Mean-Field Parallel Decoding for Discrete Diffusion Language Models

Tamim Zoabi
School of Electrical & Computer Engineering
Tel Aviv University, Israel
tamimzoabi@mail.tau.ac.il

Ameen Ali
School of Computer Science and AI
Tel Aviv University, Israel
ameenali@mail.tau.ac.il

Liran Ringel
Department of Computer Science
Technion, Israel Institute of Technology, Israel
liranringel@cs.technion.ac.il

Lior Wolf
School of Computer Science and AI
Tel Aviv University, Israel
wolf@cs.tau.ac.il

Abstract

Discrete diffusion language models enable parallel token generation, offering a pathway to low-latency decoding. However, selecting tokens independently by marginal confidence limits effective parallelism: tokens that appear reliable in isolation can form incompatible configurations when several positions are updated at once. We introduce a training-free decoding framework that coordinates these parallel updates. At each forward pass, the method assigns a commit score to each masked position and refines these scores using pairwise interactions derived from the model's predictive distributions. A variational relaxation yields a simple fixed-point update that suppresses conflicting simultaneous commitments within a single forward pass. This mechanism allows the decoder to commit more tokens in parallel while maintaining competitive generation quality. The method is lightweight, requires no auxiliary model or retraining, and drops into existing diffusion decoding pipelines without modification. Experiments on reasoning and code-generation benchmarks show consistent improvements in the quality-latency trade-off.

1 Introduction

Large-scale language models [Achiam et al., 2023, Bai et al., 2023, Comanici et al., 2025, Liu et al., 2024] have traditionally relied on autoregressive (AR) decoding, which generates tokens sequentially from left to right. While AR models achieve state-of-the-art performance across a wide range of tasks, their sequential nature imposes a fundamental bottleneck on inference latency, as the generation of each token requires a full forward pass of the model. To overcome this limitation, non-autoregressive and semi-autoregressive models have emerged, aiming to generate multiple tokens in parallel to improve throughput and reduce wall-clock time.

Among these approaches, Discrete Diffusion Language Models (dLLMs) [Austin et al., 2021a, Li et al., 2022, Zhu et al., 2025, Nie et al., 2025, Bie et al., 2026] have gained significant attention. By treating text generation as an iterative denoising process, dLLMs allow for flexible decoding schedules where multiple masked positions can be unmasked simultaneously. However, aggressive parallelism in dLLMs introduces a critical challenge: "joint inconsistency" [Ringel et al., 2026, Luo et al., 2026, Zhang et al., 2026]. Most existing dLLMs employ a factorized reverse policy that predicts token distributions for each masked token independently [Ye et al., 2025, Zhu et al., 2025]. While this independence assumption simplifies the model architecture, it fails to capture the complex linguistic dependencies, such as syntactic agreement and semantic coherence, that exist between

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

tokens. Consequently, when multiple tokens are committed in parallel based solely on their marginal confidences, the resulting sequence often suffers from "joint mismatch," where tokens are individually plausible but mutually incompatible.

Several recent approaches attempt to reduce this inconsistency by modifying either the decoding procedure or the underlying diffusion formulation. Dependency-aware inference methods such as DAWN [Luo et al., 2026], the closest prior work to ours, estimate inter-token dependencies from attention and use them to schedule safer parallel commitments, while related analyses of diffusion decoding show that marginally confident tokens can still conflict when committed jointly [Ringel et al., 2026]. Another line of work, including semi-autoregressive and block diffusion models, constrains the generation process by decoding autoregressively at the block level while refining tokens within each block in parallel [Bie et al., 2025, 2026]. This blockwise formulation improves stability relative to fully unconstrained parallel decoding, but it does not by itself resolve the dependency structure among tokens committed within the same block. Other strategies rely on auxiliary verification, repeated influence estimation, or training objectives that encourage stronger joint modeling, however, these methods can introduce additional computation, require extra model components, or depend on retraining the base model. These limitations motivate a lightweight, training-free mechanism that estimates token interactions during inference and uses them to regulate which positions can be safely committed together.

In this paper, we present a single-pass decoding framework, **Mean-Field Parallel Decoding (MPD)**. As illustrated in Figure 1, we formulate parallel token commitment as a structured inference problem over binary variables, moving away from independent marginal-based selection (Figure 1, left). We construct pairwise interactions defined on masked positions directly in the model's output space using the Jensen-Shannon divergence (JSD) between predictive distributions (Figure 1, center). This signal does not recover the full joint conditional, it targets a narrower source of unsafe parallel commits, which we call *predictive redundancy and counterfactual coupling*. By deriving a variational mean-field relaxation, we obtain a fixed-point update in which high-confidence positions modulate others through these interactions. This induces interaction-aware commit decisions, enabling increased parallelism while limiting conflicting commitments (Figure 1, right), all within a single forward pass and without auxiliary models or retraining.

Our contributions are summarized as follows:

(1) We formalize the parallel commit selection problem in dLLMs as a structured inference task over binary commit variables, connecting iterative denoising with energy-based models. (2) We introduce a training-free, gradient-free interaction measure based on distributional overlap (JSD) that requires only a single forward pass and no auxiliary models. (3) We derive an efficient mean-field iteration whose commit decisions empirically stabilize after two iterations, allowing adaptive, interaction-aware parallelism that scales with the model's local certainty. (4) We show on three dLLM backbones and four reasoning and code benchmarks that MPD attains the highest throughput among the compared decoders in every reported setting with competitive accuracy, shifting the quality-latency Pareto frontier rather than uniformly improving accuracy, controlled experiments show that the gain is not explained by the choice of interaction signal or by block size alone.

2 Related Work

Discrete diffusion language models (dLLMs) extend denoising diffusion to discrete token spaces, enabling iterative refinement and parallel decoding [Austin et al., 2021a, Li et al., 2022, Hoogeboom et al., 2021]. Recent large-scale instantiations (e.g., LLaDA, Dream) [Ye et al., 2025, Nie et al., 2025, Bie et al., 2025, 2026] demonstrate competitive performance with flexible decoding schedules while maintaining a factorized reverse parameterization over token positions.

A central direction focuses on accelerating generation via parallel token commitment. In autoregressive models, speculative decoding and its extensions [Leviathan et al., 2023, Chen et al., 2023, Kim et al., 2025b, Chen et al., 2026, Ringel and Romano, 2026] generate multiple tokens and verify them with a target model. In diffusion-based decoding, token selection is typically guided by local uncertainty metrics such as entropy or margin [Ghazvininejad et al., 2019, Ye et al., 2025, Nie et al., 2025]. More recent methods, including KLASS [Kim et al., 2025a], which uses a KL-based selection criterion, and LocalLeap [Kong et al., 2025], which relaxes the commit criterion near high-confidence anchors, improve throughput by prioritizing high-confidence positions. These approaches rely on

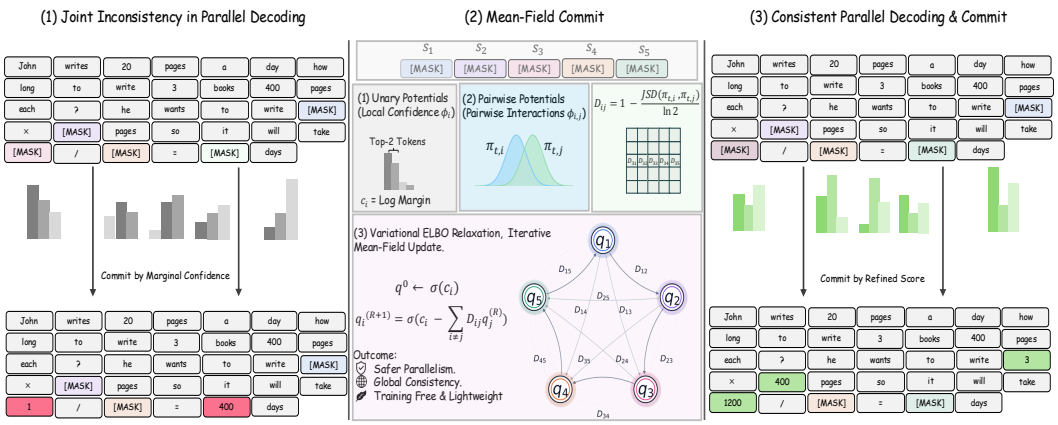


Figure 1: **Mean-Field Parallel Decoding Framework.** (Left) Joint Inconsistency: Standard parallel decoding commits tokens based on position-wise confidence (Top-3 confidence shown). This ignores inter-token dependencies, leading to an incompatible sequence (e.g., repeating the token "3"). (Center) Mean-Field Commit: Our framework frames joint token commitment as a variational mean-field optimization. It computes local margins (c_i) and pairwise similarities (D_{ij} using Jensen-Shannon Divergence), then iterates ($R = 2$) to update commit intensities q_i. (Right) Consistent Parallel Decoding & Commit: The refined, interaction-aware selection enables consistent parallel updates. This induces inhibitory signals that unmask only non-conflicting high-confidence positions, yielding a more reliable parallel update while increasing parallelism.

marginal statistics and do not explicitly account for dependencies between simultaneously committed tokens.

