FASTGUIDE: ACCELERATING REWARD GUIDANCE FOR DIFFUSION LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient-based reward guidance provides a flexible way to use downstream reward models to control masked diffusion language models at inference time. However, its computational cost remains high as each decoding iteration incurs expensive diffusion model forward passes and reward model backpropagation steps. To address this, we introduce FastGuide, an adaptive hybrid of parallel and autoregressive decoding to accelerate reward guidance for diffusion language models. In analogy to parallel decoding, FastGuide amortizes the cost of reward model backpropagation by computing guidance once per decoding step and reusing it to generate multiple tokens. Within each decoding step, FastGuide makes diffusion forward passes autoregressive by unmasking tokens one at a time while efficiently recomputing token distributions after each unmasking by utilizing KV caching techniques and sparse recomputation of attention. Lastly, to adapt hybrid decoding to the model's confidence, FastGuide defers any token that the model is unconfident about under its recomputed distribution. Experiments on three reward benchmarks demonstrate that FastGuide is up to $4.4\times$ faster than sequential reward-guided decoding while retaining similar generation quality.

1 INTRODUCTION

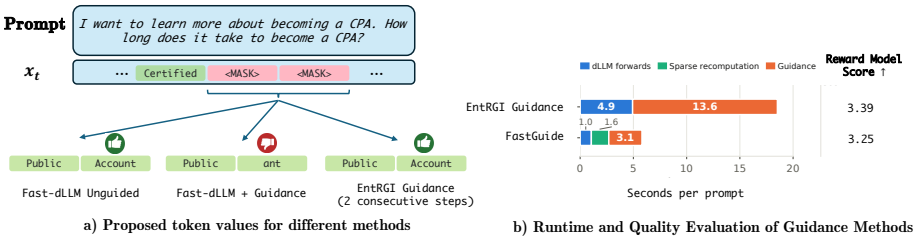
Diffusion Large Language Models (dLLMs) have emerged as a promising alternative to traditional autoregressive models and have been praised for their reasoning power and generation speed (Ye et al., 2025a; Nie et al., 2026). A popular class of dLLMs is discrete masked dLLMs (Austin et al., 2021; Sahoo et al., 2024), which are trained on partially masked token sequences to predict marginal distributions over the vocabulary for each masked token. At inference time, dLLM generation starts from a fully masked sequence of fixed length and decoding algorithms iteratively decide a) where to unmask, and b) which token values to place at the selected positions. For instance, a common strategy is to compute a confidence score for each token position, pick the most confident token, and sample from the corresponding marginal distribution of that token (Ye et al., 2025b).

As these models begin to be deployed in practical settings (Inception Labs, 2025; Google DeepMind, 2025), an important area of research is ensuring alignment of the dLLM outputs to desired objectives, e.g., instruction following, truthfulness, or safety (Wen et al., 2026; Xiong et al., 2026). A recent line of work has explored training-free gradient-based guidance algorithms for dLLM alignment using a differentiable reward model (Murata et al., 2024; Tejaswi et al., 2026). For these approaches, at each diffusion timestep t with partially masked sequence $\boldsymbol{x}_t$ and unmasked tokens $\boldsymbol{x}_t^{\mathrm{u}}$, the dLLM logits at each token position i are steered via the updates

$$\ell_t^{\text{guided}}(x_t^i \mid \boldsymbol{x}_t^{\mathrm{u}}) = \underbrace{\ell_t^{\text{unguided}}(x_t^i \mid \boldsymbol{x}_t^{\mathrm{u}})}_{\text{base dLLM logits}} + \underbrace{\boldsymbol{r}_t(x_t^i \mid \boldsymbol{x}_t^{\mathrm{u}})}_{\text{guidance vector}}. \tag{1}$$

Above, ℓ_t^{unguided} denotes the per-token dLLM logits conditioned on the unmasked tokens $\boldsymbol{x}_t^{\mathrm{u}}$, and $\boldsymbol{r}_t$ denotes the corresponding per-token guidance vector. The guidance vector is estimated by cheaply constructing a fully unmasked sequence, evaluating that sequence with a reward model, and computing gradients of the reward model with respect to its input to steer the dLLM logits. Although gradient-based guidance can outperform alternatives such as Best-Of-N sampling (Tejaswi et al., 2026), the sampling speed of these approaches is slow, as shown in the top row of Figure 1b. This is due to the dependency structure of Equation 1. At each diffusion timestep, when one token is

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.



a) Proposed token values for different methods

b) Runtime and Quality Evaluation of Guidance Methods

Figure 1: Left: Reward guidance (Tejaswi et al., 2026) can increase incompatibilities across token values proposed during parallel decoding. Above, `CPA` should expand to `Certified Public Accountant`. While unguided generation with parallel decoding method Fast-dLLM (Wu et al., 2025) produces the correct token prefix for `CPA`, Fast-dLLM with guidance fails. Right: Our method is much faster than sequential guided generation while retaining similar generation quality.

unmasked, both the dLLM logits and the guidance vector change because the conditioning variable changes. Recomputing the dLLM logits requires a forward pass through the dLLM, which involves bidirectional attention over the sequence, and recomputing the guidance vector requires a forward and backward pass through a large reward model (e.g., a Skywork-Llama-8B model (Liu et al., 2026)). Both of these steps can be large inference-time bottlenecks.

Parallel decoding is the dominant strategy to accelerate dLLM inference, so a natural idea is to follow existing work and unmask multiple tokens in each diffusion timestep (Ben-Hamu et al., 2025; Kim et al., 2026a; Wu et al., 2025). However, existing work is developed for unguided generation, so it does not establish when guidance can be reused, which is the main inference-time bottleneck (Figure 1b). Further, it is unclear which tokens remain safe to unmask together once guidance has altered the token distributions. This leads to our main research question:

> *Can we accelerate reward-guided dLLM inference while preserving its generation quality?*

In this paper, we present, to the best of our knowledge, the first study of accelerating gradient-based reward guidance for dLLMs. Our main contributions are:

1. We show that, surprisingly, reusing the guidance vector to unmask a group of k tokens is benign. However, reusing the dLLM logits degrades generation quality. Specifically, we observe that (**O1**) at masked positions with high confidence scores, the token distribution after guidance remains similar when guidance from an earlier diffusion timestep is reused. Conversely, (**O2**) guidance can increase incompatibilities between co-proposed token values compared to unguided decoding (Figure 1a), preventing reuse of the dLLM logits.
2. We introduce FastGuide, a training-free, hybrid, and adaptive acceleration strategy for reward-guided dLLM decoding. Motivated by **O1**, similar to parallel decoding, we select at each iteration a group of candidate token positions with high confidence scores and compute guidance once for the group. To address **O2**, we then autoregressively unmask candidates within this group, recomputing the dLLM logits after each unmasking so that subsequent token choices are conditioned on prior unmaskings. To make these updates efficient, we use sparse recomputation with KV caching (Ma et al., 2025), updating only a small subset of dLLM token representations. Finally, token candidates whose recomputed distributions have low confidence scores are deferred to a later iteration, adaptively balancing the parallel and autoregressive decoding steps.
3. Our guidance reuse principle allows us to extend five existing parallel decoders to the guided setting. Across three reward benchmarks and two dLLMs, FastGuide generates responses up to $4.4\times$ faster than sequential guided decoding while retaining similar reward model scores.

2 Background and Related Work

In this section, we review masked dLLM inference, parallel decoding strategies, and reward guidance methods. Given a prompt $\boldsymbol{p}$, the masked dLLM generates a fixed-length response $\boldsymbol{x}_0$ of length L where each token comes from a vocabulary $\mathbb{V}$, where $\mathbb{V}$ includes a `<MASK>` token.

Masked dLLM inference. A masked dLLM generates $\boldsymbol{x}_0$ by iteratively denoising a partially masked state $\boldsymbol{x}_t \in \mathbb{V}^L$ (Austin et al., 2021; Sahoo et al., 2024). We will refer to t as the diffusion timestep, ranging from T to 0, where $\boldsymbol{x}_T$ is a sequence with all masked tokens. At every diffusion timestep t, we denote the set of masked and unmasked positions by

$$\mathbb{M}_t = \{i \in [L] : x_t^i = \texttt{<MASK>}\}, \qquad \mathbb{U}_t = [L] \setminus \mathbb{M}_t, \tag{2}$$

where $[L] = \{1, \ldots, L\}$. We will use $\boldsymbol{x}_t^{\mathtt{u}}$ as a shorthand to represent the set of unmasked tokens at time t. At each denoising step, the dLLM processes the full state $\boldsymbol{x}_t$ and predicts a categorical distribution at every masked position:

$$p_\theta(x_t^i \mid \boldsymbol{x}_t^{\mathtt{u}}) = \mathrm{softmax}\Big(\ell_t^{\text{unguided}}(x_t^i \mid \boldsymbol{x}_t^{\mathtt{u}})\Big), \qquad i \in \mathbb{M}_t. \tag{3}$$

Throughout the paper, we will use ℓ_t to refer to logits and p_θ as the corresponding normalized probability distribution. A decoding rule then chooses which token positions to unmask and which token values to place at those positions. A common confidence-based decoding rule is to pick the token i whose predicted distribution $p_\theta(x_t^i \mid \boldsymbol{x}_t^{\mathtt{u}})$ has minimum entropy (which we denote as *high-confidence tokens*), and then sample from that distribution (Nie et al., 2026).

Parallel decoding. Instead of sampling one token at each diffusion timestep (sequential decoding), parallel decoding methods unmask k tokens at each step, where k can be fixed a priori (Nie et al., 2026) or adaptive at inference time (Kim et al., 2025b; Wu et al., 2025). We refer to these k token positions as $\mathbb{A}_t = \{a_1, \ldots, a_k\} \subseteq \mathbb{M}_t$ and $\boldsymbol{c} = (c_1, \ldots, c_k) \in \mathbb{V}^k$ as their proposed values. The true joint distribution over these k tokens would be

$$p_\theta(x_t^{a_1} = c_1, \ldots, x_t^{a_k} = c_k \mid \boldsymbol{x}_t^{\mathtt{u}}). \tag{4}$$

However, the model only outputs marginal distributions $p_\theta(x_t^{a_j} = c_j \mid \boldsymbol{x}_t^{\mathtt{u}})$ at every token position, so a common approximation is to model this joint distribution as a product of marginal distributions $\prod_{j=1}^k p_\theta(x_t^{a_j} = c_j \mid \boldsymbol{x}_t^{\mathtt{u}})$. The key challenge is to mitigate this approximation error, known as the joint-marginal mismatch (Ben-Hamu et al., 2025; Liu et al., 2025a). For instance, some works address this by selecting positions with limited dependencies using signals such as confidence scores (Wu et al., 2025), analysis of temporal stability of logits (Kim et al., 2025b; Ben-Hamu et al., 2025), or attention-based dependency graphs (Kim et al., 2026a). However, they are developed for unguided generation, and thus do not address recomputation of the guidance vector as in Equation 1 or whether the selection rules remain reliable for guided token distributions. An alternative set of approaches uses external planners, such as autoregressive LLMs (Israel et al., 2026; Liu et al., 2025a) to autoregressively commit the k tokens. In contrast to these methods, FastGuide is a hybrid decoding strategy that uses parallel decoding techniques for the reward guidance and autoregressive decoding over $\mathbb{A}_t$. Further, FastGuide obviates the need for external models by efficiently using the dLLM itself to autoregressively unmask tokens. We give a detailed comparison with other decoders in Appendix A.

Reward guidance for alignment. In this work, we focus on the gradient-based reward guidance framework (leaving discussion of other methods to Appendix A), which has been shown to outperform methods such as Best-of-N sampling (Liu et al., 2025b; Tejaswi et al., 2026). This paradigm draws inspiration from classifier-guidance approaches for continuous diffusion models, which has been remarkably successful in numerous domains for training-free steering (Daras et al., 2024; Dhariwal & Nichol, 2021; Thaker et al., 2025). For dLLMs, gradient-based guidance methods steer the dLLM logits with a differentiable sequence-level reward, such as a downstream reward model. As shown in Equation 1, the guided logits at each position are the sum of the dLLM logits and a per-token guidance vector $\boldsymbol{r}_t$ that shifts the distribution toward higher-reward tokens. The guidance vector $\boldsymbol{r}_t$ is estimated by completing the masked sequence, evaluating the completion with a reward model, and differentiating the reward with respect to the dLLM logits. Existing gradient-based guidance methods have focused on improved input completion techniques, e.g., by sampling token values from the predicted marginal distribution at each token (Rout et al., 2025), computing an expected embedding for each position with respect to the predicted distribution at each token (Murata et al., 2024), or interpolating between the two (Tejaswi et al., 2026). As such, they have been largely limited to sequential decoding evaluations, resulting in long inference times (Figure 1b). In this work, we focus on the orthogonal problem of accelerating reward guidance since computing forward and backward passes through large reward models at each diffusion timestep can be very costly.

