausible-looking solution before completing the required logic. Fig. 4 (left) illustrates this behavior on HumanEval/57 (monotonic). LoPA commits `return` early in decoding, closing the program before it accounts for monotonically decreasing inputs. RPS makes more conservative early commitments and later completes the complementary state logic, thereby avoiding this premature termination. This example reflects a broader failure mode of confidence-driven lookahead: LoPA can favor a wrong-but-decisive token because it immediately collapses downstream uncertainty. RPS instead combines cautious pivot selection with a plausibility safeguard that screens the quality of an early commitment. This distinction is especially consequential in code generation, where control-flow tokens such as `return` and `break` directly alter the execution path. An incorrect commitment can therefore produce a program that is syntactically valid and executable, yet globally incomplete or logically wrong—an error that subsequent decoding cannot easily repair.

Speedup decomposition. As shown in Fig. 2, our packed lookahead evaluates all candidate branches jointly with the normal sequence in a single forward pass. Although this increases per-step computation, Table 4 shows that a lookahead pass is only 15% slower than a normal pass, while reducing the average number of forward passes from 77.80 under confidence decoding to 35.73 with RPS. Thus, a modest per-step overhead yields a substantial reduction in decoding iterations.

Table 4: Average forward-pass breakdown per sample for RPS on LLaDA GSM8K.

Forward type	Avg. count	Avg. latency	Time %
Normal	12.3	219 ms	31.4%
Lookahead	23.4	252 ms	68.6%
Total	**35.7**	**241 ms**	100%

Compatibility with KV caching. We combine RPS with Fast-dLLM prefix caching to test whether iteration-level and per-forward optimizations are complementary. As in Fig. 4 (right), the combination achieves the highest throughput in both settings, with up to 17.82× TPS speedup and less than 0.5% accuracy change. This complementarity is natural: RPS reduces decoding iterations via proactive commitment, while KV caching lowers the attention cost per iteration.

6 Conclusion

We presented Ripple-Pivot Search (RPS), a training-free parallel decoding method for diffusion language models motivated by the ripple effect. Proactively committing a pivot position in the mid-entropy regime can substantially reduce uncertainty across the remaining masked positions, creating more opportunities for parallel commitment in subsequent decoding steps. RPS exploits this effect by addressing both *where to commit* and *what to commit*. It seeks mid-entropy pivots with high potential for downstream uncertainty reduction and determines their token assignments through lookahead evaluation over plausible candidates. Across three dLLMs and four reasoning and code-generation benchmarks, RPS achieves 4–10× wall-clock speedup over the default decoder while largely preserving generation quality, and improves accuracy over the previous lookahead baseline by up to 5.49% while delivering higher throughput in most settings. When combined with KV caching, RPS further achieves up to 18× wall-clock speedup over the default decoder. Together, these results demonstrate the effectiveness of jointly searching over commitment positions and token assignments for efficient dLLM decoding.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript to improve readability, including grammar, wording, and overall language fluency. We did not use generative AI tools for any of the research subtasks for which disclosure is required, and the remaining required-disclosure subtasks are not applicable to this work. All AI-assisted edits were reviewed and verified by two authors to ensure that they preserved the intended meaning and did not introduce unsupported claims or factual errors. The authors made the final decisions regarding the manuscript's content and wording. We take responsibility for the final content of this work, including all text, claims, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies an inference-time decoding method for existing diffusion language models and evaluates it on established mathematical reasoning and code-generation benchmarks. It does not introduce a new dataset or change model training. Faster inference may lower the cost of generating misleading or harmful content, including unsafe code. Our experiments assess generation accuracy and efficiency; they do not assess safety, fairness, or privacy. The reported speedups should therefore not be interpreted as improvements on those dimensions. Deployments of RPS should retain appropriate model- and application-level safeguards and evaluate output risks in their intended setting.

REPRODUCIBILITY STATEMENT

To facilitate independent reproduction of RPS and its evaluation, we document both the decoding algorithm and the experimental protocol in the main text and appendices. §4 defines the decoding procedure and lookahead objective, complemented by complete pseudocode in Appendix A and proofs in Appendix B. §5 reports the evaluated models and benchmarks, few-shot settings, generation lengths, metrics, principal hyperparameters, results, and ablations; Appendix C details baseline configurations, parameter sweeps, and hardware, while Appendix D provides additional LLaDA-1.5 results. These materials provide the basis for reimplementing RPS and repeating the reported comparisons under the stated settings.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via adaptive parallel decoding. *CoRR*, abs/2509.25188, 2025.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b. *CoRR*, abs/2512.15745, 2025.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal,

Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé 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, Joshua 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. *CoRR*, abs/2107.03374, 2021.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 22058–22075. Association for Computational Linguistics, 2026.