Modeling inter-token dependencies in non-autoregressive generation has been explored through iterative refinement [Ghazvininejad et al., 2019] and latent-variable formulations [Gu et al., 2018]. In the context of diffusion language models, recent work has begun to explicitly address the limitations of factorized decoding. Adaptive parallel decoding (APD) [Israel et al., 2025] incorporates an auxiliary autoregressive model to approximate joint effects. More recent works directly model dependencies at decoding time: DAPD [Kim et al., 2026] constructs an attention-induced dependency graph and selects independent token subsets, DAWN [Luo et al., 2026] estimates inter-token dependencies from attention and constructs a dependency graph to guide parallel commit selection, avoiding jointly updating strongly coupled positions, and DEMASK [Ringel et al., 2026] learns a lightweight predictor to estimate pairwise conditional influence and bound joint mismatch. DAWN is the closest prior work: both methods use pairwise structure over masked positions (a weighted graph) to avoid unsafe parallel commits. DAWN thresholds attention into binary edges and resolves conflicts with a staged scheduler (several calibrated thresholds and greedy independent-set selection), whereas MPD keeps continuous interaction strengths and derives all decisions from one energy with one commit threshold (Appendix A), Section 5.4 shows that this decision rule improves accuracy with either interaction signal.

Structured prediction frameworks, including energy-based models and graphical models, provide a principled way to encode dependencies via pairwise or higher-order interactions [LeCun et al., 2006]. In particular, Markov Random Fields and mean-field variational inference [Wainwright and Jordan, 2008, Jordan et al., 1999] have been widely used to approximate joint configurations through iterative updates combining unary and pairwise terms. Similar formulations have been applied in vision and sequence modeling for enforcing consistency constraints, but typically require learned energy functions or additional parameterization. Our method applies this formulation to decoding, with pairwise interactions defined in the model's output space. Combined with a variational mean-field procedure, it enables coordinated parallel token commitment without additional training or auxiliary models. Distributional divergences such as KL and JSD are widely used to compare predictive distributions in neural models, including uncertainty estimation, calibration, and distillation. In our setting, JSD is not used as a training objective, instead, it serves as a bounded, inference-time proxy for overlap between masked-token predictive distributions.

3 Background: Discrete Diffusion Language Models

Discrete Diffusion Language Models (dLLMs) define a generative process over a sequence of L tokens $x = (x_1, \ldots, x_L) \in \mathcal{V}^L$, where $\mathcal{V}$ is a discrete vocabulary, by reversing a discrete-state corruption process. Let x_0 be the clean data and x_T be a fully corrupted sequence, typically where all tokens are replaced by a special `[MASK]` token. The forward process $q(x_t \mid x_{t-1})$ gradually adds noise, while the model Φ_θ learns to approximate the reverse transition $p_\theta(x_{t-1} \mid x_t)$.

Iterative Denoising. At any denoising step $t \in \{T, \ldots, 1\}$, let $\mathcal{U}_t = \{i \in \{1, \ldots, L\} : x_{t,i} = \texttt{[MASK]}\}$ denote the set of indices that remain masked. The model Φ_θ predicts a categorical distribution over the vocabulary $\mathcal{V}$ for each masked position $i \in \mathcal{U}_t$:

$$p_\theta(x_i \mid x_t) = \text{Cat}(x_i, \pi_{t,i}), \quad \forall i \in \mathcal{U}_t, \tag{1}$$

where $\pi_{t,i} \in \Delta^{|\mathcal{V}|-1}$ is the predicted probability vector. In standard iterative decoding, a subset of positions $\mathcal{C}_t \subseteq \mathcal{U}_t$, referred to as the *commit set*, is selected to be unmasked. The tokens at these positions are realized and fixed for the subsequent step $t - 1$.

The Parallelism-Consistency Trade-off. The efficiency of dLLMs stems from the ability to commit multiple tokens in parallel ($|\mathcal{C}_t| > 1$). However, most dLLMs employ a factorized reverse policy:

$$p_\theta(x_{\mathcal{C}_t} \mid x_t) \approx \prod_{i \in \mathcal{C}_t} p_\theta(x_i \mid x_t). \tag{2}$$

This independence assumption holds only when the selected positions in $\mathcal{C}_t$ do not interact under the current context $x_{\setminus \mathcal{U}_t}$. In practice, tokens often exhibit strong interactions arising from syntactic and semantic constraints. As a result, aggressive parallelism can produce *joint inconsistency* [Ringel et al., 2026, Luo et al., 2026], where tokens that are individually probable form incompatible configurations when committed simultaneously. A common failure mode of aggressive parallel decoding is that adjacent or syntactically related masked positions share similar predictive distributions, committing them simultaneously from independent marginals produces redundant or mutually incompatible tokens. JSD between predictive distributions provides a direct, low-cost test for this overlap, computed entirely from the single forward pass already performed by the model. Similar marginals do not, however, identify the full joint conditional, so MPD targets a narrower notion than joint inconsistency in general: *predictive redundancy and counterfactual coupling*, i.e., pairs of positions for which revealing the token at one changes the predictive distribution at the other. Section 5.4 measures how well JSD tracks this coupling.

4 Methodology

4.1 Pairwise Interaction Commit Selection

We treat the selection of the commit set $\mathcal{C}_t$ as a structured inference problem. We define a vector of binary indicators $S = (s_1, \ldots, s_m) \in \{0, 1\}^m$, where $m = |\mathcal{U}_t|$ is the number of currently masked positions. Here, $s_i = 1$ indicates that position i is selected for the commit set $\mathcal{C}_t$, and $s_i = 0$ otherwise.

4.1.1 Energy-Based Formulation

We define a pairwise Markov Random Field (MRF) [Wainwright and Jordan, 2008] over the commit indicators S to capture the trade-off between local confidence and joint compatibility. The probability of a commit configuration S is given by the Gibbs distribution:

$$P(S, \mathcal{U}_t) = \frac{1}{Z} \exp\left(\sum_{s_i \in S} \phi_i(s_i) + \sum_{\substack{s_i, s_j \in S \\ i<j}} \phi_{ij}(s_i, s_j) \right), \tag{3}$$

where Z is the partition function, ϕ_i is the unary potential, and ϕ_{ij} is the pairwise potential.

Algorithm 1: Mean-Field Parallel Decoding

Input: Masked indices $\mathcal{U}_t$, logits $\{\ell_i\}_{i\in\mathcal{U}_t}$, commit threshold τ, number of mean-field iterations R

Output: Commit set $\mathcal{C}_t$

// 1. Local Confidence Estimation

$\pi_{t,i} \leftarrow \text{softmax}(\ell_i)$ for all $i \in \mathcal{U}_t$ $c_i \leftarrow \log \pi_{t,i}(v_i^{(1)}) - \log \pi_{t,i}(v_i^{(2)})$ // 2. Pairwise Compatibility Construction

$\tilde{D}_{ij} \leftarrow (1 - \text{JSD}(\pi_{t,i}, \pi_{t,j})/\ln 2)$ for $i \neq j$ $D \leftarrow \tilde{D}/\max(\tilde{D})$ // 3. Mean-Field Relaxation

$q^{(0)} \leftarrow \sigma(c)$ **for** $r = 0$ **to** $R - 1$ **do**

  $q^{(r+1)} \leftarrow \sigma(c - Dq^{(r)})$

// 4. Thresholding

$\mathcal{C}_t \leftarrow \{i \in \mathcal{U}_t : q_i^{(R)} \geq \tau\}$ **return** $\mathcal{C}_t$

Unary Potentials (Local Confidence). The unary potential $\phi_i(s_i) = c_i s_i$ encourages committing positions with high local certainty. We define the local confidence c_i as the log-margin between the top-2 predicted tokens:

$$c_i = \log \pi_i(v_i^{(1)}) - \log \pi_i(v_i^{(2)}), \tag{4}$$

where $v_i^{(1)}$ and $v_i^{(2)}$ denote the most likely and second most likely tokens under the predictive distribution π_i, respectively.