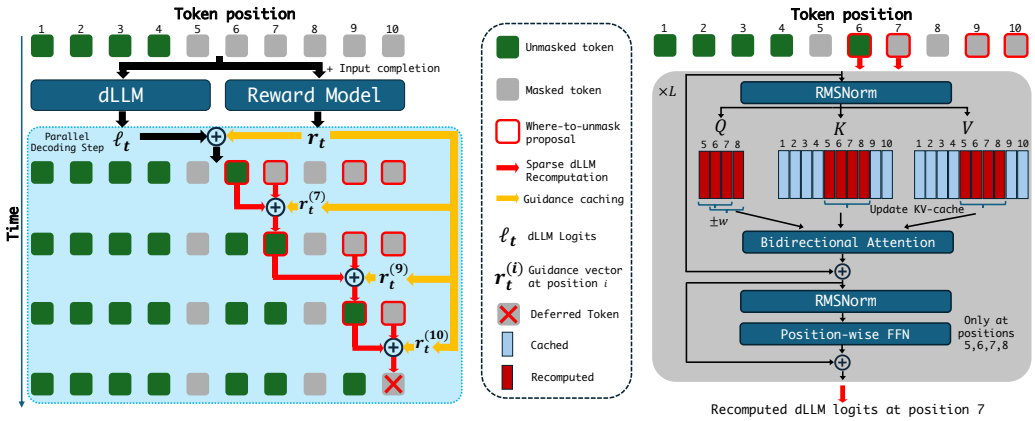


Figure 2: Overview of FastGuide. Left: Illustration of one hybrid decoding step of FastGuide with $k = 4$. FastGuide utilizes guidance caching and autoregressive sparse dLLM recomputation. After recomputation, tokens are deferred if token confidence drops below a threshold. Right: Sparse dLLM recomputation of token 7 after committing token 6, with window radius $w = 1$. Attention and FFN outputs are recomputed only at the selected positions, whose queries attend to the full sequence using updated K/V inside the windows and cached K/V elsewhere.

3 Method: FastGuide

In this section, we develop FastGuide, an algorithm to accelerate reward-guided decoding for dLLMs. First, in Section 3.1, we derive our hybrid decoding strategy using guidance caching and autoregressive dLLM recomputation. In Section 3.1.2, we use KV caching and sparse recomputation of attention (Ma et al., 2025) to efficiently perform autoregressive dLLM recomputation. Lastly, we introduce confidence deferral in Section 3.2, which is a technique to make decoding adaptive rather than unmasking a fixed number of tokens in each step. Our final method is summarized in Figure 2 and pseudocode is given in Appendix C. Throughout this section, we instantiate unguided parallel decoding as the widely used rule of confidence-based unmasking of the top k tokens (Nie et al., 2026), and we instantiate reward guidance as the recently introduced EntRGi algorithm (Tejaswi et al., 2026). In Section 4, we compare FastGuide with other parallel decoders and evaluate other guidance algorithms. Evaluations in this section were performed on RM-Bench (Liu et al., 2025b) with the Dream text diffusion model (Ye et al., 2025b) and Skywork-Reward-V2-Qwen3-0.6B reward model (Liu et al., 2026), with further experimental details given in Appendix B.3.

3.1 Hybrid Decoding: Guidance Caching and dLLM Recomputation

Recall from Equation 1 that both the dLLM logits and the guidance vector depend on the unmasked tokens $\boldsymbol{x}_t^{\mathrm{u}}$ at each timestep. Existing sequential reward-guided decoders recompute both terms after every unmasking, which requires one dLLM forward pass and one guidance computation per token. Since sequential reward-guided decoding has shown promising performance (Tejaswi et al., 2026), we take it as our starting point and seek to maintain its generation quality while reducing its inference time by unmasking k tokens per diffusion timestep. Since Equation 1 consists of two terms, this raises the question of whether the dLLM logits, the guidance vector, or both can be safely reused to unmask a candidate token set $\mathbb{A}_t = \{a_1, \ldots, a_k\}$ (defined in Section 2).

Separating dLLM and guidance reuse. To study this question, we independently vary whether each term is reused or recomputed within the candidate token set. Specifically, consider an ordering of $\mathbb{A}_t$ (e.g., descending confidence scores) and let $\boldsymbol{x}_t^{(s)} \in \mathbb{V}^L$ denote the sequence after unmasking the first s candidates of $\mathbb{A}_t$, with $\boldsymbol{x}_t^{(0)} = \boldsymbol{x}_t$. Then, for token position $a_j \in \mathbb{A}_t$, sequential guided decoding corresponds to iteratively computing for all $j \in \{1, \ldots, k\}$:

$$\mathrm{softmax}\left(\ell_t^{\mathrm{unguided}}\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(s)})^{\mathrm{u}}\right) + \boldsymbol{r}_t\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(s')})^{\mathrm{u}}\right)\right) \tag{5}$$

for $s = s' = j-1$. Instead, we analyze variants of Equation 5 where $s, s' \in \{0,\ j-1\}$. Here, $s = 0$ (resp. $s' = 0$) denotes reuse of the dLLM logits (resp. guidance vector) computed at the start of the

candidate group, whereas $s = j - 1$ (resp. $s' = j - 1$) denotes autoregressive recomputation after each token unmasking. We refer to computing guidance once and reusing it across the candidate group ($s' = 0$) as *guidance caching*, which is particularly attractive because guidance computation dominates inference time (Figure 1b). When both s and s' are 0, both terms are sampled from the guided distributions computed at the initial state, following the product-of-marginals approximation of Section 2. Figure 3 demonstrates that guidance caching along with dLLM recomputation retains most of the quality of recomputing both terms (orange vs. purple). Conversely, reusing the dLLM logits (blue and green curves) leads to quality degradation irrespective of guidance caching.

Hybrid decoding. This analysis motivates combining guidance caching with autoregressive dLLM recomputation, yielding a hybrid decoding strategy. As in parallel decoding, we amortize the guidance computation over k token candidates. Within this candidate set, we unmask candidates autoregressively, recomputing the dLLM logits after each unmasking. Thus, the token at position a_j is chosen from

$$\text{softmax}\left(\ell_t^{\text{unguided}}\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(j-1)})^{\text{u}}\right) + \boldsymbol{r}_t\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(0)})^{\text{u}}\right)\right). \tag{6}$$

3.1.1 Counterfactual Analysis of Hybrid Decoding

To better understand the asymmetry between guidance reuse and dLLM logits reuse, we run two counterfactual studies along sequential decoding trajectories (Figure 4).

Observation 1 (O1): guidance reuse has little effect at high-confidence positions. At every diffusion timestep, we hold the current dLLM logits fixed and replace the guidance vector with one cached from an earlier step. We measure the total-variation distance between the resulting guided distributions (Figure 4a). We find that this distance is smaller at high-confidence positions, which are precisely the positions that standard decoding strategies unmask (Ye et al., 2025b). This helps explain why guidance caching is effective.

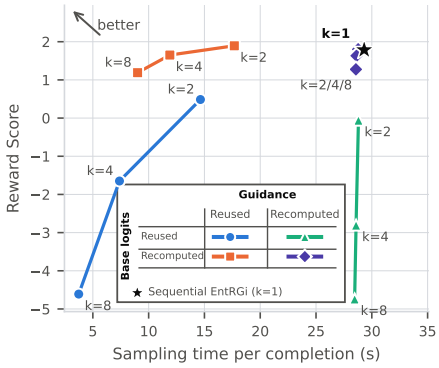


Figure 3: **Disentangling Reuse of dLLM Logits and Guidance.**

Observation 2 (O2): guidance can make co-proposed tokens locally incompatible. Tokens proposed in parallel from the same partially masked sequence need not be likely under the true joint distribution of the tokens since the model only outputs marginal distributions for each token. While this joint-marginal mismatch is present even without guidance (Ben-Hamu et al., 2025), we find that guidance can exacerbate this mismatch. To measure the effect of guidance on the joint-marginal mismatch, we consider sequential unguided trajectories and toggle guidance at each state, examining its effect on the top-2 confident tokens that would be committed in parallel. We quantify the joint-marginal mismatch at these two tokens as the dLLM-predicted Pointwise Mutual Information (PMI), which is the difference in log probabilities of the joint $p(a, b) = p(a)p(b \mid a)$ and product of marginal distributions $p(a)p(b)$. Negative PMI indicates that committing the first token lowers the dLLM-predicted probability of the second relative to its marginal probability. Figure 4b shows that guidance actually increases incompatibility among proposed tokens. The average number of incompatible pairs rises from 1.04 without guidance to 2.47 with guidance, and the fraction of prompts with at least one incompatible pair rises from 58% to 90%. This illustrates the role of dLLM recomputation in our hybrid decoding strategy as incompatible proposals can be revised before unmasking.

3.1.2 Sparse dLLM Recomputation

Despite the speedups obtained from reducing guidance computations, exact dLLM recomputation still requires the same number of dLLM forward passes as sequential decoding. To reduce this cost, we utilize dLLM acceleration techniques (Ma et al., 2025; Liu et al., 2025c). Specifically, for every hybrid decoding step, once the k candidate positions are fixed, recomputing logits for the next candidate requires updated logits only at its position a_j. Thus, we make a connection to Ma et al.

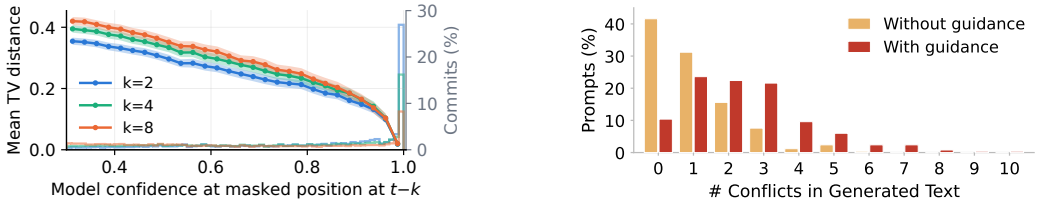


Figure 4: Left: (a) Replacing fresh guidance with a cached guidance produces small changes in high-confidence token distributions. Right: (b) Percentage of prompts exhibiting certain number of conflicts between proposed tokens (as measured by PMI being less than -1).

(2025), where the authors fix the token generation order of unguided sequential dLLM decoding to develop efficient sparse dLLM forward passes. Specifically, we can approximate the full forward pass by recomputing attention and feed-forward outputs for every transformer layer at a small set of positions: the union of windows of radius w around the most recently committed token and the next candidate position. To recompute attention, the queries at only these positions attend to the full sequence, using newly computed keys and values inside the windows and cached keys and values elsewhere (see Figure 2). This reduces the time complexity of each recomputation from $O(L^2)$ to $O(wL)$. The updated keys and values overwrite the cache for subsequent recomputation passes. The sparse dLLM forward produces approximate base logits $\widehat{\ell}_t^{\text{unguided}}\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(j-1)})^{\text{u}}\right)$, which we combine with the cached reward gradient as in Equation 6.

3.2 Adaptivity via Confidence Deferral

Unmasking exactly k tokens in every step can be restrictive (Wu et al., 2025) since the number of tokens that are safe to unmask together depends on the prompt and decoding state. We therefore introduce an adaptive confidence deferral strategy. Specifically, after sparse dLLM recomputation, we compute

$$\gamma_j = \max_{v \in \mathbb{V}} \text{softmax}\left(\widehat{\ell}_t^{\text{unguided}}\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(j-1)})^{\text{u}}\right) + \boldsymbol{r}_t\left(x_t^{a_j} \mid (\boldsymbol{x}_t^{(0)})^{\text{u}}\right)\right)(v). \tag{7}$$

If $\gamma_j \geq \tau$ for a chosen threshold τ, we commit the verified token. If not, the position remains masked and is reconsidered in a later step, when both the dLLM logits and the guidance are recomputed. The first token candidate is always committed, so every step makes progress. This makes the degree of parallelism adaptive and postpones low-confidence candidates until a later step so both dLLM logits and guidance can be recomputed.

In summary, for each decoding step, FastGuide performs 1 dLLM forward pass, 1 guidance computation step, and $k-1$ sparse recomputation passes, as opposed to the k dLLM forward passes and k guidance computations for sequential guided decoding. Refer to Appendix C for psuedocode.

4 Experiments

Below, we describe our experimental setup, deferring other experimental details to Appendix B.

Benchmarks. Following prior work on reward guidance, we evaluate FastGuide on three reward benchmarks (JudgeBench (Tan et al., 2025), RM-Bench (Liu et al., 2025b), and Reward-Bench-2 (Malik et al., 2025)), spanning various categories such as Coding, Math, Precise Instruction Following (IF), and Safety.

Baselines. We evaluate FastGuide on two popular open-source dLLMs: Dream-7B-Instruct (Ye et al., 2025b) and LLaDA-8B-Instruct (Nie et al., 2026). While existing parallel decoding baselines are evaluated on unguided decoding, our guidance caching strategy (Section 3) allows us to extend existing methods to the guided setting. To our knowledge, our work represents the first evaluation of these decoders in the reward-guided setting. We compare against several training-free parallel decoding baselines such as confidence-based decoding (which we abbreviate as Conf.) (Nie et al.,