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. *CoRR*, abs/2110.14168, 2021.

Hengyu Fu, Baihe Huang, Virginia Adams, Charles Wang, Venkat Srinivasan, and Jiantao Jiao. From bits to rounds: Parallel decoding with exploration for diffusion language models. *CoRR*, abs/2511.21103, 2025.

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

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective dllms. *CoRR*, abs/2507.18578, 2025.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: kl-guided fast inference in masked diffusion models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

Pengxiang Li, Dilxat Muhtar, Tianlong Chen, Lu Yin, and Shiwei Liu. Why diffusion language models struggle with truly parallel (non-autoregressive) decoding? *CoRR*, abs/2602.23225, 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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 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. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles. *CoRR*, abs/2506.10848, 2025.

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. *CoRR*, abs/2505.22618, 2025.

Shutong Wu and Jiawei Zhang. Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models. *CoRR*, abs/2510.00294, 2025.

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. *CoRR*, abs/2512.16229, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

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

Yushi Ye, Feng Hong, Huangjie Zheng, Xu Chen, Zhiyong Chen, Yanfeng Wang, and Jiangchao Yao. Rejection mixing: Fast semantic propagation of mask tokens for efficient DLLM inference. *CoRR*, abs/2602.22868, 2026.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Yiwen Hu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *Frontiers Comput. Sci.*, 20(12): 2012627, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 11425–11460. Association for Computational Linguistics, 2026a.

Kaiwen Zhu, Quansheng Zeng, Yuandong Pu, Shuo Cao, Xiaohui Li, Yi Xin, Qi Qin, Jiayang Li, Yu Qiao, Jinjin Gu, and Yihao Liu. Accelerating masked image generation by learning latent controlled dynamics. *CoRR*, abs/2602.23996, 2026b.

A RPS ALGORITHM PSEUDOCODE

Algorithm 1 RPS decoding

Require: prompt y; generation length L; block size B; hyperparameters $\tau_{\text{pivot}}, k_{\max}, r, \lambda, \tau$

$x \leftarrow$ `[MASK]`^{L} ▷ Initialize response with all masks
for $b = 1, \dots, L/B$ **do** ▷ Semi-autoregressive block loop
  $\{p_i\} \leftarrow$ forward pass over $[y \| x]$ ▷ Block-opening forward
  $\mathcal{M} \leftarrow$ masked positions in block b
  while $\mathcal{M} \neq \emptyset$ **do**
    $\mathcal{S} \leftarrow \{i \in \mathcal{M} : P_i^{\max} \geq \tau\}$; if $\mathcal{S} = \emptyset$, set $\mathcal{S} \leftarrow \{\arg\max_{i \in \mathcal{M}} P_i^{\max}\}$ ▷ Standard commit
    Commit $x_i \leftarrow \arg\max_v p_i(v)$ for $i \in \mathcal{S}$; update $\mathcal{M} \leftarrow \mathcal{M} \setminus \mathcal{S}$
    if $\mathcal{M} = \emptyset$ **then break**
    end if
    $\mathcal{T}_i \leftarrow \text{top}_{k_{\max}}(p_i)$ and $\mu_i \leftarrow \sum_{v \in \mathcal{T}_i} p_i(v)$ for each $i \in \mathcal{M}$
    $\mathcal{M}_{\text{feas}} \leftarrow \{i \in \mathcal{M} : \mu_i \geq \tau_{\text{pivot}}\}$ ▷ Probability-mass constraint
    if $\mathcal{M}_{\text{feas}} = \emptyset$ **then**
      $\{p_i\} \leftarrow$ forward pass over $[y \| x]$; **continue**
    end if
    $i^\star \leftarrow \arg\max_{i \in \mathcal{M}_{\text{feas}}} \big[-\sum_{v \in \mathcal{T}_i} p_i(v) \log p_i(v) \big]$ ▷ Truncated-entropy maximization
    $\mathcal{C} \leftarrow \{c : p_{i^\star}(c)/P_{i^\star}^{\max} \geq r\}$, capped at $k_{\max}$ tokens; add `[MASK]`
    if $|\mathcal{C} \setminus \{\texttt{[MASK]}\}| \leq 1$ **then**
      $\{p_i\} \leftarrow$ forward pass over $[y \| x]$; **continue**
    end if
    $x_{\text{ahead}} \leftarrow [x \,\|\, x_b^{(c_1)} \| \,\cdots\, \| \, x_b^{(c_{|\mathcal{C}|})}]$; $\{p_i^c\} \leftarrow$ forward pass over $[y \| x_{\text{ahead}}]$ ▷ Lookahead forward
    $c^\star \leftarrow \arg\max_{c \in \mathcal{C}} \text{score}(c)$ via Eq. (4) ▷ Scoring and commit
    if $c^\star \neq$ `[MASK]` **then**
      Commit $c^\star$ at $i^\star$; $\mathcal{M} \leftarrow \mathcal{M} \setminus \{i^\star\}$; $\{p_i\} \leftarrow \{p_i^{(c^\star)}\}$ ▷ Reuse winner's logits
    else
      $\{p_i\} \leftarrow \{p_i^{(\texttt{[MASK]})}\}$ ▷ Reuse anchor's logits
    end if
  end while
end for
return x