Pairwise Potentials (Predictive Overlap). The pairwise potential $\phi_{ij}(s_i, s_j) = -D_{ij} s_i s_j$, where $s_i, s_j \in \{0, 1\}$ denote the commit indicators for masked positions i and j, penalizes the simultaneous commitment of strongly coupled positions. We construct the pairwise matrix $D \in [0, 1]^{m\times m}$ using the normalized Jensen-Shannon Divergence (JSD) between the predicted distributions $\pi_{t,i}$ and $\pi_{t,j}$:

$$\tilde{D}_{ij} = 1 - \frac{\text{JSD}(\pi_{t,i}, \pi_{t,j})}{\ln 2}, \tag{5}$$

where the JSD is defined as:

$$\text{JSD}(p, q) = \frac{1}{2}\text{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}\text{KL}(q\|\frac{p+q}{2}). \tag{6}$$

We set $D_{ij} = \tilde{D}_{ij}/\max_{a,b} \tilde{D}_{ab}$ for $i \neq j$ to normalize the scores within the current active set, and $D_{ii} = 0$ to avoid self-inhibition. This "antiferromagnetic" coupling ensures that if two positions have similar output distributions (high D_{ij}), they compete for commitment, discouraging redundant or inconsistent parallel updates.

4.1.2 Variational Mean-Field Inference

Since finding the MAP assignment $S^* = \arg\max P(S)$ is NP-hard for general D [Wainwright and Jordan, 2008], we employ a mean-field approximation. We approximate $P(S)$ with a fully factorized distribution $q(S) = \prod_{i=1}^{m} q_i^{s_i}(1 - q_i)^{1-s_i}$, where $q_i \in [0, 1]$ represents the *commit intensity* (the marginal probability that $s_i = 1$). We maximize the Evidence Lower Bound (ELBO):

$$\mathcal{L}(q) = \mathbb{E}_q \left[\sum_i c_i s_i - \sum_{i<j} D_{ij} s_i s_j \right] + \mathcal{H}(q), \tag{7}$$

where $\mathcal{H}(q) = \sum_i H(q_i)$ is the sum of binary entropies $H(q_i) = -q_i \log q_i - (1 - q_i) \log(1 - q_i)$. Expanding the expectation:

$$\mathcal{L}(q) = \sum_{i=1}^{m} c_i q_i - \frac{1}{2} \sum_{i\neq j} D_{ij} q_i q_j + \sum_{i=1}^{m} \left[-q_i \log q_i - (1 - q_i) \log(1 - q_i) \right]. \tag{8}$$

Setting the partial derivative $\frac{\partial \mathcal{L}}{\partial q_i} = 0$ yields the fixed-point equation:

$$c_i - \sum_{j \neq i} D_{ij} q_j - \log \frac{q_i}{1 - q_i} = 0, \tag{9}$$

which implies:

$$q_i = \sigma \left(c_i - \sum_{j \neq i} D_{ij} q_j \right), \tag{10}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the logistic sigmoid function.

Iterative update and commit rule. We approximate this fixed point with R synchronous mean-field updates, where R is the number of iterations:

$$q^{(0)} = \sigma(c), \qquad q^{(r+1)} = \sigma\Big(c - D q^{(r)}\Big), \quad r = 0, \ldots, R-1, \tag{11}$$

and commit $\mathcal{C}_t = \{i \in \mathcal{U}_t : q_i^{(R)} \geq \tau\}$, where τ is the commit threshold. Because σ is monotone, position i is committed if and only if $c_i - \sum_{j \neq i} D_{ij} q_j^{(R-1)} \geq \log \frac{\tau}{1-\tau}$: its confidence must exceed its accumulated conflict with positions that are themselves likely to be committed. Thus τ is the only accept/reject threshold, and R is a fixed iteration budget. The synchronous update is not guaranteed to converge and can oscillate between odd and even iterates under strong interactions, empirically, even-iterate commit decisions stabilize quickly, and $R = 2$ closely matches $R = 8$ (Section 5.4). Algorithm 1 summarizes the procedure.

Complexity and Runtime Overhead. The proposed framework is entirely training-free and operates within the scope of a single forward pass of the base discrete diffusion model, requiring no auxiliary predictors or fine-tuning stages. The primary computational overhead arises from the $O(m^2|\mathcal{V}|)$ complexity of the pairwise Jensen–Shannon Divergence (JSD) computation, where m is the number of masked positions and $|\mathcal{V}|$ is the vocabulary size. This operation consists of independent, element-wise calculations that are highly amenable to parallelization on modern GPUs, and for the block sizes used in our main experiments the additional latency is small compared to the model's execution time. Each mean-field iteration costs one $m \times m$ matrix-vector product, and we use $R = 2$, so the refinement does not bottleneck the inference pipeline.

5 Experiments

We empirically evaluate the proposed decoding framework under standard discrete diffusion generation settings. Our experiments focus on quantifying the impact of structured commit selection on the quality-latency trade-off, compared to marginal-based and dependency-aware baselines.

5.1 Setups

Datasets. We evaluate the proposed decoding method on four standard benchmarks spanning mathematical reasoning and code generation. For reasoning, we use GSM8K [Cobbe et al., 2021] and MATH [Hendrycks et al., 2021], which test grade-school arithmetic and competition-level mathematical problem solving, respectively. For code generation, we use HumanEval [Chen et al., 2021] and MBPP [Austin et al., 2021b], which evaluate functional correctness on Python programming tasks. All decoding methods are evaluated with the same model, dataset split, prompts, generation budget, and evaluator, so that observed differences reflect the decoding strategy rather than the evaluation protocol.

Models. We evaluate all decoding methods on three recent dLLM backbones: `Dream-v0-Instruct-7B`[1] [Ye et al., 2025], `LLaDA-1.5`[2] [Zhu et al., 2025], and

[1] `https://huggingface.co/Dream-org/Dream-v0-Instruct-7B`
[2] `https://huggingface.co/GSAI-ML/LLaDA-1.5`

Table 1: Performance comparison between Mean-Field Parallel Decoding (Ours) and baselines on GSM8K, MATH, HumanEval, and MBPP with `LLaDA-8B-Instruct`, `LLaDA-1.5`, and `Dream-v0-Instruct-7B`. We report accuracy (%), throughput in tokens per second (TPS), and speedup relative to Entropy decoding. †KLASS as implemented in the DAWN codebase under our unified protocol (Appendix B), not a reproduction of the results reported by Kim et al. [2025a].