	Method	Dream-7B Top@1	Dream-7B Seq Gap	Dream-7B LMUnit	Dream-7B s/gen	LLaDA-8B Top@1	LLaDA-8B Seq Gap	LLaDA-8B LMUnit	LLaDA-8B s/gen
	JudgeBench								
SEQ	Conf (BoN)	$+1.52_{\pm.16}$	–	$3.97_{\pm.03}$	$10.8_{\pm4.7}$	$+2.65_{\pm.21}$	–	$4.05_{\pm.03}$	$11.8_{\pm5.1}$
SEQ	Conf	$+2.53_{\pm.16}$	–	$4.05_{\pm.03}$	$37.3_{\pm15.5}$	$+3.23_{\pm.20}$	–	$4.08_{\pm.03}$	$45.3_{\pm18.5}$
PARALLEL	Conf (BoN)	$-3.14_{\pm.20}$	-4.66	$3.01_{\pm.05}$	$2.7_{\pm1.1}$	$-2.23_{\pm.22}$	-4.88	$3.05_{\pm.05}$	$3.0_{\pm1.3}$
PARALLEL	Conf	$+0.38_{\pm.20}$	-2.15	$3.64_{\pm.04}$	$9.7_{\pm3.8}$	$+1.54_{\pm.22}$	-1.69	$3.78_{\pm.04}$	$11.8_{\pm5.1}$
PARALLEL	Fast-dLLM	$+0.26_{\pm.21}$	-2.27	$3.62_{\pm.05}$	$7.9_{\pm4.1}$	$-0.12_{\pm.21}$	-3.35	$3.39_{\pm.05}$	$6.5_{\pm2.9}$
PARALLEL	KLASS	$-0.06_{\pm.22}$	-2.59	$3.61_{\pm.05}$	$10.8_{\pm12.5}$	$-0.81_{\pm.19}$	-4.04	$3.10_{\pm.05}$	$5.6_{\pm2.7}$
PARALLEL	EB-Sampler	$-0.49_{\pm.19}$	-3.02	$3.49_{\pm.04}$	$9.4_{\pm3.5}$	$+0.09_{\pm.21}$	-3.14	$3.52_{\pm.04}$	$10.1_{\pm3.8}$
PARALLEL	DAPD	$+1.92_{\pm.17}$	-0.61	$3.95_{\pm.03}$	$8.6_{\pm3.3}$	$+2.74_{\pm.22}$	-0.49	$3.99_{\pm.04}$	$9.4_{\pm4.2}$
PARALLEL	FastGuide	$\mathbf{+2.08}_{\pm.17}$	$\mathbf{-0.45}$	$\mathbf{4.02}_{\pm.03}$	$9.6_{\pm5.0}$	$\mathbf{+2.76}_{\pm.21}$	$\mathbf{-0.47}$	$\mathbf{4.03}_{\pm.03}$	$10.3_{\pm3.4}$
	RM-Bench								
SEQ	Conf (BoN)	$+4.41_{\pm.12}$	–	$4.06_{\pm.02}$	$6.3_{\pm1.9}$	$+4.39_{\pm.12}$	–	$3.94_{\pm.02}$	$6.6_{\pm2.2}$
SEQ	Conf	$+5.42_{\pm.11}$	–	$4.15_{\pm.02}$	$22.9_{\pm6.9}$	$+5.25_{\pm.11}$	–	$4.02_{\pm.02}$	$24.9_{\pm8.8}$
PARALLEL	Conf (BoN)	$-1.96_{\pm.14}$	-6.37	$2.75_{\pm.02}$	$1.6_{\pm.5}$	$-1.57_{\pm.14}$	-5.96	$2.79_{\pm.02}$	$1.7_{\pm.5}$
PARALLEL	Conf	$+2.32_{\pm.14}$	-3.09	$3.61_{\pm.02}$	$5.8_{\pm1.7}$	$+2.83_{\pm.13}$	-2.43	$3.63_{\pm.02}$	$6.3_{\pm2.2}$
PARALLEL	Fast-dLLM	$+1.34_{\pm.15}$	-4.08	$3.51_{\pm.03}$	$5.6_{\pm2.2}$	$+0.74_{\pm.14}$	-4.51	$3.31_{\pm.03}$	$4.5_{\pm1.7}$
PARALLEL	KLASS	$+0.38_{\pm.15}$	-5.04	$3.33_{\pm.03}$	$9.6_{\pm9.9}$	$-0.11_{\pm.14}$	-5.36	$3.03_{\pm.03}$	$4.1_{\pm1.9}$
PARALLEL	EB-Sampler	$+0.40_{\pm.15}$	-5.01	$3.20_{\pm.02}$	$5.9_{\pm2.0}$	$+0.49_{\pm.15}$	-4.76	$3.19_{\pm.02}$	$5.9_{\pm1.6}$
PARALLEL	DAPD	$+3.58_{\pm.13}$	-1.83	$3.87_{\pm.02}$	$5.9_{\pm1.3}$	$+4.36_{\pm.12}$	-0.89	$3.90_{\pm.02}$	$6.2_{\pm2.3}$
PARALLEL	FastGuide	$\mathbf{+4.70}_{\pm.12}$	$\mathbf{-0.72}$	$\mathbf{4.04}_{\pm.02}$	$6.2_{\pm1.5}$	$\mathbf{+4.53}_{\pm.12}$	$\mathbf{-0.72}$	$\mathbf{3.92}_{\pm.02}$	$6.5_{\pm1.8}$
	Reward-Bench-2								
SEQ	Conf (BoN)	$+2.60_{\pm.10}$	–	$4.18_{\pm.01}$	$5.5_{\pm1.4}$	$+2.19_{\pm.09}$	–	$4.00_{\pm.02}$	$6.1_{\pm1.6}$
SEQ	Conf	$+3.39_{\pm.10}$	–	$4.23_{\pm.01}$	$20.4_{\pm5.2}$	$+2.88_{\pm.09}$	–	$4.05_{\pm.02}$	$26.9_{\pm9.3}$
PARALLEL	Conf (BoN)	$-1.98_{\pm.09}$	-4.58	$2.95_{\pm.02}$	$1.4_{\pm.4}$	$-2.22_{\pm.08}$	-4.41	$2.88_{\pm.02}$	$1.5_{\pm.4}$
PARALLEL	Conf	$+1.24_{\pm.10}$	-2.15	$3.78_{\pm.02}$	$5.2_{\pm1.2}$	$+0.98_{\pm.09}$	-1.90	$3.67_{\pm.02}$	$6.6_{\pm2.3}$
PARALLEL	Fast-dLLM	$+1.36_{\pm.11}$	-2.03	$3.82_{\pm.02}$	$6.8_{\pm2.5}$	$+0.05_{\pm.09}$	-2.83	$3.55_{\pm.02}$	$7.9_{\pm4.0}$
PARALLEL	KLASS	$+1.06_{\pm.10}$	-2.33	$3.83_{\pm.02}$	$15.7_{\pm13.3}$	$-0.09_{\pm.09}$	-2.97	$3.44_{\pm.02}$	$8.0_{\pm4.4}$
PARALLEL	EB-Sampler	$+0.56_{\pm.10}$	-2.83	$3.57_{\pm.02}$	$6.6_{\pm2.0}$	$+0.01_{\pm.09}$	-2.87	$3.40_{\pm.02}$	$9.0_{\pm3.1}$
PARALLEL	DAPD	$+3.01_{\pm.10}$	-0.38	$4.15_{\pm.01}$	$6.2_{\pm1.7}$	$+2.49_{\pm.09}$	-0.39	$\mathbf{4.00}_{\pm.02}$	$7.2_{\pm2.2}$
PARALLEL	FastGuide	$\mathbf{+3.25}_{\pm.10}$	$\mathbf{-0.14}$	$\mathbf{4.19}_{\pm.01}$	$7.3_{\pm2.1}$	$\mathbf{+2.58}_{\pm.09}$	$\mathbf{-0.30}$	$\mathbf{4.00}_{\pm.02}$	$7.3_{\pm2.5}$

Table 1: Quantitative results for reward-guided generation on JudgeBench, RM-Bench, and Reward-Bench-2. All parallel decoding methods use cached EntRGi reward guidance, unless (BoN) is mentioned, denoting Best-of-N sampling. Subscripts denote the standard error over prompts.

2026), Fast-dLLM (Wu et al., 2025), KLASS (Kim et al., 2025b), EB-sampler (Ben-Hamu et al., 2025) and DAPD (Kim et al., 2026a). For fair comparison, we do not utilize additional optimizations of the full dLLM forward pass such as prompt caching (Wu et al., 2025), which are orthogonal improvements and can be combined with FastGuide. We also compare to Best-Of-N sampling with $N = 4$. We use the recent state-of-the-art reward guidance method EntRGI (Tejaswi et al., 2026) with the Skywork-Reward-V2-Qwen3-1.7B reward model (Liu et al., 2026). We defer evaluation of other guidance methods and different reward models to Appendix D.1 and D.2.

Metrics. Following Tejaswi et al. (2026), we generate 4 trajectories for each prompt and measure generation quality by the reward model score. We report Top@1 (the maximum reward model score across all trajectories) and Seq Gap (the gap between parallel decoding Top@1 and the corresponding sequential decoding baseline using the same guidance algorithm). To monitor reward overoptimization (Gao et al., 2023), we report LMUnit (external judge LLM LMUnit-Qwen2.5-72B (Saad-Falcon et al., 2025) score from $1 - 5$). We further report Avg@4 (the average reward model score across the 4 trajectories) in Appendix D.3. To evaluate inference time, we report s/gen (wall-clock inference time for each generation), computed on a Nvidia A5000 GPU.

Hyperparameters. We set the generation length to be 128 for all methods, and baseline hyperparameters can be found in Appendix B.2. FastGuide uses a candidate pool of $k = 8$ for all experiments along with confidence deferral parameter $\tau = 0.5$ and attention window size $w = 2$. We consider a higher throughput regime by increasing k to 16 in Appendix D. We use batch size 4 for Dream-7B and batch size 2 for LLaDA-8B.

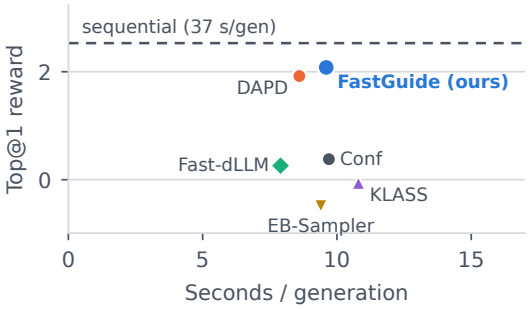


Figure 5: Quality versus inference time on JudgeBench using Dream-7B model.

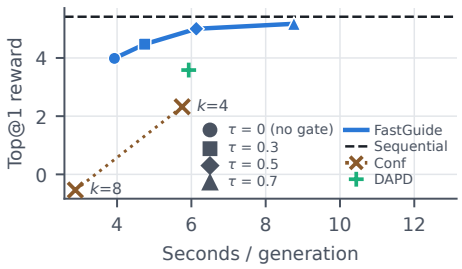


Figure 6: τ sweep for confidence deferral on RM-Bench using Dream-7B model.

4.1 Main Results

Reward vs Inference Time. Table 1 and Figure 5 show our main results. FastGuide is 2.8–4.4× faster than sequential guided decoding across the three benchmarks and two models while remaining within 0.75 reward points of sequential guidance on both models. Its high LMUnit scores provide complementary evidence of response quality and also guard against reward overoptimization. Compared to unguided sequential Best-of-N sampling, sequential guidance improves Top@1 by 0.6–1.0 reward points but costs 3.5–4.4× the inference time. FastGuide reduces this cost to 0.9–1.3× that of Best-of-N while retaining a Top@1 improvement of 0.1–0.65 reward points. These conclusions also hold on LLaDA-8B, where FastGuide again attains the smallest Seq Gap, while Fast-dLLM and KLASS degrade even more severely than on Dream-7B. Cached guidance on top of confidence-based decoding (Conf row) improves Top@1 by roughly 4 reward points over its unguided counterpart (Conf (BoN) row), confirming that guidance caching preserves the benefit of guidance. FastGuide further narrows this gap through sparse dLLM recomputation and confidence deferral. An important benefit of FastGuide is that sparse dLLM recomputation transforms the compute-bound full dLLM forward pass, that involves bidirectional attention over the whole sequence, into a memory-bound sparse forward over a few positions. Thus, standard techniques such as batching can reduce inference time per trajectory, which we explore in Appendix D.4.

Comparison to Parallel Decoders. Most existing parallel decoders lose a large part of the benefit of guidance, with Fast-dLLM, KLASS, and EB-sampler suffering substantial performance degradation relative to sequential decoding. In contrast, DAPD, which selects positions from the dLLM's attention graph rather than from marginal confidence, is by far the strongest baseline. FastGuide matches or improves on DAPD on every benchmark and model, with the largest gains on RM-Bench (+1.1 Top@1 on Dream-7B). Figure 7 analyzes this Seq Gap further by splitting performance along benchmark categories. DAPD exhibits larger reward gaps on coding and math tasks. This pattern is consistent with the greater sensitivity to incompatible token choices in these categories, and the qualitative example in Figure 8 further illustrates this. In contrast, for categories such as chat and safety, an incompatible pair of tokens costs little fluency. FastGuide has a small Seq Gap across nearly all categories, highlighting the benefit of sparse dLLM recomputation in the reward-guided setting.

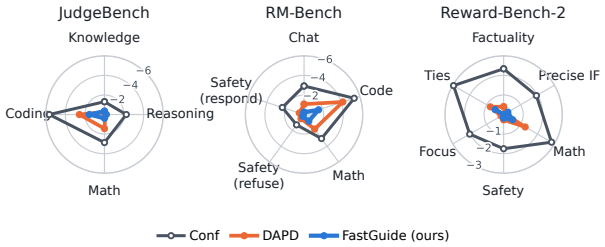


Figure 7: **Sequential gap by category (closer to 0 is better).** FastGuide has the smallest gap to sequential decoding, especially in reasoning categories such as coding where baselines degrade.

Qualitative Samples. Figure 8 shows an illustrative mathematical reasoning prompt from RM-Bench. Unguided DAPD has a correct reasoning trace, but under guidance it commits an incorrect line of arithmetic reasoning (5^2 in place of 8^2) that the rest of the derivation then propagates to a wrong final answer, showing an instance of **O2**. Instead, FastGuide recomputes each candidate

Prompt: If the endpoints of a line segment are (2, -2) and (10, 4), what is the length of the segment?