B PROOFS FOR THE LOOKAHEAD OBJECTIVE

Proof of Proposition 1. For $i \in \mathcal{M} \setminus \{i^\star\}$, define

$$q_i = \max_v p_i^c(v), \qquad \mathcal{U}_\tau(c) = \{i \in \mathcal{M} \setminus \{i^\star\} : q_i < \tau\}, \qquad U_\tau(c) = |\mathcal{U}_\tau(c)|.$$

Fix $i \in \mathcal{U}_\tau(c)$, so $q_i < \tau < 1$. If $q_i \geq 1/2$, let $v_i^\star \in \arg\max_v p_i^c(v)$ and define $\widetilde{p}_i^c(v) = p_i^c(v)/(1 - q_i)$ for $v \neq v_i^\star$. The entropy decomposition gives

$$\begin{aligned} H(p_i^c) &= h(q_i) + (1 - q_i) H(\widetilde{p}_i^c) \\ &\geq h(q_i). \end{aligned}$$

If $q_i < 1/2$, then $p_i^c(v) \leq q_i$ for every v, and hence

$$\begin{aligned} H(p_i^c) &= \sum_v p_i^c(v) \log \frac{1}{p_i^c(v)} \\ &\geq \sum_v p_i^c(v) \log \frac{1}{q_i} = -\log q_i. \end{aligned}$$

Since $h(x)$ is non-increasing on $[1/2, 1]$ and $h(\tau) \leq \log 2$, for every $i \in \mathcal{U}_\tau(c)$,

$$H(p_i^c) \geq \begin{cases} h(q_i) \geq h(\tau), & q_i \in [1/2, \tau), \\ -\log q_i > \log 2 \geq h(\tau), & q_i < 1/2. \end{cases}$$

Summing this bound over $\mathcal{U}_\tau(c)$ yields

$$\begin{aligned} n\bar{H}_c = \sum_{i \in \mathcal{M} \setminus \{i^\star\}} H(p_i^c) &\geq \sum_{i \in \mathcal{U}_\tau(c)} H(p_i^c) \\ &\geq U_\tau(c) h(\tau), \end{aligned}$$

and therefore

$$U_\tau(c) \leq \min\left\{ n, \left\lfloor \frac{n\bar{H}_c}{h(\tau)} \right\rfloor \right\}.$$