	GSM8K			MATH			HumanEval			MBPP		
Method	Acc.↑	TPS↑	Speedup↑	Acc.↑	TPS↑	Speedup↑	Acc.↑	TPS↑	Speedup↑	Acc.↑	TPS↑	Speedup↑
LLaDA-8B-Instruct												
Entropy	79.15	15.87	1.00×	33.36	23.59	1.00×	40.24	38.02	1.00×	29.4	15.72	1.00×
KLASS†	75.05	36.23	2.27×	31.80	42.20	1.79×	39.63	70.30	1.83×	27.2	35.60	2.28×
LocalLeap	77.71	68.96	4.32×	32.96	77.87	3.30×	40.24	152.62	4.04×	30.6	72.64	4.60×
DAWN	77.78	69.76	4.38×	32.34	80.40	3.41×	40.24	152.46	4.05×	28.0	74.46	4.87×
Ours	76.34	**78.75**	**4.93**×	31.10	**96.37**	**4.07**×	40.70	**180.81**	**4.88**×	28.0	**81.11**	**6.10**×
LLaDA-1.5												
Entropy	81.80	14.90	1.00×	33.62	20.91	1.00×	44.51	13.04	1.00×	39.2	5.77	1.00×
KLASS†	79.37	35.22	2.37×	32.50	38.52	1.87×	43.90	18.94	1.48×	29.8	18.25	4.08×
LocalLeap	80.36	65.69	4.41×	33.02	68.87	3.29×	42.07	41.91	3.23×	38.0	43.50	7.57×
DAWN	79.60	66.74	4.47×	32.16	70.90	3.39×	42.68	41.50	3.20×	37.4	45.36	7.90×
Ours	80.74	**75.18**	**4.96**×	32.46	**86.71**	**4.10**×	42.24	**54.91**	**4.32**×	38.6	**50.95**	**8.62**×
Dream-v0-Instruct-7B												
Entropy	75.51	11.15	1.00×	38.30	28.18	1.00×	53.04	32.05	1.00×	54.20	13.40	1.00×
KLASS†	73.54	19.61	1.82×	36.64	29.26	1.70×	53.65	38.63	1.83×	56.60	29.00	1.30×
LocalLeap	73.61	49.80	4.49×	38.38	68.62	2.51×	56.09	84.42	3.05×	55.00	66.86	5.39×
DAWN	73.31	50.15	4.53×	38.42	69.77	2.57×	54.87	89.56	3.26×	55.60	69.50	5.62×
Ours	74.90	**54.75**	**4.91**×	40.44	**74.38**	**3.61**×	-	-	-	54.40	**75.68**	**5.80**×

`LLaDA-8B-Instruct`[3] [Nie et al., 2025]. These checkpoints cover instruction tuning (Dream-v0-Instruct-7B and LLaDA-8B-Instruct) and preference optimization (LLaDA-1.5), allowing us to assess whether the proposed commit-selection rule generalizes across dLLM families. For each backbone, we use the publicly released checkpoint and keep the model weights fixed, only the decoding strategy is varied across methods.

Baselines. We compare against representative decoding strategies that cover uncertainty-based decoding, confidence-based acceleration, locality-aware scheduling, and dependency-aware parallel commitment. The Entropy baseline commits tokens according to position-wise predictive uncertainty and serves as the reference for computing relative speedup. KLASS [Kim et al., 2025a] selects tokens using a KL-based confidence criterion, providing a stronger marginal-selection baseline. LocalLeap [Kong et al., 2025] uses a training-free, anchor-guided schedule: it identifies high-confidence tokens as anchors and relaxes the commit criterion for nearby positions within a bounded local neighborhood, thereby exploiting local determinism without explicitly constructing a pairwise dependency matrix. DAWN [Luo et al., 2026] introduces dependency-aware scheduling heuristics designed to increase safe parallelism during diffusion decoding. Together, these baselines allow us to compare the proposed structured mean-field commit rule against marginal confidence selection, local confidence propagation, and existing dependency-aware parallel decoding methods. Because DEMASK [Ringel et al., 2026] requires a separately trained dependency predictor, we exclude it from the training-free comparison in Table 1 and compare against it separately in Section 5.4.

Metrics. We report three metrics: task accuracy, tokens per second (TPS), and speedup relative to the Entropy baseline. Accuracy measures task-level generation quality, using the standard evaluation protocol for each benchmark. TPS measures decoding throughput and is computed as the number of generated tokens divided by wall-clock decoding time. Speedup is computed as the ratio of the Entropy baseline's total wall-clock decoding time to that of the evaluated method, under the same model and benchmark. Reporting these metrics jointly characterizes the quality-latency trade-off:

[3] `https://huggingface.co/GSAI-ML/LLaDA-8B-Instruct`

Table 2: Ablation of the proposed decoding components and inference controls using `Dream-v0-Instruct-7B` on GSM8K (τ=0.9, R=2, Block size = 32). We report task accuracy, throughput, and the number of denoising function evaluations (NFE).

Variant	Acc. ↑	TPS ↑	NFE ↓
Baseline (Entropy)	75.51	11.15	256.0
No pairwise interaction	72.40	56.20	48.7
Uniform interaction matrix	71.95	51.92	52.6
One-shot update, R=1	54.51	20.11	140.0
Ours	73.24	49.13	54.3

Table 3: Crossover of decision rule and interaction signal (`Dream-v0-Instruct-7B`, 500 GSM8K examples, 5-shot, length 256, block 32).

Rule	Signal	Acc.	Tok/s	NFE
DAWN	Attention	73.6	92.95	51.42
DAWN	JSD	73.6	92.83	51.42
MPD	Attention	74.8	94.29	50.83
MPD	JSD	**76.0**	93.52	51.85

a desirable decoder should improve throughput without causing a disproportionate degradation in task accuracy. Where relevant, we also report the number of function evaluations (NFE), i.e., model forward passes.

Hardware and Implementation Details. All experiments in Table 1 are implemented within the released DAWN[4] evaluation codebase. Every method shares the same checkpoint, prompts, generation budget, evaluator, hardware, and timing boundary, while each baseline uses its own per-method and per-task configuration from that implementation (Appendix B), no baseline is run with MPD's hyperparameters. Throughput is measured end-to-end during decoding, including the overhead of commit selection and mean-field inference. For LLaDA models, we tune the decoding parameters of MPD on 100 samples from the GSM8K training set and use block size 20, commit threshold $\tau = 0.85$, and $R = 2$ mean-field iterations. The experiments in Table 1 are conducted on a cluster with $8\times$ NVIDIA H100 GPUs, the hardware used for each additional experiment is stated in Appendix C.

5.2 Quantitative Results

Table 1 summarizes the quality and latency trade-off across the reasoning and code-generation benchmarks. In every reported setting, our method obtains the highest TPS among all compared methods while keeping task accuracy close to that of the baselines. The average speedup relative to entropy-based decoding is $5.12\times$, with a range from $3.61\times$ on `Dream-v0-Instruct-7B` for MATH to $8.62\times$ on `LLaDA-1.5` for MBPP. These results indicate that the proposed Mean-Field Parallel Decoding rule increases parallelism substantially beyond marginal confidence selection.

The comparison with Entropy and KLASS highlights the benefit of moving beyond position-wise commit criteria. Entropy is generally the most conservative baseline: it often preserves strong accuracy, but does so with substantially lower throughput. KLASS improves efficiency with a stronger confidence-based rule, yet its gains remain below those of methods that use dependency or scheduling information. The proposed method shifts the trade-off toward higher throughput while keeping accuracy within a narrow margin of the entropy baseline, and in some cases matches or exceeds it.

Compared with the dependency-aware baselines LocalLeap and DAWN, the proposed method provides a consistent throughput advantage, although it is not the most accurate method in every setting. Across the eight LLaDA-family settings, DAWN is more accurate than MPD in three (GSM8K and MATH on LLaDA-8B-Instruct, and HumanEval on LLaDA-1.5) by 1.44, 1.24, and 0.44 points, while MPD is 12.9%, 19.9%, and 32.3% faster in these settings, in the remaining five, MPD matches or exceeds DAWN's accuracy (by up to 1.2 points) while also being faster (Appendix A). MPD thus shifts the quality-latency Pareto frontier rather than dominating every baseline in accuracy, without an auxiliary verifier or additional forward passes.

[4] `https://github.com/lizhuo-luo/DAWN`

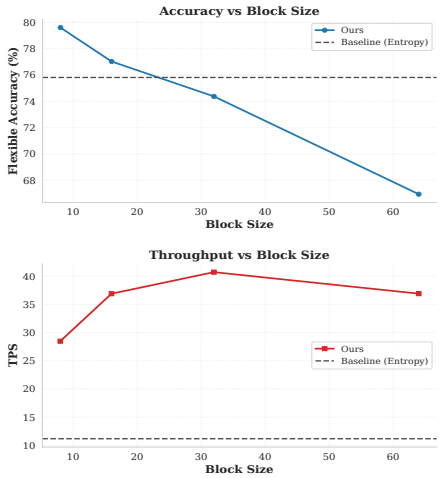


Figure 2: Block-size sensitivity on GSM8K using `Dream-v0-Instruct-7B`. We evaluate our method with $R = 2$ and $\tau = 0.95$.