DAPD, guided Reward -1.22

```
To find the length of the line
segment, we can use the distance
formula, which is $\sqrt{(x_2 -
x_1)^2 + (y_2 - y_1)^2}$. Plugging
in the given endpoints, we
have $\sqrt{(10 - 2)^2) + (4 -
(-2)) ^2} = \sqrt{5^2 + 6^2} =
\sqrt{10 + 36} = \sqrt{46}$.
The length of the line segment is
$\boxed{6}$ units.The answer is:
$\boxed{6}$
```

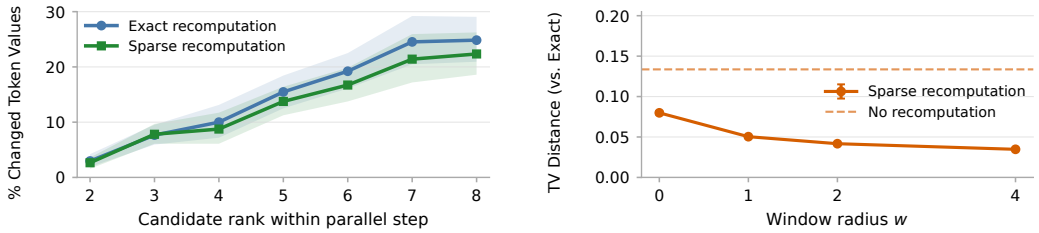
DAPD, unguided Reward $+6.59$

```
We can use the distance formula
to find the length of the line
segment. The distance formula
between two points $(x_1, y_1)$
and $(x_2, y_2)$ is given by
$\sqrt{(x_2 - x_1)^2 + (y_2 -
y_1)^2}$. Plugging in the given
values, we have $\sqrt{(10 -
2)^2 + (4 - (-2))^2} = \sqrt{8^2
+ 6^2} = \sqrt{64 + 36} =
\sqrt{100}
```

FastGuide Reward $+10.44$

```
To find the length of the line
segment, we can use the distance
formula, which is derived from
the Pythagorean Theorem. The
formula is: Distance = sqrt((x2 -
x1)^2 + (y2 - y1)^2) Plugging in
the given coordinates: Distance =
sqrt((10 - 2)^2 + (4 - (-2))^2) =
sqrt(8^2 + 6^2) = sqrt(64 + 36) =
sqrt(100) = 10 So, the length of
the line segment is 10 units.
```

Figure 8: **Qualitative generations on Math RM-Bench prompt.** Red highlighted tokens denote incorrect tokens. This reinforces that guidance can worsen performance of parallel decoders, while FastGuide succeeds.



(a) (b)

Figure 9: **Analysis of sparse dLLM recomputation vs. exact dLLM recomputation.**

token distribution before unmasking it and recovers the correct answer. The reward model scores in Figure 8 reflect this ordering. Appendix D.9 shows further qualitative examples.

4.2 Ablation Studies

Effect of confidence deferral. Figure 6 sweeps the deferral threshold τ of FastGuide on RM-Bench, showing that τ provides an extra knob for interpolating the quality-efficiency tradeoff. Specifically, without deferral (τ=0), unmasking a fixed 8 tokens per step is fastest, but generation quality degrades. As more tokens are deferred, quality increases, approaching the performance of sequential decoding, but at the expense of inference time due to increased guidance computation steps and full dLLM forwards.

Effect of dLLM Recomputation. Figure 9 analyzes the sparse dLLM recomputation step of FastGuide. Specifically, Figure 9b shows that sparse dLLM recomputation is a useful approximation of the full dLLM forward pass as measured by the TV distance between sparsely and exactly recomputed token distributions, while Figure 9a shows that both sparse and exact recomputation change a large fraction of token values during decoding. Finally, Appendix D.6 shows that generation quality is largely agnostic to the order of the candidate set for which autoregressive decoding is performed.

5 Conclusion

In this paper, we introduced FastGuide, a training-free hybrid decoding algorithm to accelerate reward-guidance for diffusion language models. Our findings show that while guidance can be reused across diffusion timesteps, guidance can worsen the token incompatibilities of parallel decoding. To address this, FastGuide performs guidance caching combined with sparse dLLM recomputation to autoregressively and efficiently recompute each proposed token distribution in a decoding step. FastGuide achieves speedups up to $4.4\times$ over sequential guided decoding while retaining similar generation quality. To the best of our knowledge, our work is the first comprehensive study of accelerating training-free, gradient-based reward guidance for diffusion language models and the first evaluation of parallel decoders under reward guidance. We hope the ideas proposed in this paper will inspire future work on developing parallel decoders specifically for the guided setting.

AI USE STATEMENT

In this work, we used generative AI tools for tasks: design or provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools for tasks: propose or refine hypotheses and help develop theoretical models or conceptual frameworks. Tasks such as: generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, clean and reformat dataset are not applicable to this work. Additionally, we used generative AI tools for creating/modifying scientific figures, writing software code, brainstorming, and identifying relevant literature. We have reviewed all AI-assisted work. We checked LLM-generated code, verified all references by hand. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have provided all experimental details in Appendix B, and will release the code upon acceptance.

REFERENCES

Sudhanshu Agrawal, Risheek Garrepalli, Raghavv Goel, Christopher Lott, Fatih Porikli, and Mingu Lee. Structuring The Future: Diffusion LLM Speculative Decoding via Calibrated Draft Graphs. *arXiv preprint arXiv:2509.18085*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.24857.

Angela Castillo, Jonas Kohler, Juan C. Pérez, Juan Pablo Pérez, Albert Pumarola, Bernard Ghanem, Pablo Arbeláez, and Ali Thabet. Adaptive Guidance: Training-free Acceleration of Conditional Diffusion Models. *arXiv preprint arXiv:2312.12487*, 2023.

Junxia Cui, Haotian Ye, Runchu Tian, Hongcan Guo, Jinya Jiang, Haoru Li, Chaojie Ren, Yiming Huang, Kaijie Zhu, Zhongkai Yu, Kun Zhou, and Jingbo Shang. SimSD: Simple Speculative Decoding in Diffusion Language Models. *arXiv preprint arXiv:2606.02544*, 2026.

Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems. *arXiv preprint arXiv:2410.00083*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Anh-Dung Dinh, Daochang Liu, and Chang Xu. Compress Guidance in Conditional Diffusion Sampling. *arXiv preprint arXiv:2408.11194*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pp. 10835–10866. PMLR, 2023.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self Speculative Decoding for Diffusion Large Language Models. *arXiv preprint arXiv:2510.04147*, 2025.

Google DeepMind. Gemini diffusion. `https://blog.google/technology/google-deepmind/gemini-diffusion/`, 2025.

Ligong Han, Hao Wang, Han Gao, Kai Xu, and Akash Srivastava. S2D2: Fast Decoding for Diffusion LLMs via Training-Free Self-Speculation. *arXiv preprint arXiv:2603.25702*, 2026.

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. *arXiv preprint arXiv:2507.18578*, 2025.

Inception Labs. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *Advances in neural information processing systems*, 38:52870–52888, 2026.

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

Minseo Kim, Coleman Hooper, Aditya Tomar, Chenfeng Xu, Mehrdad Farajtabar, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Beyond next-token prediction: A performance characterization of diffusion versus autoregressive language models. *arXiv preprint arXiv:2510.04146*, 2025a.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. In *Advances in Neural Information Processing Systems*, 2025b. arXiv:2511.05664.

Yeongmin Kim, Donghyeok Shin, Byeonghu Na, Minsang Park, Richard Lee Kim, and Il-Chul Moon. Lookahead Sample Reward Guidance for Test-Time Scaling of Diffusion Models. *arXiv preprint arXiv:2602.03211*, 2026b.

Younjoo Lee, Seungkyun Dan, Junghoo Lee, Jaiyoung Park, and Jung Ho Ahn. DyLLM: Efficient diffusion LLM inference via saliency-based token selection and partial attention. *arXiv preprint arXiv:2603.08026*, 2026.

Bolian Li, Yifan Wang, Anamika Lochab, Ananth Grama, and Ruqi Zhang. Cascade Reward Sampling for Efficient Decoding-Time Alignment. *arXiv preprint arXiv:2406.16306*, 2024.

Kaihua Liang, Xin Tan, An Zhong, Hong Xu, and Marco Canini. FOCUS: DLLMs know how to tame their compute bound. In *International Conference on Machine Learning*, 2026.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-Guided Speculative Decoding for Efficient LLM Reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. In *International Conference on Learning Representations*, volume 2025, pp. 88953–88979, 2025a.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Rm-bench: Benchmarking reward models of language models with subtlety and style. In *International Conference on Learning Representations*, volume 2025, pp. 44323–44355, 2025b.

Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, et al. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. In *International Conference on Learning Representations*, volume 2026, pp. 133805–133838, 2026.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-Cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025c.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-Free Video Diffusion Model Acceleration with High Quality. *arXiv preprint arXiv:2410.19355*, 2024.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.15781.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. *arXiv preprint arXiv:2506.01937*, 2025.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled Decoding from Language Models. *arXiv preprint arXiv:2310.17022*, 2023.

Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Bac Nguyen, Stefano Ermon, and Yuki Mitsufuji. G2d2: Gradient-guided discrete diffusion for inverse problem solving. *arXiv preprint arXiv:2410.14710*, 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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Chinmay Pani, Zijing Ou, and Yingzhen Li. Test-time alignment of discrete diffusion models with sequential Monte Carlo. *arXiv preprint arXiv:2505.22524*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Litu Rout, Andreas Lugmayr, Yasamin Jafarian, Srivatsan Varadharajan, Constantine Caramanis, Sanjay Shakkottai, and Ira Kemelmacher-Shlizerman. Test-time anchoring for discrete diffusion posterior sampling. *arXiv preprint arXiv:2510.02291*, 2025.

Jon Saad-Falcon, Rajan Vivek, William Berrios, Nandita Shankar Naik, Matija Franklin, Bertie Vidgen, Amanpreet Singh, Douwe Kiela, and Shikib Mehri. Lmunit: Fine-grained evaluation with natural language unit tests. In *EMNLP (Findings)*, pp. 3303–3324, 2025.

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.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *International Conference on Machine Learning*, 2025.

Marta Skreta, Tara Akhound-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alán Aspuru-Guzik, Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. Feynman-Kac correctors in diffusion: Annealing, guidance, and product of experts. In *International Conference on Machine Learning*, 2025.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations*, 2025.

Atula Tejaswi, Litu Rout, Constantine Caramanis, Sanjay Shakkottai, and Sujay Sanghavi. Entropy aware reward guidance for diffusion language model alignment. *arXiv preprint arXiv:2602.05000*, 2026.

Darshan Thaker, Abhishek Goyal, and René Vidal. Frequency-guided posterior sampling for diffusion-based image restoration. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12873–12882. IEEE, 2025.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking Discrete Diffusion Models with Inference-Time Scaling. *arXiv preprint arXiv:2503.00307*, 2025.

Zichen Wen, Jiashu Qu, Zhaorun Chen, Xiaoya Lu, Dongrui Liu, Zhiyuan Liu, Ruixi Wu, Yicun Yang, Xiangqi Jin, Haoyun Xu, et al. The devil behind the mask: An emergent safety vulnerability of diffusion llms. In *International Conference on Learning Representations*, volume 2026, pp. 5460–5489, 2026.

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. *arXiv preprint arXiv:2505.22618*, 2025.

Luhuan Wu, Brian L Trippe, Christian A Naesseth, David M Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Shutong Wu and Jiawei Zhang. Free Draft-and-Verification: Toward Lossless Parallel Decoding for Diffusion Large Language Models. *arXiv preprint arXiv:2510.00294*, 2025.

Yanzheng Xiang, Lan Wei, Yizhen Yao, Qinglin Zhu, Hanqi Yan, Chen Jin, Philip Alexander Teare, Dandan Zhang, Lin Gui, Amrutha Saseendran, and Yulan He. Stop the Flip-Flop: Context-Preserving Verification for Fast Revocable Diffusion Decoding. *arXiv preprint arXiv:2602.06161*, 2026.

Zhen Xiong, Yujun Cai, Zhecheng Li, and Yiwei Wang. Unveiling the potential of diffusion large language model in controllable generation. In *International Conference on Learning Representations*, volume 2026, pp. 66670–66686, 2026.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-Based Diffusion Language Models for Text Generation. *arXiv preprint arXiv:2410.21357*, 2024.

Feihong Yan, Qingyan Wei, Jiayi Tang, Jiajun Li, Yulin Wang, Xuming Hu, Huiqi Li, and Linfeng Zhang. LazyMAR: Accelerating Masked Autoregressive Models via Feature Caching. *arXiv preprint arXiv:2503.12450*, 2025.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *International Conference on Learning Representations*, volume 2025, pp. 77875–77898, 2025a.

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

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential Monte Carlo. In *International Conference on Machine Learning*, 2024.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

Appendix Contents

A Extended Related Work **14**

B Experimental Details **15**

B.1 Benchmarks 15

B.2 Hyperparameters 16

B.3 Experimental Details for Section 3 17

C FastGuide pseudocode **18**

D Additional Experiments **19**

D.1 Guidance Caching with other Reward Guidance Algorithms 19

D.2 FastGuide with Different Reward Models 19

D.3 Measuring Average Reward Model Performance and Throughput 20

D.4 Effect of Batching 24

D.5 Sparse dLLM Verification Window Size Ablation 24

D.6 Verification Order Ablation 25

D.7 Effect of Greedy Token Selection 25

D.8 Extended Computational Analysis 25

D.9 Qualitative Examples 26

A Extended Related Work

Other reward guidance methods. Aligning dLLM outputs with a reward can be done by fine-tuning or at inference time. Training-based approaches adapt RLHF-style objectives (Ouyang et al., 2022; Rafailov et al., 2023) to dLLMs, e.g., via policy-gradient or preference optimization (Zhao et al., 2025; Zhu et al., 2025), but require retraining whenever the reward changes. Training-free approaches instead steer a frozen dLLM at inference time and split into two families. Sampling-based methods such as Best-Of-N or Sequential Monte Carlo (SMC) methods only require reward evaluations, but can require many sequences/particles to be decoded (Wu et al., 2023; Zhao et al., 2024; Singhal et al., 2025; Skreta et al., 2025; Pani et al., 2025). Gradient-based methods instead differentiate the reward model with respect to the dLLM logits and steer a single trajectory, at the cost of a forward and backward pass through the reward model at every diffusion timestep (Murata et al., 2024; Rout et al., 2025; Tejaswi et al., 2026). One of the main challenges of reward guidance is obtaining gradients since the tokens in the input sequence are discrete. To tackle this, existing methods propose to use stop-gradient approximations (Rout et al., 2025) or take gradients with respect to the continuous token embeddings (Murata et al., 2024; Tejaswi et al., 2026).