Since $N_\tau(c) = n - U_\tau(c)$,

$$\begin{aligned} N_\tau(c) &= n - U_\tau(c) \\ &\geq \max\left\{ 0, n - \left\lfloor \frac{n\bar{H}_c}{h(\tau)} \right\rfloor \right\}, \end{aligned}$$

which is Eq. (5). □

Proof of Proposition 2. Let

$$S(c) = -\bar{H}_c + \lambda \log a_c,$$

Then

$$\begin{aligned} \arg\max_{c \in \mathcal{C}} S(c) &= \arg\max_{c \in \mathcal{C}} \{-\bar{H}_c + \lambda \log a_c\} \\ &= \arg\min_{c \in \mathcal{C}} \{\bar{H}_c - \lambda \log a_c\}, \end{aligned}$$

which proves Eq. (6). For any $c, d \in \mathcal{C}$,

$$\begin{aligned} S(c) \geq S(d) &\iff -\bar{H}_c + \lambda \log a_c \geq -\bar{H}_d + \lambda \log a_d \\ &\iff \bar{H}_d - \bar{H}_c \geq \lambda(\log a_d - \log a_c) \\ &\iff \bar{H}_d - \bar{H}_c \geq \lambda \log \frac{a_d}{a_c}, \end{aligned}$$

which proves Eq. (7) in both directions. □

C Implementation Details

Baselines. Default uses highest-confidence unmasking with one token per step. Confidence follows Fast-dLLM with threshold $\tau = 0.9$. KLASS, EB-Sampler, and WINO follow the settings in their original papers and report the best result from the prescribed sweep ranges: for KLASS, we use confidence threshold $\tau = 0.9$ and select the KL threshold ϵ_{KL} from $\{0.015, 0.01, 0.005, 0.001\}$; for EB-Sampler, we sweep $\gamma \in \{0.1, 0.01, 0.001\}$ with the confidence-based error proxy; for WINO, we sweep the drafting threshold $\tau_1 \in \{0.5, 0.6, 0.7, 0.8\}$ while fixing the verification threshold $\tau_2 = 0.9$. For LoPA, we reimplement its core decoding algorithm without the LoPA-Dist distributed inference system or its system-level optimizations, so that all methods run on the same single-device inference stack.

Hardware. All experiments are conducted on a node equipped with 4× NVIDIA A100-SXM4-80GB GPUs. Inference is performed in a data-parallel manner across all four GPUs. Throughput (TPS) is normalized to an equivalent single-GPU basis by dividing the aggregate four-GPU throughput by four.

D LLaDA-1.5 Results

Table 5: **LLaDA-1.5.** Performance and inference speedup comparison across 4 benchmarks.

Benchmark	Method	Acc (%)	Avg NFE	NFE Speedup	Avg TPS	TPS Speedup
GSM8K (5-shot)	Default	80.52	256.0	1.00×	3.80	1.00×
	Confidence	81.05	74.62	3.43×	12.96	3.41×
	WINO	80.29	57.48	4.46×	16.64	4.38×
	LoPA	80.21	37.61	6.81×	23.33	6.14×
	RPS	80.52	33.73	7.59×	25.99	6.84×
HumanEval (0-shot)	Default	42.07	256.0	1.00×	5.03	1.00×
	Confidence	41.46	100.92	2.54×	12.90	2.56×
	WINO	42.68	84.76	3.02×	14.08	2.80×
	LoPA	40.24	47.46	5.39×	24.30	4.83×
	RPS	41.46	42.31	6.05×	27.21	5.41×
MATH500 (4-shot)	Default	39.80	256.0	1.00×	5.32	1.00×
	Confidence	39.60	95.11	2.69×	14.26	2.68×
	WINO	40.20	75.30	3.40×	17.83	3.35×
	LoPA	38.20	57.70	4.44×	22.80	4.29×
	RPS	39.60	49.37	5.19×	25.91	4.87×
MBPP (3-shot)	Default	38.40	256.0	1.00×	1.46	1.00×
	Confidence	38.40	42.68	6.00×	8.37	5.73×
	WINO	39.00	46.48	5.51×	7.91	5.42×
	LoPA	40.20	24.66	10.38×	14.38	9.85×
	RPS	44.20	23.50	10.89×	14.40	9.86×

E Limitations

Limitations. While RPS demonstrates strong acceleration gains while largely preserving generation quality, it still has several limitations. First, our characterization of the ripple effect is empirical and qualitative rather than theoretically derived. In particular, the motivating analysis in §1 is conducted on GSM8K and is intended to illustrate the phenomenon, not to establish that the precise location of the cascade peak quantitatively generalizes across models or tasks. Second, when combined with prefix cache acceleration (§5.3), accuracy degradation may arise from the cache approximation itself. This effect is not specific to RPS and is also observed for other parallel decoding methods, but mitigating it remains outside the scope of this work.