Table 4: Sensitivity to decoding controls on GSM8K using `Dream-v0-Instruct-7B`. We vary the commit threshold τ and mean-field iterations R around the default setting $\tau = 0.95$, $R = 2$, and block size 20.

Setting	Value	Acc.	TPS	Speed	NFE
Baseline	–	75.51	11.15	1.00×	256.0
τ	0.75	66.41	57.78	5.23×	47.8
	0.80	71.42	52.46	4.75×	52.5
	0.85	72.86	47.57	4.35×	57.5
	0.90	75.36	42.84	3.90×	64.1
	0.95	77.03	36.89	3.38×	74.1
R	2	77.03	36.88	3.38×	74.1
	4	75.36	35.64	3.28×	76.2
	6	75.06	34.96	3.23×	77.3
	8	74.53	34.70	3.20×	78.2

5.3 Ablation Studies

We conduct a set of ablation studies to evaluate the contribution of each component in the proposed decoding method. As shown in Table 2, the entropy baseline achieves the highest accuracy but requires substantially more denoising function evaluations, reflecting its conservative token-commitment behavior. Removing the pairwise interaction term greatly improves throughput and reduces NFE, but it also leads to a larger accuracy drop, indicating that local confidence alone is not sufficient to preserve generation quality under aggressive parallel decoding.

The uniform interaction matrix provides a further control for the pairwise interaction term. Its weaker performance compared with the full method suggests that the structure of the JSD-derived interaction matrix is critical, rather than merely the presence of a generic pairwise penalty. The one-shot update variant with R=1 performs substantially worse, showing that a single mean-field update is insufficient to produce a reliable commit set. Overall, the full method provides the best balance among accuracy, throughput, and NFE, supporting the use of structured pairwise interactions together with iterative mean-field refinement.

Furthermore, we analyze the sensitivity of the proposed decoder to its main inference controls on GSM8K, as shown in Table 4. The commit threshold τ directly controls the aggressiveness of parallel token commitment. Lower thresholds produce higher throughput and fewer denoising steps, but they also reduce accuracy, indicating that overly aggressive commitment can amplify conflicts between simultaneously committed tokens. Increasing τ makes the decoder more conservative, improving accuracy while gradually reducing TPS and speedup. We also vary the number of mean-field iterations R. The results show that using R=2 is sufficient in this setting, additional iterations provide no accuracy benefit and slightly reduce throughput due to the added inference cost. Together with the threshold analysis, this shows a smooth and interpretable quality-latency trade-off.

Finally, Figure 2 analyzes the effect of block size on `Dream-v0-Instruct-7B` using GSM8K, with $R = 2$ and $\tau = 0.95$ fixed. Increasing the block size improves throughput by expanding the set of positions considered for parallel commitment at each step, while accuracy decreases for large blocks. Section 5.4 shows that this degradation is largely inherited from the base model's predictive marginals. Since a fixed τ also yields different effective commit rates as the active set grows, we choose block size and τ jointly per model to obtain Pareto-optimal operating points.

5.4 Further Analyses

Decision rule versus interaction signal. We supply each interaction signal (attention, as in DAWN, or our JSD-based D) unchanged to each decision rule, for MPD with attention, the scores are symmetrized as part of this control. Swapping attention for JSD inside DAWN's scheduler leaves

accuracy and NFE unchanged, whereas the MPD rule improves accuracy with either signal (by 1.2 and 2.4 points) at nearly identical cost (Table 3), so the gain does not reduce to replacing attention with JSD.

Is the gain explained by block size? We compare MPD with a top-k confidence control that, at every step, uses the same prompt, model, block, argmax predictions, number of committed tokens, and NFE, so that only the selected positions differ. Over Dream and LLaDA, GSM8K and MBPP, and block sizes 16–128 (500 prompts per cell), MPD has the higher accuracy point estimate in all 16 cells (by 0.4–5.8 points, 10 of 16 paired bootstrap 95% intervals exclude zero) and lower repetition in all 16 (Table 13), for every model/task pair, the advantage is smaller at block size 128 than at 16. The accuracy loss at large blocks is largely inherited from the base model: with greedy one-token-per-step decoding, LLaDA-8B-Instruct falls from 79.6% (block 64) to 59.6% (block 256) on GSM8K, and MPD follows this trend (78.2% to 56.2%) with 2.4–2.8$\times$ fewer NFE, as larger blocks leave each marginal less conditionally informed [Zhang et al., 2026, Chen et al., 2025]. On Dream at block 256, by contrast, MPD improves over greedy decoding (52.4% vs. 36.8%, Appendix D).

Does JSD track counterfactual coupling? In 24 GSM8K problems, we mask 10 answer positions, reveal the top-1 prediction at one, and measure the JSD shift at another. Over 1,080 position pairs, D_{ij} has a Spearman correlation of 0.552 (95% CI [0.465, 0.652]) with this counterfactual influence, and the correlation is 0.306 for pairs whose top-1 tokens differ (Appendix E), JSD is thus an informative, though partial, proxy for coupling. Appendix F shows qualitative examples.

Stability and further evaluations. On identical decoding states, the commit set at $R = 2$ matches that at $R = 8$ with 98.35–99.05% Jaccard similarity across Dream and LLaDA on GSM8K and MBPP, on Dream/GSM8K, $R = 1$ reaches only 39.92% (Appendix G). In Appendix C, MPD outperforms DAWN on constrained JSON generation [Mündler et al., 2025], matches DEMASK on GSM8K (within 0.03 points) and exceeds it by 2.43 points on HumanEval at 1.57–1.73$\times$ its throughput, increases parallelism on the editable-state LLaDA2.1-mini [Bie et al., 2026], and cuts NFE by 13.7–16.6$\times$ on DiffuCoder [Gong et al., 2026], with pass@1 changes of -2.4 to $+6.7$ points.

6 Limitations

MPD has several limitations. First, the pairwise interaction matrix costs $O(m^2|\mathcal{V}|)$ per step. Block-based decoding bounds m, but for very large blocks the cost is no longer negligible: at block size 256 on LLaDA-8B-Instruct, MPD's throughput matches that of greedy decoding despite 2.4$\times$ fewer forward passes. Second, JSD is a pairwise proxy for predictive redundancy and counterfactual coupling, it does not model higher-order structure or correct an incorrect marginal, and its advantage is smaller at the largest block sizes. Third, our evidence that two mean-field iterations suffice is empirical, not a convergence guarantee. Finally, the best block size and threshold vary across models and tasks.

7 Conclusions

In this work, we introduced Mean-Field Parallel Decoding, a training-free framework that mitigates the joint inconsistency bottleneck in discrete diffusion language models. By formulating parallel token selection as a variational mean-field optimization, we replace independent marginal updates with a structured process that balances local confidence against global compatibility. This approach enables the decoder to safely unmask a larger number of tokens in each step by using the model's own predictive distributions to suppress conflicting updates. Our experiments demonstrate that this coordinated selection significantly improves the quality-latency trade-off across reasoning and coding tasks without requiring auxiliary models or retraining. Ultimately, Mean-Field Parallel Decoding provides a principled, single-pass mechanism for improving the throughput of diffusion-based generation while preserving the linguistic coherence essential for complex text synthesis.

Acknowledgments and Disclosures

This work was supported by a grant from the Tel Aviv University Center for AI and Data Science (TAD). This research was also supported by the Ministry of Innovation, Science & Technology, Israel (1001576154) and the Michael J. Fox Foundation (MJFF-022407). The contribution of Tamim Zoabi is part of a PhD thesis research conducted at Tel Aviv University.

Competing interests. The authors declare no competing interests.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *NeurIPS*, 2021a.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

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

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, Chengxi Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Yuan Lu, Yuxin Ma, Xingyu Mou, Zhenxuan Pan, Kaida Qiu, Yuji Ren, Jianfeng Tan, Yiding Tian, Zian Wang, Lanning Wei, Tao Wu, Yipeng Xing, Wentao Ye, Liangyu Zha, Tianze Zhang, Xiaolu Zhang, Junbo Zhao, Da Zheng, Hao Zhong, Wanli Zhong, Jun Zhou, Junlin Zhou, Liwang Zhu, Muzhi Zhu, and Yihong Zhuang. Llada2.1: Speeding up text diffusion via token editing, 2026.