Accelerating guidance. In continuous diffusion, the cost of guidance has been reduced by only computing classifier gradients on a subset of timesteps (Dinh et al., 2024), by skipping the classifier-free guidance residual (Castillo et al., 2023; Lv et al., 2024; Yan et al., 2025), or by replacing reward backpropagation with a closed-form estimate (Kim et al., 2026b). Our guidance caching has two key differences. In continuous diffusion, the state evolves smoothly with the noise level, so guidance reuse is controlled by a timestep schedule. In contrast, in masked dLLMs, the effect of guidance reuse is position-dependent as well as timestep-dependent. While guidance may change across the entire sequence, **O1** demonstrates that at high-confidence positions (which are precisely the positions that are unmasked), the guided token distribution is stable. Second, our work provides a novel observation as to how guidance interacts with the joint-marginal mismatch (**O2**). For traditional autoregressive LLMs, segment-level and blockwise methods amortize reward-model evaluations over

several tokens (Li et al., 2024; Mudgal et al., 2023; Liao et al., 2025), but they use the reward as a score for search or rejection rather than as a gradient, and further do not face the joint-marginal mismatch since tokens are generated sequentially.

Parallel decoding. We now review other parallel decoders. The first class uses external models to supply joint structure to mitigate the joint-marginal mismatch. For instance, Israel et al. (2026), Liu et al. (2025a), and Xu et al. (2024) use the structure from an autoregressive verifier, a copula, or an energy function respectively. However, in the reward-guided setting, this requires the auxiliary model to be consistent with the guided distribution, or yield similarly high-reward results. The second class is training-free self-verification methods. Self-speculative decoders (Gao et al., 2025; Wu & Zhang, 2025; Agrawal et al., 2025; Han et al., 2026; Cui et al., 2026) perform parallel generation by drafting several tokens and verifying each draft conditioned on the earlier ones, using batched full forward passes or a single pass over a duplicated sequence. However, the goal of these methods is to remain lossless by using rejection sampling techniques, which also do not cleanly extend to the reward-guided setting. The final class of methods is revocable parallel decoders (Wang et al., 2025; Hong et al., 2025; Xiang et al., 2026), which unmask first and re-mask tokens that fail a verification step. We emphasize that these methods have not been evaluated under gradient-based reward guidance, which presents its own challenges of guidance recomputation and the influence of guidance on token compatibilities (Section 3).

FastGuide differs from these methods in two main ways. First, FastGuide trades exactness for cost: rather than relying on an auxiliary model or on full forward passes to remain lossless, it approximates each recomputed token distribution with a sparse forward pass over a small window (Ma et al., 2025), and we validate this approximation empirically (Section 4.2). Second, FastGuide acts *before* unmasking: the recomputed distribution is used to re-select the token value and to defer low-confidence candidates, rather than to accept or reject a drafted token or to re-mask a token after it has already entered the context.

Inference-time speedups. Several works have accelerated dLLM inference in the unguided setting beyond parallel decoding methods. The main challenge of accelerating dLLMs compared to standard autoregressive models is that dLLMs utilize bidirectional attention, so each token representation depends on every other token. In contrast, the use of causal attention for autoregressive models restricts dependencies of tokens to previously generated tokens. To address this, several works such as DyLLM (Lee et al., 2026), dKV-Cache (Ma et al., 2025), and dLLM-Cache (Liu et al., 2025c) propose methods to use selective attention, diffusion model layer skipping, and KV caching to speed up diffusion model forward passes. Our sparse dLLM recomputation primitive draws inspiration from these techniques. Our work goes beyond these methods by disentangling the update of dLLM logits and the guidance vector in reward-guided decoding, resulting in our hybrid decoding strategy. As Section 4 shows, autoregressive dLLM recomputation helps recondition the dLLM after every token unmasking, reducing token incompatibilities among co-proposed tokens, which guidance makes more frequent (**O2**). We leave further optimization of dLLM forward passes such as attention-based KV caching (Lee et al., 2026) to future work.

B Experimental Details

B.1 Benchmarks

The descriptions below use *high reward* and *low reward* for the preferred and rejected responses supplied by each benchmark. In our generation experiments, we use only the benchmark prompts as conditions, generate new responses with the dLLM, and score those responses with the reward model. Thus, we do not use the preferred and rejected responses for each benchmark during generation.

JudgeBench. JudgeBench (Tan et al., 2025) evaluates whether a judge can distinguish objectively correct answers from plausible but incorrect ones in four categories: general knowledge, reasoning, mathematics, and coding. Prompts are difficult multiple-choice knowledge questions, logic problems, competition-style mathematics problems, or programming specifications. Each example pairs two responses produced by the same model; the objectively correct response is the high-reward

response, while the response containing a subtle factual, logical, mathematical, or implementation error is the low-reward response. The benchmark consists of 270 prompts.

RM-Bench. RM-Bench (Liu et al., 2025b) tests whether reward models prioritize substantive quality over response style across five domains: `chat`, `code`, `math`, `safety-refuse`, and `safety-response`. Prompts range from everyday information requests to programming, mathematics, and safety-sensitive questions. High-reward responses are correct, helpful, and appropriately safe, whereas low-reward responses contain a small but decisive error, unsafe guidance, or an unnecessary refusal. Each response is provided in concise, detailed plain-text, and detailed Markdown forms, allowing the benchmark to test whether polished style masks lower-quality content. The benchmark consists of 1327 prompts.

Reward-Bench-2. Reward-Bench-2 (Malik et al., 2025) covers factuality, precise instruction following, mathematics, safety, focus, and ties. Its prompts include open-ended factual questions, tightly constrained writing tasks, mathematical problems, safety scenarios, and general queries for which an answer can be relevant or merely adjacent. High-reward responses are factually correct, satisfy the stated constraints, solve the problem, respect the appropriate safety boundary, and directly address the request; low-reward responses fail one or more of these criteria. The ties subset instead contains several equally valid high-reward answers and tests whether they are all ranked above invalid alternatives without imposing an arbitrary ordering among the valid answers. The benchmark consists of 1825 prompts.

B.2 Hyperparameters

Sampling Details. We fix the generation length $L = 128$ for all methods (excluding the benchmark prompt). Tokens are sampled with temperature 0.7 for all methods, mirroring the evaluation of Tejaswi et al. (2026). For FastGuide, this applies both to the first token unmasked in each step and to every subsequent candidate, which is sampled from its recomputed guided distribution (Algorithm 1). Appendix D.7 evaluates the deterministic variant that takes the most likely recomputed token instead.

Reward Guidance Details. We follow the protocol of EntRGi (Tejaswi et al., 2026), where guidance consists of 3 gradient backpropagation steps of the reward model at every diffusion timestep. This choice was found to produce high-quality samples while mitigating reward hacking (Tejaswi et al., 2026). Since LLaDA (Nie et al., 2026) has a different tokenizer than the Skywork reward models, we adopt the choice of EntRGi and set the embedding of all non-overlapping tokens to be the zero embedding.

Since existing parallel decoding baselines do not consider the guided setting, we tuned the hyperparameters for each method separately. We consider two regimes of FastGuide: one which achieves higher quality, lower throughput and the other which achieves lower quality, higher throughput. We refer to these as the *high-quality regime* and the *high-throughput regime*, respectively. FastGuide uses a candidate pool size of $k = 8$ in the first and $k = 16$ in the second (evaluated in Appendix D). For each regime, we picked the hyperparameters of every baseline such that it gives the highest quality at throughput comparable to FastGuide in that regime. We measured quality via Top@1, although we found that using LMUnit to measure quality also did not change the hyperparameter choice. Each baseline was swept on 64 RM-Bench prompts. Below, we describe the hyperparameters for each baseline and summarize the choices for each hyperparameter in Table 2.

Fast-dLLM Hyperparameters. Fast-dLLM (Wu et al., 2025) has one key parameter, which is the confidence threshold c for unmasking. In every step, all tokens with confidence score higher than c are unmasked.

KLASS Hyperparameters. KLASS (Kim et al., 2025b) has three parameters: a confidence threshold c, a KL threshold ϵ, and a history length h. In every step, a token is unmasked if its confidence score is higher than c and the KL divergence between its predicted distributions at consecutive steps has stayed below ϵ for the last h steps; if no token qualifies, the single most confident token is unmasked. We swept $c \in [0.1, 0.9]$, $\epsilon \in [0.001, 5]$, and $h \in \{1, 2\}$. We found that the KL stability rule of KLASS does not generalize to guided distributions: at the published settings

($c = 0.9$, $\epsilon \leq 0.01$, $h = 2$) it often defaults to sequential decoding, since many token distributions change drastically after guidance, and both thresholds must be relaxed substantially to unmask several tokens per step. Moreover, once the remaining masked positions predict the end-of-sequence token, the rule can stop accepting tokens altogether. This explains why KLASS has much longer and more variable inference time compared to other baselines in Table 1.

EB-Sampler Hyperparameters. EB-Sampler (Ben-Hamu et al., 2025) has one key parameter, which is the entropy budget γ. In every step, masked tokens are sorted by increasing entropy of their predicted distribution, and the longest prefix whose total entropy, excluding its largest entry, is at most γ is unmasked. The lowest-entropy token is always unmasked, so $\gamma = 0$ recovers sequential decoding and larger γ unmasks more tokens per step. We swept $\gamma \in [0.1, 16]$.

DAPD Hyperparameters. DAPD (Kim et al., 2026a) builds a dependency graph over masked tokens, connecting two tokens if their symmetrized attention score (averaged over heads and over the top 30% of layers) exceeds a threshold, and unmasks an independent set of this graph in every step. The threshold increases linearly during decoding from $\tau_{\min} = 0.005$ to $\tau_{\max} = 0.05$. Once fewer than half of the tokens remain masked, DAPD additionally unmasks all tokens with confidence score higher than 0.9. We sweep the threshold schedule by multiplying by a single scale s, since larger thresholds remove edges from the graph and unmask more tokens per step. We swept $s \in [0.5, 32]$.

Method	High-quality regime	High-throughput regime
Confidence	top-4 per step	top-8 per step
Fast-dLLM	$c = 0.8$	$c = 0.6$
KLASS	$c = 0.4$, $\epsilon = 1.5$, $h = 1$	$c = 0.25$, $\epsilon = 1.0$, $h = 1$
EB-Sampler	$\gamma = 4$	$\gamma = 16$
DAPD	schedule scale $s = 4$	schedule scale $s = 16$
FastGuide	$k = 8$, $\tau = 0.5$, $w = 2$	$k = 16$, $\tau = 0.5$, $w = 2$

Table 2: **Operating points of all parallel decoders.** The high-quality regime trades throughput for quality, and the high-throughput regime trades quality for throughput.

LMUnit Details. The LMUnit score uses a judge LLM to give a score from 1 to 5. Following Tejaswi et al. (2026), we do not treat LMUnit as the primary quality metric. Instead, it provides an independent assessment of response quality to help detect potential reward overoptimization (Gao et al., 2023). We use the prompt in Figure 10 as input to LMUnit, similar to Saad-Falcon et al. (2025) and Tejaswi et al. (2026). Each response is scored on five fixed unit tests, and we report the mean.

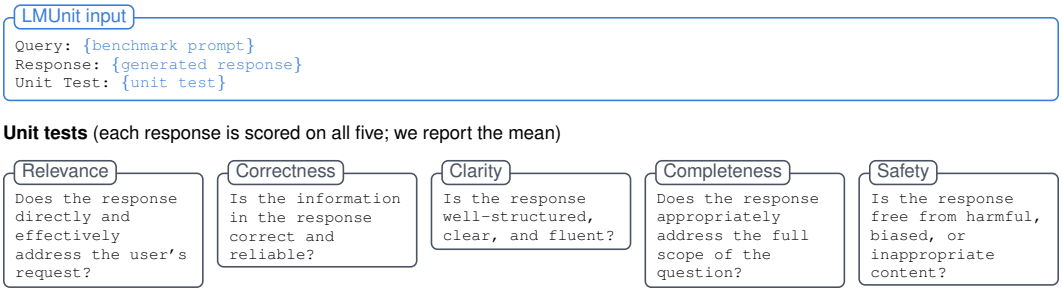


Figure 10: **Prompts given to the LMUnit judge.** We also include in the system message the phrase `rationale=No`, which instructs LMUnit to output only a score for the given unit test without a written rationale as suggested by Saad-Falcon et al. (2025)

B.3 Experimental Details for Section 3

All diagnostic experiments in Section 3 use Dream-7B with the smaller `Skywork-Reward-V2-Qwen3-0.6B` reward model on 250 RM-Bench prompts (50 per domain), with the sampling and guidance settings of Appendix B.2.

Reuse versus recomputation (Figure 3). At every step, the $k \in \{2, 4, 8\}$ most confident positions under the guided distribution at the start of the step are fixed as the candidate set and unmasked in descending confidence. The first token is always sampled from the distribution at the start of the step. For tokens $2, \ldots, k$, the four variants of Equation 5 either reuse or recompute the dLLM logits (with an exact full forward pass) and the guidance vector. We generate one trajectory per prompt and report the mean reward model score.

Guidance reuse (Figure 4a). At every step t and every masked position we form two guided distributions from the current dLLM logits: one with the guidance vector computed at step t and one with the guidance vector computed at step $t - (k - 1)$, which is the oldest guidance that a parallel step of size k would reuse. The curves for $k = 2, 4, 8$ therefore correspond to guidance that is 1, 3, and 7 steps old. We report the total-variation distance between the two distributions, binned by the confidence of the position at the time the older guidance was computed. Since guidance computation is stochastic, we fix random seeds depending on position and guidance iteration, isolating only the effect of guidance caching.