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

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. *arXiv preprint arXiv:2511.04647*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*, 2019.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. DiffuCoder: Understanding and improving masked diffusion models for code generation. In *ICLR*, 2026.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *ICLR*, 2018.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *NeurIPS*, 2021.

Daniel Mingyi Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion LLMs via adaptive parallel decoding. In *NeurIPS*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *ICLR*, 2025.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

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

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. In *NeurIPS*, 2025a.

Sungkyun Kim, Jaemin Kim, Dogyung Yoon, Jiho Shin, Junyeol Lee, and Jiwon Seo. Speculative verification: Exploiting information gain to refine speculative decoding. *arXiv preprint arXiv:2509.24328*, 2025b.

Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Guorui Zhou, et al. Accelerating diffusion llm inference via local determinism propagation. *arXiv preprint arXiv:2510.07081*, 2025.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In *NeurIPS*, 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Lizhuo Luo, Zhuoran Shi, Jiajun Luo, Zhi Wang, Shen Ren, Wenya Wang, and Tianwei Zhang. Dawn: Dependency-aware fast inference for diffusion llms, 2026.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion LLMs with context-free grammars. *arXiv preprint arXiv:2508.10111*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *NeurIPS*, 2025.

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees. *arXiv preprint arXiv:2604.12989*, 2026.

Liran Ringel, Ameen Ali, and Yaniv Romano. Dependency-guided parallel decoding in discrete diffusion language models. *arXiv preprint arXiv:2604.02560*, 2026.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

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

Shaorong Zhang, Longxuan Yu, Rob Brekelmans, Luhan Tang, Salman Asif, and Greg Ver Steeg. Generation order and parallel decoding in masked diffusion models: An information-theoretic perspective. *arXiv preprint arXiv:2602.00286*, 2026.

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, 2025.

A Stage-by-Stage Comparison with DAWN

Table 5 aligns DAWN [Luo et al., 2026] and MPD stage by stage. Both methods encode pairwise structure over masked positions. The distinction lies in how this structure is used. DAWN converts attention into a sparse conflict graph and passes it through a staged discrete scheduler, MPD keeps continuous interaction strengths and places confidence and pairwise conflict inside a single scalar energy (Eq. 3), whose mean-field update (Eq. 11) determines every commit decision through one threshold.

Table 5: Stage-by-stage comparison of DAWN and MPD.

Stage	DAWN	MPD
Model signal	Logits and attention a aggregated over selected heads and layers.	Logits only, predictive distributions of all masked positions from the same pass.
Interaction representation	Attention sinks removed with threshold t_{sink}, directed attention binarized as $G^{\rightarrow}_{ij} = \mathbb{1}[a_{ij} \geq t_{\text{edge}}] \cdot \mathbb{1}[j \notin S_{\text{sink}}]$.	Symmetric, continuous $D_{ij} = D_{ji}$, $D_{ii} = 0$, with $D_{ij} = \tilde{D}_{ij} / \max_{a,b} \tilde{D}_{ab}$ and $\tilde{D}_{ij} = 1 - \text{JSD}(\pi_{t,i}, \pi_{t,j}) / \ln 2$.
Pairwise conflict	Direction discarded after thresholding: $C_{ij} = G^{\rightarrow}_{ij} \vee G^{\rightarrow}_{ji}$.	Every interaction contributes by magnitude via $D_{ij} q_i q_j$, no binary cutoff.
Initial eligibility	High-confidence and anchor-induced sets formed with t_{high} and t_{induced}, remaining candidates retained with t_{low}.	All masked positions initialized from confidence, $q^{(0)} = \sigma(c)$, no position is removed by an interaction cutoff.
Conflict resolution	One-shot, greedy, confidence-weighted independent-set selection under C on the eligible subgraph.	R synchronous updates $q^{(r+1)} = \sigma(c - Dq^{(r)})$, every position is refined against all continuous conflicts.
Commit	$U_{\text{anchor}} \cup U_{\text{conflict}}$: union of separately constructed hard sets.	$\{i : q_i^{(R)} \geq \tau\}$, a thresholded solution of one energy.
Adding position i	Eligibility depends on which hard gates and graph constraints are crossed.	$\Delta_i E = -c_i + \sum_{j \in S} D_{ij}$: the confidence benefit must exceed the accumulated conflict with the current set S.
Common objective	None linking graph construction, gates, anchor induction, and scheduling.	All decisions derive from the energy in Eq. 3.
Calibrated thresholds	Five: t_{sink}, t_{edge}, t_{high}, t_{induced}, t_{low}.	One: τ (R is a fixed iteration budget, not an accept/reject gate).

Calibration. DAWN reports grid-searching its sink, edge, and induced thresholds per evaluated model on HumanEval, and its low-confidence threshold directly moves the accuracy/throughput operating point. Its trade-off is thus distributed across coupled, discontinuous cutoffs, where crossing one threshold can change the graph, the candidate set, or the independent set. MPD is not parameter-free, but its operating point is determined by one energy and one final commit criterion rather than by separate thresholds for interaction existence, anchor status, candidate eligibility, and graph scheduling.

Per-setting comparison. Table 6 isolates the DAWN and MPD rows of Table 1 for the LLaDA family.

Table 6: DAWN versus MPD on the LLaDA family (from Table 1). TPS change is the relative throughput of MPD over DAWN.

Model	Task	DAWN Acc.	MPD Acc.	DAWN TPS	MPD TPS	TPS change
LLaDA-8B-Instruct	GSM8K	77.78	76.34	69.76	78.75	+12.9%
LLaDA-8B-Instruct	MATH	32.34	31.10	80.40	96.37	+19.9%
LLaDA-8B-Instruct	HumanEval	40.24	40.70	152.46	180.81	+18.6%
LLaDA-8B-Instruct	MBPP	28.0	28.0	74.46	81.11	+8.9%
LLaDA-1.5	GSM8K	79.60	80.74	66.74	75.18	+12.6%
LLaDA-1.5	MATH	32.16	32.46	70.90	86.71	+22.3%
LLaDA-1.5	HumanEval	42.68	42.24	41.50	54.91	+32.3%
LLaDA-1.5	MBPP	37.4	38.6	45.36	50.95	+12.3%

Crossover protocol. In the crossover of Table 3, each interaction signal is supplied unchanged to each decision rule. For the MPD-with-attention cell, the attention scores are symmetrized and normalized before being used as the interaction matrix,

$$D_{ij}^{\text{attn}} = \frac{(a_{ij} + a_{ji})/2}{\max_{u \neq v} (a_{uv} + a_{vu})/2}, \qquad D_{ii}^{\text{attn}} = 0,$$

this step belongs to the control and is not part of DAWN. MPD uses $R = 2$ and $\tau = 0.85$.

B Decoding Configurations

All methods in Table 1 are run with the released DAWN evaluation implementation and its per-method and per-task configurations. Every method shares the same checkpoint, prompts, few-shot counts, generation budget, evaluator, hardware, and timing boundary, no baseline is run with MPD's hyperparameters. Table 7 lists the settings for `Dream-v0-Instruct-7B`, for which the few-shot counts are 5 (GSM8K), 4 (Minerva MATH), 0 (HumanEval), and 3 (MBPP).

Table 7: Decoding configurations for `Dream-v0-Instruct-7B` in Table 1. All runs use a generation length of 256 tokens. †KLASS as implemented in the DAWN codebase.

Method	Generation schedule	Selection / sampling configuration
Entropy (sequential)	One position per NFE	Greedy entropy / top-1 selection
KLASS†	Single block of 256	Temperature 0.20, top-p 0.95, confidence 0.90 (0.80 on HumanEval), KL 0.001 (0.005 on MATH)
LocalLeap	Block 32 (8 blocks)	Confidence 0.90, relaxed confidence 0.80, radius 4
DAWN	Block 32 (8 blocks)	Confidence 0.80, induced 0.75, sink 0.03, edge 0.10

C Additional Tasks, Models, and Baselines

Constrained JSON generation. We evaluate JSON-Mode-Eval Extended [Mündler et al., 2025] (272 instances) with LLaDA Instruct. Every method receives the same schema-conditioned prompt, a 256-token budget, greedy decoding, and batch size one. Confidence-1 commits a single highest-confidence token per forward pass. MPD uses block size 64, commit threshold 0.9, $R = 2$, and computes the JSD on the model's predictive distributions at temperature 1. Table 8 reports exact match, schema validity, and JSON parse rate. MPD improves over DAWN on all six metrics, including 12.2% higher throughput and 24.1% fewer NFE, and its exact match is within 1.1 points of Confidence-1 at 19.47× its throughput.