Guidance increases incompatible proposals (Figure 4b). Along a sequential decoding trajectory, we examine 64 evenly spaced states. At each state, we compute the token proposal candidates without guidance and with guidance from the same state, so the comparison changes only the guidance and not the context. For each, we take the two most confident positions a and b with proposed tokens x_a and x_b, which is the pair that a parallel decoder with $k = 2$ would unmask. We then unmask x_a, run one additional unguided dLLM forward pass, and compute $\mathrm{PMI} = \log p(x_b \mid x_a) - \log p(x_b)$ under the dLLM. A pair is counted as incompatible if its PMI is less than -1, that is, if unmasking the first token lowers the dLLM log probability of the second by a factor of e. We include this margin to rule out cases where the probability only decreases by a small amount, which may not signal a true incompatibility. The figure reports, for each prompt, the number of incompatible pairs among its 64 states. Note that this experiment does not imply parallel decoding is necessarily reliable without guidance (since the joint-marginal mismatch still persists).

C FastGuide pseudocode

Algorithm 1 summarizes one hybrid decoding step of FastGuide as described in Section 3.

Algorithm 1 One hybrid decoding step of FastGuide.

Require: State $\boldsymbol{x}_t$, candidate budget k, window radius w, threshold τ
Compute $\ell_{t,0}$ and $\boldsymbol{r}_{t,0}$ with one full guided pass; cache all K/V states
Rank masked positions by decreasing guided confidence to obtain $(a_1, \ldots, a_k)$
Sample a token at a_1 from the guided distribution and commit it; set $a_{\mathrm{prev}} \leftarrow a_1$
for $j = 2, \ldots, k$ **do**
  $\mathcal{W}_j \leftarrow$ windows of radius w around a_{prev} and candidate a_j
  Recompute K/V states at positions in $\mathcal{W}_j$ in every layer and overwrite the corresponding cache entries
  Sparsely refresh $\widehat{\ell}^{a_j}_{t,j-1}$ with queries in $\mathcal{W}_j$ attending to the updated cache
  $\phi^{a_j}_{t,j} \leftarrow \widehat{\ell}^{a_j}_{t,j-1} + \boldsymbol{r}^{a_j}_{t,0}$; sample $c_j \sim \mathrm{softmax}(\phi^{a_j}_{t,j}/T)$
  if $\max_c \mathrm{softmax}(\phi^{a_j}_{t,j})(c) \geq \tau$ **then**
    Commit c_j at a_j; $a_{\mathrm{prev}} \leftarrow a_j$
  end if
end for
return Updated partially masked state

For the Dream text diffusion model (Ye et al., 2025b), logits for candidate a_j are produced at position $a_j - 1$, so we center the attention window at that location. For the LLaDA text diffusion model (Nie et al., 2026), we center the window at a_j.

D Additional Experiments

D.1 Guidance Caching with other Reward Guidance Algorithms

Table 3 shows the performance of FastGuide on other popular reward guidance methods on JudgeBench and RM-Bench. The first is the method of Murata et al. (2024) (titled Expectation), which uses the expected token embedding as input to the reward model. The second is the method of Rout et al. (2025) (titled APS), which samples from the token distribution to generate an input to the reward model. Table 3 demonstrates that FastGuide is robust to the reward guidance algorithm and consistently retains the performance of the corresponding sequential decoding algorithm using the same guidance method.

		Dream-7B			
	Method	**Top@1**	**Seq Gap**	**LMUnit**	**s/gen**
	JudgeBench				
SEQ	Conf	$+2.53_{\pm.16}$	–	$4.05_{\pm.03}$	$37.3_{\pm15.5}$
SEQ	Conf (Expectation)	$+2.61_{\pm.17}$	–	$4.07_{\pm.03}$	$37.3_{\pm16.4}$
SEQ	Conf (APS)	$+2.06_{\pm.16}$	–	$4.04_{\pm.03}$	$37.0_{\pm15.8}$
PARALLEL	Conf	$+0.38_{\pm.20}$	-2.15	$3.64_{\pm.04}$	$9.7_{\pm3.8}$
PARALLEL	Conf (Expectation)	$-0.26_{\pm.20}$	-2.88	$3.53_{\pm.04}$	$9.6_{\pm3.9}$
PARALLEL	Conf (APS)	$-0.75_{\pm.20}$	-2.81	$3.53_{\pm.04}$	$9.7_{\pm3.8}$
PARALLEL	FastGuide	$\mathbf{+2.08}_{\pm.17}$	-0.45	$\mathbf{4.02}_{\pm.03}$	$9.6_{\pm5.0}$
PARALLEL	FastGuide (Expectation)	$+1.99_{\pm.18}$	-0.62	$3.99_{\pm.03}$	$10.5_{\pm5.6}$
PARALLEL	FastGuide (APS)	$+1.68_{\pm.17}$	$\mathbf{-0.38}$	$3.96_{\pm.03}$	$10.3_{\pm5.1}$
	RM-Bench				
SEQ	Conf	$+5.42_{\pm.11}$	–	$4.15_{\pm.02}$	$22.9_{\pm6.9}$
SEQ	Conf (Expectation)	$+5.18_{\pm.11}$	–	$4.14_{\pm.02}$	$22.4_{\pm7.0}$
SEQ	Conf (APS)	$+5.12_{\pm.11}$	–	$4.13_{\pm.02}$	$22.7_{\pm6.9}$
PARALLEL	Conf	$+2.32_{\pm.14}$	-3.09	$3.61_{\pm.02}$	$5.8_{\pm1.7}$
PARALLEL	Conf (Expectation)	$+1.61_{\pm.14}$	-3.56	$3.46_{\pm.02}$	$5.4_{\pm1.8}$
PARALLEL	Conf (APS)	$+1.13_{\pm.14}$	-3.98	$3.45_{\pm.02}$	$5.5_{\pm1.7}$
PARALLEL	FastGuide	$\mathbf{+4.70}_{\pm.12}$	-0.72	$4.04_{\pm.02}$	$6.2_{\pm1.5}$
PARALLEL	FastGuide (Expectation)	$\mathbf{+4.70}_{\pm.12}$	$\mathbf{-0.47}$	$\mathbf{4.05}_{\pm.02}$	$6.5_{\pm2.0}$
PARALLEL	FastGuide (APS)	$+4.25_{\pm.12}$	-0.86	$4.00_{\pm.02}$	$6.6_{\pm2.2}$

Table 3: **Generalization across reward-guidance objectives.** FastGuide stays close to sequential decoding under the Expectation and APS objectives (in parentheses), while confidence-based parallel decoding degrades under all three.

D.2 FastGuide with Different Reward Models

Table 4 tests the performance of FastGuide on different reward models in addition to the Skywork-Qwen3-1.7B reward model tested in Table 1. For the Qwen3-0.6B and Llama-3.1-8B model, FastGuide similarly retains the performance of sequential decoding. Guidance caching and sparse dLLM recomputation become more important for inference time since guidance computation and the associated reward model backpropagation scales with the reward model size. On Skywork-Llama-3.1-8B, inference time decreases from 88.1 seconds to only 17.9 seconds, yielding a $4.9\times$ speedup. Since the Llama-3.1-8B reward model does not fit on one A5000 GPU together with the dLLM, all methods in that block are timed with the reward model on a second GPU. The other two blocks use a single GPU.

	Method	Dream-7B Top@1	Seq Gap	LMUnit	s/gen
JudgeBench, Skywork-Reward-V2-Qwen3-0.6B (smaller)					
SEQ	Conf (BoN)	$+1.72_{\pm.13}$	–	$3.90_{\pm.03}$	$11.3_{\pm5.4}$
SEQ	Conf (Expectation)	$+2.86_{\pm.13}$	–	$4.03_{\pm.03}$	$29.7_{\pm16.8}$
SEQ	Conf (APS)	$+2.46_{\pm.12}$	–	$4.02_{\pm.03}$	$30.0_{\pm16.9}$
SEQ	Conf	$+2.66_{\pm.12}$	–	$3.99_{\pm.03}$	$29.9_{\pm16.3}$
PAR	Conf	$+0.74_{\pm.16}$	-1.92	$3.61_{\pm.04}$	$7.6_{\pm4.1}$
PAR	DAPD	$+2.09_{\pm.14}$	-0.57	$3.90_{\pm.03}$	$7.2_{\pm3.9}$
PAR	FastGuide	$\mathbf{+2.34}_{\pm.13}$	$\mathbf{-0.32}$	$\mathbf{3.97}_{\pm.03}$	$7.5_{\pm3.5}$
JudgeBench, Skywork-Reward-V2-Qwen3-1.7B (Table 1)					
SEQ	Conf (BoN)	$+1.52_{\pm.16}$	–	$3.97_{\pm.03}$	$10.8_{\pm4.7}$
SEQ	Conf (Expectation)	$+2.62_{\pm.17}$	–	$4.06_{\pm.03}$	$37.3_{\pm16.4}$
SEQ	Conf (APS)	$+2.05_{\pm.16}$	–	$4.04_{\pm.03}$	$37.0_{\pm15.8}$
SEQ	Conf	$+2.52_{\pm.16}$	–	$4.04_{\pm.03}$	$37.3_{\pm15.5}$
PAR	Conf	$+0.35_{\pm.21}$	-2.17	$3.63_{\pm.04}$	$9.7_{\pm3.8}$
PAR	DAPD	$+1.89_{\pm.17}$	-0.63	$3.96_{\pm.03}$	$8.6_{\pm3.3}$
PAR	FastGuide	$\mathbf{+2.07}_{\pm.16}$	$\mathbf{-0.45}$	$\mathbf{4.02}_{\pm.03}$	$9.6_{\pm5.0}$
JudgeBench, Skywork-Reward-V2-Llama-3.1-8B (larger)					
SEQ	Conf (BoN)	$+7.28_{\pm.32}$	–	$3.96_{\pm.03}$	$11.2_{\pm5.5}$
SEQ	Conf (Expectation)	$+8.86_{\pm.31}$	–	$4.04_{\pm.03}$	$86.9_{\pm46.5}$
SEQ	Conf (APS)	$+8.68_{\pm.31}$	–	$3.99_{\pm.03}$	$88.2_{\pm46.3}$
SEQ	Conf	$+9.30_{\pm.32}$	–	$4.04_{\pm.03}$	$88.1_{\pm46.0}$
PAR	Conf	$+5.25_{\pm.35}$	-4.05	$3.64_{\pm.04}$	$22.7_{\pm11.4}$
PAR	DAPD	$+7.49_{\pm.36}$	-1.81	$3.87_{\pm.03}$	$19.2_{\pm10.0}$
PAR	FastGuide	$\mathbf{+8.29}_{\pm.31}$	$\mathbf{-1.01}$	$\mathbf{3.94}_{\pm.03}$	$17.9_{\pm9.7}$

Table 4: **Robustness to the reward model on JudgeBench.**

D.3 Measuring Average Reward Model Performance and Throughput

In this section, we evaluate the Avg@4 metric, which is the average reward model performance over 4 trajectories, as well as show the average throughput (tokens unmasked per step) for all methods. Table 5 and Table 7 show these for Dream and LLaDA text diffusion models respectively. Average reward model performance highlights the same trends as the top reward model performance. On both models and all three benchmarks, FastGuide attains the highest Avg@4 among parallel decoders and stays within 1.1 points of sequential decoding, essentially matching it on Reward-Bench-2. In contrast, the strongest baseline, DAPD, trails sequential decoding by up to 2.14, and every other guided baseline by at least 2.4. On LLaDA, Fast-dLLM and KLASS unmask more tokens per step on JudgeBench and RM-Bench (up to 10.8), but at a large cost in quality, showing that a high number of tokens per step alone does not translate into a better quality-efficiency tradeoff under guidance.

Additionally, we consider varying throughput by increasing k for FastGuide to 16 candidate tokens per step, showing results on Dream diffusion model in Table 6. Baseline hyperparameters were adjusted accordingly (see Appendix B.2). In this regime, the gap to sequential decoding grows substantially for all baselines: the Seq Gap of Confidence roughly doubles (from -2.15 to -3.09 in Table 5 to -4.26 to -5.95), and that of DAPD, the strongest baseline, grows from at most -1.83 to as much as -4.07. Baselines do become faster (2.6–4.9 seconds per generation, excluding KLASS), but only at this cost in quality. In contrast, the Seq Gap of FastGuide remains within 1.20 on all three benchmarks. Notably, although FastGuide now considers 16 candidates per step, its throughput only increases from 4.4–5.7 to 5.5–8.0 tokens per step, and its inference time is essentially unchanged (6.2–9.6 versus 5.9–8.4 seconds per generation). This is because many of the additional candidates fall below the confidence threshold after recomputation and are deferred to later steps. This highlights the benefit of adaptivity. Rather than trading quality for speed at a fixed rate, FastGuide unmasks additional tokens only when they remain compatible with the tokens already unmasked in the same step.

	Method	Dream-7B Top@1	Seq Gap	Avg@4	LMUnit	tok/step	s/gen
				JudgeBench			
SEQ	Conf (BoN)	$+1.52_{\pm.16}$	–	$-1.07_{\pm.15}$	$3.97_{\pm.03}$	1.00	$10.8_{\pm4.7}$
SEQ	Conf	$+2.53_{\pm.16}$	–	$+0.08_{\pm.15}$	$4.05_{\pm.03}$	1.00	$37.3_{\pm15.5}$
PARALLEL	Conf (BoN)	$-3.14_{\pm.20}$	-4.66	$-6.04_{\pm.17}$	$3.01_{\pm.05}$	4.00	$2.7_{\pm1.1}$
PARALLEL	Conf	$+0.38_{\pm.20}$	-2.15	$-2.64_{\pm.19}$	$3.64_{\pm.04}$	4.00	$9.7_{\pm3.8}$
PARALLEL	Fast-dLLM	$+0.26_{\pm.21}$	-2.27	$-2.66_{\pm.20}$	$3.62_{\pm.05}$	5.16	$7.9_{\pm4.1}$
PARALLEL	KLASS	$-0.06_{\pm.22}$	-2.59	$-2.89_{\pm.21}$	$3.61_{\pm.05}$	5.10	$10.8_{\pm12.5}$
PARALLEL	EB-Sampler	$-0.49_{\pm.19}$	-3.02	$-3.34_{\pm.17}$	$3.49_{\pm.04}$	4.60	$9.4_{\pm3.5}$
PARALLEL	DAPD	$+1.92_{\pm.17}$	-0.61	$-0.78_{\pm.18}$	$3.95_{\pm.03}$	4.41	$8.6_{\pm3.3}$
PARALLEL	FastGuide	$\mathbf{+2.08}_{\pm.17}$	$\mathbf{-0.45}$	$\mathbf{-0.47}_{\pm.16}$	$\mathbf{4.02}_{\pm.03}$	5.67	$9.6_{\pm5.0}$
				RM-Bench			
SEQ	Conf (BoN)	$+4.41_{\pm.12}$	–	$+1.76_{\pm.13}$	$4.06_{\pm.02}$	1.00	$6.3_{\pm1.9}$
SEQ	Conf	$+5.42_{\pm.11}$	–	$+3.05_{\pm.12}$	$4.15_{\pm.02}$	1.00	$22.9_{\pm6.9}$
PARALLEL	Conf (BoN)	$-1.96_{\pm.14}$	-6.37	$-4.71_{\pm.13}$	$2.75_{\pm.02}$	4.00	$1.6_{\pm.5}$
PARALLEL	Conf	$+2.32_{\pm.14}$	-3.09	$-0.83_{\pm.15}$	$3.61_{\pm.02}$	4.00	$5.8_{\pm1.7}$
PARALLEL	Fast-dLLM	$+1.34_{\pm.15}$	-4.08	$-1.36_{\pm.15}$	$3.51_{\pm.03}$	5.27	$5.6_{\pm2.2}$
PARALLEL	KLASS	$+0.38_{\pm.15}$	-5.04	$-2.18_{\pm.14}$	$3.33_{\pm.03}$	5.28	$9.6_{\pm9.9}$
PARALLEL	EB-Sampler	$+0.40_{\pm.15}$	-5.01	$-2.15_{\pm.15}$	$3.20_{\pm.02}$	4.38	$5.9_{\pm2.0}$
PARALLEL	DAPD	$+3.58_{\pm.13}$	-1.83	$+0.91_{\pm.14}$	$3.87_{\pm.02}$	4.34	$5.9_{\pm1.3}$
PARALLEL	FastGuide	$\mathbf{+4.70}_{\pm.12}$	$\mathbf{-0.72}$	$\mathbf{+2.11}_{\pm.13}$	$\mathbf{4.04}_{\pm.02}$	5.43	$6.2_{\pm1.5}$
				Reward-Bench-2			
SEQ	Conf (BoN)	$+2.60_{\pm.10}$	–	$+0.69_{\pm.10}$	$4.18_{\pm.01}$	1.00	$5.5_{\pm1.4}$
SEQ	Conf	$+3.39_{\pm.10}$	–	$+1.64_{\pm.10}$	$4.23_{\pm.01}$	1.00	$20.4_{\pm5.2}$
PARALLEL	Conf (BoN)	$-1.98_{\pm.09}$	-4.58	$-4.33_{\pm.09}$	$2.95_{\pm.02}$	4.00	$1.4_{\pm.4}$
PARALLEL	Conf	$+1.24_{\pm.10}$	-2.15	$-1.27_{\pm.10}$	$3.78_{\pm.02}$	4.00	$5.2_{\pm1.2}$
PARALLEL	Fast-dLLM	$+1.36_{\pm.11}$	-2.03	$-0.94_{\pm.11}$	$3.82_{\pm.02}$	4.16	$6.8_{\pm2.5}$
PARALLEL	KLASS	$+1.06_{\pm.10}$	-2.33	$-1.14_{\pm.10}$	$3.83_{\pm.02}$	3.48	$15.7_{\pm13.3}$
PARALLEL	EB-Sampler	$+0.56_{\pm.10}$	-2.83	$-1.64_{\pm.10}$	$3.57_{\pm.02}$	3.52	$6.6_{\pm2.0}$
PARALLEL	DAPD	$+3.01_{\pm.10}$	-0.38	$+1.07_{\pm.10}$	$4.15_{\pm.01}$	3.86	$6.2_{\pm1.7}$
PARALLEL	FastGuide	$\mathbf{+3.25}_{\pm.10}$	$\mathbf{-0.14}$	$\mathbf{+1.43}_{\pm.10}$	$\mathbf{4.19}_{\pm.01}$	4.40	$7.3_{\pm2.1}$

Table 5: **Full results on Dream-7B with Avg@4 and tokens/step added.**

	Method	Dream-7B Top@1	Seq Gap	LMUnit	tok/step	s/gen
	JudgeBench					
SEQ	Conf (BoN)	$+1.52_{\pm.16}$	–	$3.97_{\pm.03}$	1.00	$10.8_{\pm4.7}$
SEQ	Conf	$+2.53_{\pm.16}$	–	$4.05_{\pm.03}$	1.00	$37.3_{\pm15.5}$
PARALLEL	Conf	$-1.91_{\pm.20}$	-4.45	$3.04_{\pm.05}$	8.00	$4.8_{\pm1.9}$
PARALLEL	Fast-dLLM	$-2.13_{\pm.23}$	-4.66	$3.04_{\pm.06}$	8.63	$4.9_{\pm2.5}$
PARALLEL	KLASS	$-2.29_{\pm.23}$	-4.82	$3.00_{\pm.06}$	7.83	$8.3_{\pm10.5}$
PARALLEL	EB-Sampler	$-3.27_{\pm.16}$	-5.80	$2.60_{\pm.04}$	9.50	$4.5_{\pm1.6}$
PARALLEL	DAPD	$+0.26_{\pm.20}$	-2.27	$3.67_{\pm.04}$	8.46	$4.8_{\pm1.7}$
PARALLEL	FastGuide	$\mathbf{+1.84}_{\pm.18}$	$\mathbf{-0.69}$	$\mathbf{3.96}_{\pm.03}$	8.02	$8.4_{\pm3.9}$
	RM-Bench					
SEQ	Conf (BoN)	$+4.41_{\pm.12}$	–	$4.06_{\pm.02}$	1.00	$6.3_{\pm1.9}$
SEQ	Conf	$+5.42_{\pm.11}$	–	$4.15_{\pm.02}$	1.00	$22.9_{\pm6.9}$
PARALLEL	Conf	$-0.54_{\pm.16}$	-5.95	$2.90_{\pm.03}$	8.00	$2.9_{\pm.9}$
PARALLEL	Fast-dLLM	$-1.20_{\pm.16}$	-6.62	$2.83_{\pm.03}$	8.99	$3.4_{\pm1.6}$
PARALLEL	KLASS	$-1.63_{\pm.16}$	-7.05	$2.75_{\pm.03}$	8.29	$7.4_{\pm10.9}$
PARALLEL	EB-Sampler	$-2.48_{\pm.15}$	-7.90	$2.31_{\pm.02}$	9.27	$2.8_{\pm.9}$
PARALLEL	DAPD	$+1.34_{\pm.16}$	-4.07	$3.51_{\pm.02}$	7.77	$3.6_{\pm.7}$
PARALLEL	FastGuide	$\mathbf{+4.21}_{\pm.12}$	$\mathbf{-1.20}$	$\mathbf{3.99}_{\pm.02}$	7.65	$5.9_{\pm1.7}$
	Reward-Bench-2					
SEQ	Conf (BoN)	$+2.60_{\pm.10}$	–	$4.18_{\pm.01}$	1.00	$5.5_{\pm1.4}$
SEQ	Conf	$+3.39_{\pm.10}$	–	$4.23_{\pm.01}$	1.00	$20.4_{\pm5.2}$
PARALLEL	Conf	$-0.87_{\pm.11}$	-4.26	$3.08_{\pm.02}$	8.00	$2.6_{\pm.6}$
PARALLEL	Fast-dLLM	$-0.49_{\pm.12}$	-3.88	$3.25_{\pm.02}$	7.46	$4.5_{\pm2.0}$
PARALLEL	KLASS	$-0.67_{\pm.11}$	-4.06	$3.30_{\pm.02}$	5.96	$13.5_{\pm13.4}$
PARALLEL	EB-Sampler	$-1.99_{\pm.10}$	-5.38	$2.61_{\pm.02}$	8.22	$2.9_{\pm.8}$
PARALLEL	DAPD	$+1.71_{\pm.10}$	-1.68	$3.93_{\pm.02}$	6.84	$3.8_{\pm1.0}$
PARALLEL	FastGuide	$\mathbf{+3.15}_{\pm.10}$	$\mathbf{-0.24}$	$\mathbf{4.18}_{\pm.01}$	5.54	$8.2_{\pm2.6}$

Table 6: **Results on Dream-7B with higher candidate token set size of $k = 16$ for FastGuide.**

	Method	Top@1	Seq Gap	Avg@4	LMUnit	tok/step	s/gen
				LLaDA-8B			
				JudgeBench			
SEQ	Conf (BoN)	$+2.65_{\pm.21}$	–	$+0.51_{\pm.20}$	$4.05_{\pm.03}$	1.00	$11.8_{\pm5.1}$
SEQ	Conf	$+3.23_{\pm.20}$	–	$+1.31_{\pm.20}$	$4.08_{\pm.03}$	1.00	$45.3_{\pm18.5}$
PARALLEL	Conf (BoN)	$-2.23_{\pm.22}$	-4.88	$-4.87_{\pm.19}$	$3.05_{\pm.05}$	4.00	$3.0_{\pm1.3}$
PARALLEL	Conf	$+1.54_{\pm.22}$	-1.69	$-1.17_{\pm.21}$	$3.78_{\pm.04}$	4.00	$11.8_{\pm5.1}$
PARALLEL	Fast-dLLM	$-0.12_{\pm.21}$	-3.35	$-2.58_{\pm.19}$	$3.39_{\pm.05}$	9.17	$6.5_{\pm2.9}$
PARALLEL	KLASS	$-0.81_{\pm.19}$	-4.04	$-3.14_{\pm.17}$	$3.10_{\pm.05}$	10.81	$5.6_{\pm2.7}$
PARALLEL	EB-Sampler	$+0.09_{\pm.21}$	-3.14	$-2.40_{\pm.20}$	$3.52_{\pm.04}$	5.34	$10.1_{\pm3.8}$
PARALLEL	DAPD	$+2.74_{\pm.22}$	-0.49	$+0.46_{\pm.21}$	$3.99_{\pm.04}$	5.06	$9.4_{\pm4.2}$
PARALLEL	FastGuide	$\mathbf{+2.76}_{\pm.21}$	$\mathbf{-0.47}$	$\mathbf{+0.64}_{\pm.21}$	$\mathbf{4.03}_{\pm.03}$	6.63	$10.3_{\pm3.4}$
				RM-Bench			
SEQ	Conf (BoN)	$+4.39_{\pm.12}$	–	$+1.89_{\pm.13}$	$3.94_{\pm.02}$	1.00	$6.6_{\pm2.2}$
SEQ	Conf	$+5.25_{\pm.11}$	–	$+3.05_{\pm.12}$	$4.02_{\pm.02}$	1.00	$24.9_{\pm8.8}$
PARALLEL	Conf (BoN)	$-1.57_{\pm.14}$	-5.96	$-4.41_{\pm.12}$	$2.79_{\pm.02}$	4.00	$1.7_{\pm.5}$
PARALLEL	Conf	$+2.83_{\pm.13}$	-2.43	$-0.24_{\pm.13}$	$3.63_{\pm.02}$	4.00	$6.3_{\pm2.2}$
PARALLEL	Fast-dLLM	$+0.74_{\pm.14}$	-4.51	$-2.01_{\pm.13}$	$3.31_{\pm.03}$	6.86	$4.5_{\pm1.7}$
PARALLEL	KLASS	$-0.11_{\pm.14}$	-5.36	$-2.64_{\pm.13}$	$3.03_{\pm.03}$	8.13	$4.1_{\pm1.9}$
PARALLEL	EB-Sampler	$+0.49_{\pm.15}$	-4.76	$-2.07_{\pm.14}$	$3.19_{\pm.02}$	5.05	$5.9_{\pm1.6}$
PARALLEL	DAPD	$+4.36_{\pm.12}$	-0.89	$+1.77_{\pm.13}$	$3.90_{\pm.02}$	4.54	$6.2_{\pm2.3}$
PARALLEL	FastGuide	$\mathbf{+4.53}_{\pm.12}$	$\mathbf{-0.72}$	$\mathbf{+1.95}_{\pm.13}$	$\mathbf{3.92}_{\pm.02}$	6.07	$6.5_{\pm1.8}$
				Reward-Bench-2			
SEQ	Conf (BoN)	$+2.19_{\pm.09}$	–	$+0.13_{\pm.09}$	$4.00_{\pm.02}$	1.00	$6.1_{\pm1.6}$
SEQ	Conf	$+2.88_{\pm.09}$	–	$+0.98_{\pm.09}$	$4.05_{\pm.02}$	1.00	$26.9_{\pm9.3}$
PARALLEL	Conf (BoN)	$-2.22_{\pm.08}$	-4.41	$-4.63_{\pm.07}$	$2.88_{\pm.02}$	4.00	$1.5_{\pm.4}$
PARALLEL	Conf	$+0.98_{\pm.09}$	-1.90	$-1.50_{\pm.09}$	$3.67_{\pm.02}$	4.00	$6.6_{\pm2.3}$
PARALLEL	Fast-dLLM	$+0.05_{\pm.09}$	-2.83	$-2.32_{\pm.09}$	$3.55_{\pm.02}$	5.05	$7.9_{\pm4.0}$
PARALLEL	KLASS	$-0.09_{\pm.09}$	-2.97	$-2.38_{\pm.09}$	$3.44_{\pm.02}$	5.63	$8.0_{\pm4.4}$
PARALLEL	EB-Sampler	$+0.01_{\pm.09}$	-2.87	$-2.34_{\pm.08}$	$3.40_{\pm.02}$	4.22	$9.0_{\pm3.1}$
PARALLEL	DAPD	$+2.49_{\pm.09}$	-0.39	$+0.62_{\pm.09}$	$\mathbf{4.00}_{\pm.02}$	4.01	$7.2_{\pm2.2}$
PARALLEL	FastGuide	$\mathbf{+2.58}_{\pm.09}$	$\mathbf{-0.30}$	$\mathbf{+0.64}_{\pm.09}$	$\mathbf{4.00}_{\pm.02}$	5.27	$7.3_{\pm2.5}$

Table 7: **Full results on LLaDA-8B with Avg@4 and tokens/step added.**

D.4 Effect of Batching

The standard dLLM forward pass uses bidirectional attention, making it a largely compute-bound operation on modern GPUs (Kim et al., 2025a; Liang et al., 2026). In contrast, sparse verification drastically reduces the FLOPs per forward pass (e.g. with a window size of 2 and sequence length of 128, sparse forward pass is only 4% FLOPs of a full forward pass). Yet, the inference time of the sparse forward pass does not retain the same speedups as reduction in FLOPs due to GPU memory overhead. Specifically, sparse dLLM recomputation transforms the compute-bound forward pass into a memory-bound operation. Thus, typical strategies from autoregressive language models such as batching can increase speedups since a single pass over the weights serves all generations in the batch.