Table 8: JSON-Mode-Eval Extended (272 instances) with LLaDA.

Method	Exact match	Schema valid	JSON parse	Tok/s	Speedup	Mean NFE
Confidence-1	55.1%	94.1%	100.0%	19.36	1.00×	256.00
DAWN	52.6%	92.6%	98.9%	335.88	17.35×	14.39
MPD	54.0%	93.4%	99.6%	377.00	19.47×	10.92

Comparison with DEMASK. We evaluate the released DEMASK [Ringel et al., 2026] dependency-predictor checkpoint (base model: Dream-v0-Instruct-7B) and MPD in the same harness on NVIDIA B200 GPUs, with identical prompts, few-shot settings, 256-token budget, argmax decoding, task scorers, and timing boundaries, HumanEval uses the same executable post-processor for both methods. MPD uses block size 20, commit threshold 0.85, and $R = 2$. Unlike DEMASK, MPD uses no learned scheduler, auxiliary labels, or additional checkpoint. Table 9 shows that MPD improves HumanEval pass@1 by 2.43 points and is within 0.03 points on GSM8K, with 1.57–1.73× higher throughput and 40.9–68.1% fewer NFE. We do not claim that MPD dominates DEMASK in accuracy.

Editable-state decoding with LLaDA2.1. LLaDA2.1 [Bie et al., 2026] is an editable-state diffusion model: in standard masked diffusion, a position is written once and then frozen, whereas a single LLaDA2.1 step may both commit new tokens at masked positions (mask-to-token, M2T) and

Table 9: MPD versus DEMASK.

Dataset	Method	Accuracy / pass@1	Tok/s	Mean NFE
GSM8K	DEMASK	74.83%	69.00	80.10
	MPD	74.80%	108.36	47.37
HumanEval	DEMASK	51.22%	96.68	140.59
	MPD	53.65%	166.89	44.83

overwrite already-committed tokens whose predicted argmax has changed (token-to-token, T2T). MPD places both action types in one candidate set,

$$\mathcal{F}_t = \{i : x_i \text{ is } \texttt{[MASK]}\} \cup \{i : x_i \text{ is committed and its new argmax differs from } x_i\},$$

and scores every candidate under the same mean-field energy with a single commit threshold τ. MPD is therefore not a post-filter over LLaDA2.1's native M2T and T2T confidence gates: a T2T correction and a new M2T commitment compete for the same parallel budget and are accepted only if they are compatible under the energy. We evaluate LLaDA2.1-mini on 500 TriviaQA [Joshi et al., 2017] examples (zero-shot, generation length 32, block size 32, greedy decoding, NVIDIA B200). Table 10 shows that with $\tau = 0.70$, MPD improves exact match from 37.0% to 37.6%, raises throughput by 18.7%, and reduces mean NFE by 17.3%. Raising τ to 0.90 increases exact match to 38.4% but lowers throughput below that of native decoding. The relative gains are modest because native LLaDA2.1 decoding is already highly parallel (6.60 tokens per forward pass), MPD raises this to 7.98.

Table 10: LLaDA2.1-mini on TriviaQA (500 examples).

Decoder	τ	R	Exact match	Tok/s	Mean NFE	Tokens/NFE
LLaDA2.1 (native)	–	–	37.0%	36.48	4.848	6.60
MPD	0.70	2	37.6%	43.31	4.008	7.98
MPD	0.90	2	38.4%	31.95	5.302	6.04

DiffuCoder code models. We evaluate DiffuCoder [Gong et al., 2026] in two variants: the instruction-tuned DiffuCoder-7B-Instruct and DiffuCoder-7B-cpGRPO, which is post-trained with outcome-based reinforcement learning.

C++ code generation. On LiveCodeBench-CPP (454 C++ problems derived from LiveCodeBench [Jain et al., 2025], execution-based pass@1), we compare MPD with a confidence baseline that commits only its single highest-confidence token per forward pass. Checkpoint, prompt, and generation budget are held fixed, so only the commitment rule differs, MPD uses $R = 2$ and $\tau = 0.98$. Table 11 shows that MPD improves pass@1 by 2.20 points and compile rate by 18.50 points on DiffuCoder-7B-Instruct, and preserves pass@1 while improving compile rate by 12.33 points on DiffuCoder-7B-cpGRPO, with 5.53–6.44× fewer NFE. Absolute pass@1 is low because these competitive-programming problems are difficult for 7B diffusion models.

Table 11: DiffuCoder on LiveCodeBench-CPP (454 problems). Speedups are relative to the confidence baseline of the same checkpoint.

Checkpoint	Pass@1	Compile rate	Mean NFE	Tok/s	NFE speedup	TPS speedup
DiffuCoder-7B-Instruct	5.29%	27.09%	749.11	49.05	1.00×	1.00×
DiffuCoder-7B-Instruct + MPD	7.49%	45.59%	135.54	163.76	5.53×	3.34×
DiffuCoder-7B-cpGRPO	4.63%	72.25%	208.63	56.35	1.00×	1.00×
DiffuCoder-7B-cpGRPO + MPD	4.63%	84.58%	32.42	202.12	6.44×	3.59×

Python code generation. We evaluate the full HumanEval set (164 problems) and the full MBPP test set (500 problems) against native DiffuCoder decoding, which uses 512 steps, top-$p = 0.95$, and temperatures 0.2 (Instruct) and 0.4 (cpGRPO). Throughput is measured on NVIDIA B200 GPUs. MPD uses block size 20 and $\tau = 0.95$ for DiffuCoder-7B-Instruct, and $\tau = 0.98$ for DiffuCoder-7B-cpGRPO. Table 12 shows that MPD reduces NFE by 13.7–16.6× and increases throughput

by 3.6-22.1×, while keeping pass@1 within 2.4 points of native decoding in every setting, on DiffuCoder-7B-Instruct with HumanEval, it also improves pass@1 by 6.7 points. The quality benefit is largest for the instruction-tuned checkpoint, whereas for cpGRPO the benefit is primarily efficiency. We hypothesize that outcome-based post-training sharpens the predictive distributions and leaves less room for interaction-aware selection to improve accuracy, we have not tested this hypothesis directly.

Table 12: DiffuCoder on HumanEval (164 problems) and MBPP (500 problems). Speedups are relative to native decoding of the same checkpoint.

Checkpoint	Benchmark	Method	Pass@1	Δ	NFE	NFE speedup	Tok/s	TPS speedup
Instruct	HumanEval	Native	60.40%	–	512	1.0×	13.19	1.0×
Instruct	HumanEval	MPD	67.10%	+6.70	30.90	16.6×	200.52	15.2×
Instruct	MBPP	Native	61.60%	–	512	1.0×	28.79	1.0×
Instruct	MBPP	MPD	61.00%	−0.60	35.20	14.5×	104.00	3.6×
cpGRPO	HumanEval	Native	65.20%	–	512	1.0×	6.35	1.0×
cpGRPO	HumanEval	MPD	63.40%	−1.80	35.43	14.5×	140.21	22.1×
cpGRPO	MBPP	Native	64.20%	–	512	1.0×	3.95	1.0×
cpGRPO	MBPP	MPD	61.80%	−2.40	37.47	13.7×	75.43	19.1×

D Block-Size Analysis

Matched-confidence control. At every denoising step, the control commits the k positions with the highest confidence, where k equals the number of positions committed by MPD at that step. The prompt, model, block, argmax predictions, per-step commit count, and NFE are therefore identical, and only the selected positions differ. Each cell uses 500 prompts, confidence intervals are paired bootstrap 95% intervals. MPD uses $\tau = 0.95$ and $R = 2$. Table 13 reports the full results.

Table 13: MPD versus the matched top-k confidence control (accuracy, %). MPD has the higher point estimate in all 16 cells and lower repetition per 100 tokens in all 16 cells, 10 of the 16 intervals exclude zero.