We study how batching affects the inference time of FastGuide by decoding $B \in \{1, 2, 3, 4\}$ generations of the same prompt together (over 50 RM-Bench prompts) and reporting the per-generation wall-clock time. Figure 11 shows that as batch size increases, sparse recomputation improves more upon exact recomputation and speedups increase. FastGuide improves from 15.3 to 5.7 seconds per generation as B grows from 1 to 4 ($2.7\times$), whereas its exact recomputation counterpart improves only from 18.5 to 10.8 seconds ($1.7\times$). Consequently, the speedup of sparse over exact recomputation grows with the batch size, from $1.21\times$ at $B{=}1$ to $1.89\times$ at $B{=}4$, and the speedup of FastGuide over sequential guided decoding grows from $2.9\times$ to $3.4\times$. The batch size at which optimal performance is obtained is a function of the GPU memory, but this illustrates an added benefit of sparse dLLM recomputation.

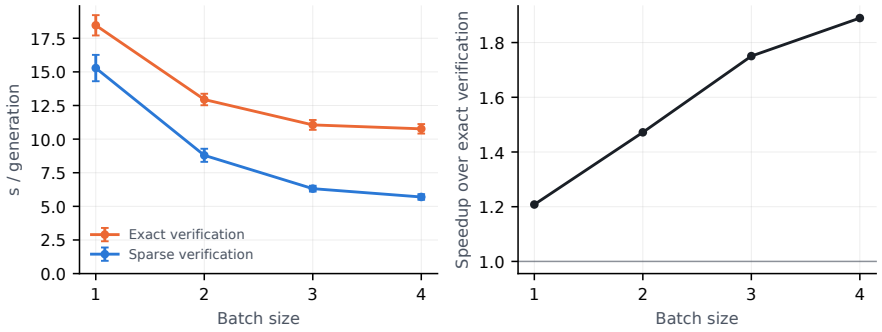


Figure 11: **Sparse dLLM verification benefits more from batching than exact recomputation.** Left: inference time per generation; right: speedup of sparse over exact recomputation.

D.5 Sparse dLLM Verification Window Size Ablation

While Figure 9 shows the TV distance between sparse and exact recomputation, we now measure the impact on generation quality through the Top@1 reward model score. We sweep the window size of sparse recomputation in Figure 12. The figure illustrates that even a small window size of 2 recovers a significant proportion of generation quality of sequential guided decoding. The dashed lines labeled Exact show exact dLLM recomputation with a full forward pass at the same k, further highlighting that sparse recomputation is a useful approximation of exact dLLM recomputation. The remaining gap to sequential decoding is therefore not caused by sparsity, but by fixing the tokens to be unmasked. Confidence deferral aims to address this problem by adding adaptivity into the sampling procedure (Figure 6).

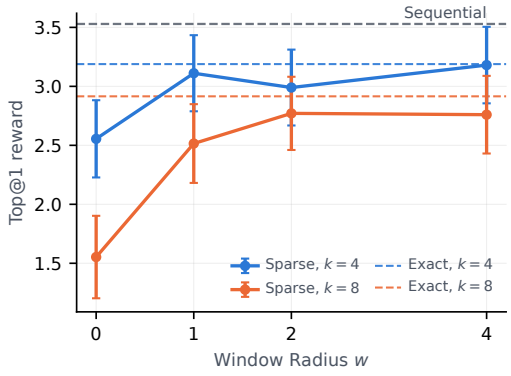


Figure 12: **Quality versus recomputation window radius w on RM-Bench.** We evaluate sparse dLLM recomputation as a function of window radius w. Here, we remove confidence deferral so all k candidates per step are unmasked.

Verification order	Dream-7B Top@1
RM-Bench	
Left → right	$+5.62_{\pm.51}$
High → low confidence	$+\mathbf{5.63}_{\pm.51}$
Low → high confidence	$+5.61_{\pm.50}$
Random	$+5.60_{\pm.48}$

Table 8: **Verification order ablation on RM-Bench (k=4).** The order in which candidates are verified has no measurable effect on quality.

D.6 Verification Order Ablation

Autoregressive dLLM verification relies on an ordering of the candidates within each hybrid decoding step (Section 3). In the main paper, we fix this ordering to be from high to low confidence. We study other ordering schemes in Table 8. We remove confidence deferral and unmask exactly $k = 4$ tokens each step to isolate the role of ordering. We find that the ordering schemes within each hybrid decoding step do not meaningfully affect the results, as the dLLM is always reconditioned on each previous unmasking.

D.7 Effect of Greedy Token Selection

FastGuide samples every unmasked token at temperature 0.7, similar to the setting for all baseline methods (Appendix B.2). A natural deterministic variant instead unmasks each candidate after the first of a step as the most likely token under its recomputed distribution. Table 9 compares the two on the full RM-Bench test set. Greedy selection improves Top@1 by 0.30 and LMUnit by 0.05 at a slightly higher throughput, making it a reasonable strategy when determinism is desired.

Token selection in FastGuide	Top@1	LMUnit	tok/step	s/gen
Sampled at temperature 0.7 (default)	$+4.70_{\pm.12}$	$4.04_{\pm.02}$	5.43	$6.2_{\pm1.5}$
Most likely token	$+5.00_{\pm.12}$	$4.09_{\pm.02}$	5.59	$6.1_{\pm1.6}$

Table 9: **Greedy selection of recomputed tokens slightly improves FastGuide.** RM-Bench, Dream-7B, $k = 8$

D.8 Extended Computational Analysis

Table 10 reports peak GPU memory and teraFLOPs per generation on RM-Bench. Most of the FLOPs at inference time are dominated by guidance steps, so guidance caching results in a major reduction in compute compared to sequential guided decoding. FastGuide further uses fewer FLOPs than other methods because sparse dLLM recomputation lets FastGuide unmask more tokens in each step, and each sparse recomputation pass uses far less compute than a full forward pass. In the high-throughput regime ($k = 16$ for FastGuide), more tokens are deferred than baselines such as DAPD, resulting in slightly increased compute. Across all methods, FastGuide does not increase the peak memory usage, which is dominated by loading of diffusion model weights and reward model weights.

In terms of inference time, Table 11 shows the breakdown of time in full dLLM forwards (computed once at the start of each hybrid decoding step), guidance computation (computed once at the start

of each hybrid decoding step), and sparse dLLM forwards (computed $k-1$ times for every hybrid decoding step).

	Method	Dream-7B VRAM (GB)	Dream-7B TFLOPs/gen
SEQ	Conf (BoN)	20.1	383
SEQ	Conf (Expectation)	21.3	1,056
SEQ	Conf (APS)	21.3	1,053
SEQ	Conf	21.3	1,055
PARALLEL HIGH-QUALITY	Conf (BoN)	20.1	96
PARALLEL HIGH-QUALITY	Conf	21.3	265
PARALLEL HIGH-QUALITY	Fast-dLLM	21.6	222
PARALLEL HIGH-QUALITY	KLASS	21.9	312
PARALLEL HIGH-QUALITY	EB-Sampler	21.6	253
PARALLEL HIGH-QUALITY	DAPD	21.4	249
PARALLEL HIGH-QUALITY	FastGuide	21.4	216
PARALLEL HIGH-THROUGHPUT	Conf	21.3	133
PARALLEL HIGH-THROUGHPUT	Fast-dLLM	21.6	136
PARALLEL HIGH-THROUGHPUT	KLASS	21.9	230
PARALLEL HIGH-THROUGHPUT	EB-Sampler	21.6	122
PARALLEL HIGH-THROUGHPUT	DAPD	21.4	146
PARALLEL HIGH-THROUGHPUT	FastGuide	21.4	179

Table 10: **Computational profile on RM-Bench.** Peak memory and total compute per generation for every method in both regimes of Table 2.

(s/gen)	Sequential	FastGuide (k=8) w/o deferral	FastGuide (k=8) w/ deferral
dLLM forwards	$4.92_{\pm.52}$	$0.62_{\pm.07}$	$1.05_{\pm.15}$
Sparse recomputation	–	$0.92_{\pm.01}$	$1.64_{\pm.30}$
Guidance	$13.60_{\pm1.13}$	$1.70_{\pm.15}$	$3.12_{\pm.49}$
Total	$18.85_{\pm1.62}$	$3.31_{\pm.22}$	$5.94_{\pm.93}$

Table 11: **Inference-time breakdown on RM-Bench.** Wall-clock time per generation at batch size 4 on Dream-7B.

D.9 QUALITATIVE EXAMPLES

Decoding sequence for Figure 8. Figure 13 shows the denoising trajectory for the three methods of Figure 8. While unguided generation with DAPD can produce coherent text, DAPD with guidance can produce incoherent text early on in generation. This reinforces **O2** that guidance can introduce incompatibilities during sampling.

Qualitative generations vs DAPD with guidance. As DAPD (Kim et al., 2026a) remains the strongest baseline in terms of quantitative results (Table 1), we compare the quality of generations across decoding trajectories in Figures 14 and 15. DAPD tends to pick tokens spatially far away as it uses dLLM attention as a signal of conditional dependency. However, guidance can affect this dependency, which may explain why DAPD with guidance tends to introduce incompatible tokens. For example, in Figure 14, certain token sequences such as `5. emonial` are not coherent English sentences. In contrast, FastGuide retains coherency and fluency in generation, obtaining a significantly higher reward model score. In Figure 15, DAPD introduces incorrect numbering (such as the 7th step repeated twice).

Qualitative effect of confidence deferral. Figures 16, 17, and 18 show the effect of confidence deferral at one intermediate step for three different prompts. For instance, in Figure 16, the word mindful is proposed three times, whereas after it is unmasked once, the probability of the other token values drop, leading them to be deferred. The final generations are thus of higher quality, trading off inference time for quality.

Prompt: If the endpoints of a line segment are (2, -2) and (10, 4), what is the length of the segment?



Figure 13: **Guided DAPD unmasks an incorrect final answer while most of its derivation is still masked.** Decoding trajectories for the prompt of Figure 8; rows show the sequence after 25%, 50%, 75%, 100% of each method's decoding steps, where <M> denotes a masked position.

Prompt: What are different drawers I should have for clothes?


DAPD, guided Reward −0.61
FastGuide Reward +1.19
25%
<M> are <M><M> for <M><M><M><

Prompt: What is a good way of landing a knockout punch in boxing?

	DAPD, guided Reward +0.86	FastGuide Reward +2.59
25%	<M>anding <M><M><M><M><M><M><M><M><M><M><M><M><M><M> . <M><M><M><M><M><M><M><M><M> technique, <M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M> speed and <M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><

Prompt: Hi, I'd like to make my own peanut brittle. Can you give me a recipe and cooking instructions for that?

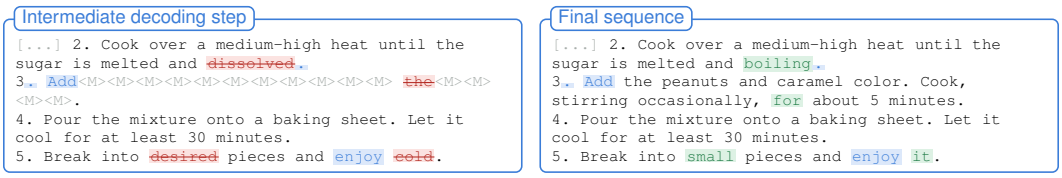


Figure 17: **Deferral lets low-confidence words be revised once their context is unmasked.** The proposals "dissolved", "the", "desired", and "cold" are deferred and later become "boiling", "for", "small", and "it" (left: unmasked and ~~deferred~~ proposals of one step with $k = 8$; right: final values of the deferred positions).

Prompt: How is oil turned into gasoline?

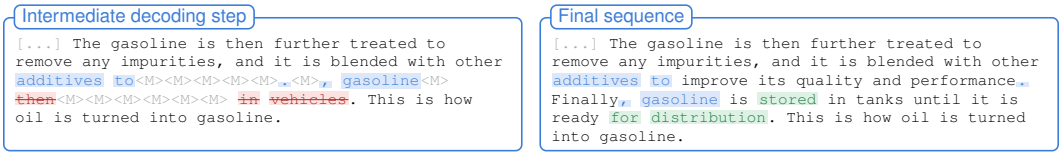


Figure 18: **Deferral revises a sentence ending that was proposed too early.** The low-confidence proposals "then", "in", and "vehicles" are deferred and later become "stored", "for", and "distribution" (left: unmasked and ~~deferred~~ proposals of one step with $k = 8$; right: final values of the deferred positions).