Model / task	B	MPD	Matched confidence	Difference	95% CI
Dream / GSM8K	16	80.6	75.2	+5.4	[2.8, 8.0]
	32	77.2	75.2	+2.0	[−0.2, 4.2]
	64	66.4	63.6	+2.8	[1.0, 4.6]
	128	35.2	34.0	+1.2	[0.0, 2.6]
Dream / MBPP	16	50.0	47.4	+2.6	[0.8, 4.4]
	32	47.4	45.4	+2.0	[0.8, 3.4]
	64	46.0	43.2	+2.8	[1.2, 4.4]
	128	26.2	25.0	+1.2	[−0.2, 2.6]
LLaDA / GSM8K	16	79.0	73.8	+5.2	[2.8, 7.6]
	32	77.0	71.2	+5.8	[3.2, 8.6]
	64	76.0	71.8	+4.2	[1.8, 6.8]
	128	73.0	68.2	+4.8	[2.6, 7.0]
LLaDA / MBPP	16	22.0	21.4	+0.6	[−0.6, 2.0]
	32	18.6	16.2	+2.4	[1.0, 4.0]
	64	16.2	14.8	+1.4	[0.0, 2.8]
	128	15.6	15.2	+0.4	[−1.2, 2.0]

Comparison with greedy decoding at large block sizes. Table 14 compares MPD with greedy decoding that commits one token per forward pass. All runs use 500 GSM8K samples, 5-shot prompting, and a generation length of 256. MPD uses $\tau = 0.98$ for LLaDA-8B-Instruct and $\tau = 0.99$ for `Dream-v0-Instruct-7B`, with $R = 2$ for both. Increasing τ makes MPD more conservative, recovering accuracy and closing the gap to greedy decoding at the cost of more NFE and lower TPS.

Table 14: Greedy decoding versus MPD across block sizes on GSM8K.

Model	Block size	Method	Accuracy (%)	Avg. NFE	TPS
LLaDA-8B-Instruct	64	Greedy	79.6	256.00	28.58
	64	MPD	78.2	92.99	76.45
	128	Greedy	74.4	256.00	28.15
	128	MPD	73.8	93.90	59.45
	256	Greedy	59.6	256.00	27.81
	256	MPD	56.2	105.32	27.71
Dream-v0-Instruct-7B	256	Greedy	36.8	256.00	15.35
	256	MPD	52.4	118.23	17.18

E Counterfactual Coupling Diagnostic

We use 24 GSM8K problems with Dream-v0-Instruct-7B. For each problem, we mask 10 answer positions and compute D from a single forward pass. For each pair of positions (i, j), we reveal the top-1 prediction at one position, rerun the model, and measure the JSD between the predictive distributions at the other position before and after the reveal, this counterfactual influence is symmetrized over both directions. Table 15 summarizes the results. The positive correlation that remains after removing pairs that share the same top-1 token shows that D is not merely an indicator of matching argmax predictions.

Table 15: Relation between D_{ij} and counterfactual influence.

Quantity	Value
Examples / position pairs	24 / 1,080
Spearman correlation between D_{ij} and counterfactual influence	0.552
Example-level bootstrap 95% CI	[0.465, 0.652]
Mean influence: highest / lowest D decile	0.341 / 0.145
Highest-to-lowest decile ratio	2.36×
Correlation when top-1 tokens differ (606 pairs)	0.306

F Qualitative Examples

We compare MPD with the matched top-k confidence scheduler of Appendix D on GSM8K problems. Both methods use the same argmax token predictions, block size, per-step commit counts, and total number of model evaluations, so only the choice of which positions to commit differs. Across four complete examples, MPD answers 4 of 4 correctly versus 1 of 4 for matched confidence, and the number of adjacent-token repetitions falls from 13 to 6. These examples are diagnostic rather than an accuracy estimate, Appendix E and Table 13 provide the aggregate evidence. Outputs are reproduced verbatim, including their errors.

Example 1. *A company divided its employees into 3 groups of 200. Each group was assigned 7 tour guides. How many people are going on the tour?* Reference answer: 621. Both methods use 24 model evaluations.

- Matched confidence (incorrect, 4 adjacent repetitions): "There are 3 * 200 = 600 employees. the tour. There are 600 + 7 = 607 people on the tour. Thus, there are a total of 607 people on the tour. The answer is 607."
- MPD (correct, 3 adjacent repetitions): "There are 3 * 200 = 600 employees. There are 3 * 7 = 21 tour guides. Thus, the total number of people on tour is 600 + 21 = 621. The answer is: 621."

Example 2. *There are 30 students, each starting with 10 pencils. After two months, one fifth of all pencils were used. At the end of the year, only one third of the remaining pencils were left. How many pencils were left?* Reference answer: 80. Both methods use 34 model evaluations.

- Matched confidence (incorrect, 3 adjacent repetitions): "There were 30 students x 10 pencils = 300 pencils in the class. After two months, 1/5 x 300 = 60 pencils had been used. So 300 - 60 = 240 pencils remained."

- MPD (correct, 2 adjacent repetitions): "There were 30 students x 10 pencils = 300 pencils in the class. After two months, 300 x 1/5 = 60 pencils were used. At the end, 240 x 1/3 = 80 pencils were left."

Example 3. *James's insurance costs 60% more than the normal cost of USD 120 per month. How much does he pay per year?* Reference answer: 2304. Both methods use 27 model evaluations.

- Matched confidence (incorrect, 1 adjacent repetition): "The cost is 120 * 60 / 100 = 72 more. So the cost is 120 + 72 = 192 a month. So he pays 192 * 12 = 2302 a year."
- MPD (correct, 1 adjacent repetition): "He pays 120 * 160 / 100 = 192 a month. So he pays 12 * 192 = 2304 a year. The answer is 2304."

Example 4. *Ellen buys two carrots, a USD 6 salad, and a yogurt costing half the salad price. The total is USD 11. What is the cost of one carrot?* Reference answer: 1. Both methods use 33 model evaluations.

- Matched confidence (correct, 5 adjacent repetitions): "The yogurt costs 6 / 2 = 3. So 11 - 6 - 3 - 3 = 2 is the price of two carrots. Ellen Ellen Ellen pays 2/2 = 2 = 1 for 1 carrot carrot every day. answer answer"
- MPD (correct, 0 adjacent repetitions): "The yogurt costs $6 / 2 = $3. So the two carrots cost $11 - $6 - $3 = $2. So Ellen pays $2 / 2 = $1 per carrot. The answer is 1."

G Stability of the Mean-Field Iterations

We evaluate $R \in \{1, 2, 4, 8\}$ on 100 GSM8K examples with `Dream-v0-Instruct-7B` (generation length 256, block size 32), holding all other parameters fixed ($\tau = 0.95$). To measure convergence without the confound of diverging generated prefixes, we additionally compare the intermediate iterates with those of the $R = 8$ trajectory on identical decoding states. At this interaction strength, the synchronous map shows an oscillation between odd and even iterates, so the appropriate statement is that even-iterate commit decisions stabilize rapidly, not that arbitrary iterates converge monotonically. With $R = 2$, the commit set recovers 99.08% of the $R = 8$ commit set, agrees on 99 of 100 extracted answers, and matches 94 of 100 complete generations, with no accuracy gain from larger R (Table 16). We repeated the identical-state comparison on Dream and LLaDA Instruct for GSM8K and MBPP (25 prompts per model/task, 4,007 states in total). At $R = 2$, the commit-set Jaccard similarity with $R = 8$ ranges from 98.35% to 99.05% across the four model/task combinations, and the mean $|q^{(2)} - q^{(8)}|$ ranges from 0.0168 to 0.0226.

Table 16: Effect of the number of mean-field iterations R (`Dream-v0-Instruct-7B`, 100 GSM8K examples). Commit Jaccard and $|q^{(R)} - q^{(8)}|$ are computed against $R = 8$ on identical decoding states.

R	Accuracy (%)	Mean NFE	Commit Jaccard vs. R=8 (%)	Mean $\lvert q^{(R)} - q^{(8)} \rvert$	Tok/s
1	73.0	238.82	39.92	0.5653	36.4
2	74.0	49.66	99.08	0.0169	147.9
4	74.0	50.70	99.57	0.0077	147.3
8	74.0	51.00	100.00	0.0000	131.9

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 4

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Attached as a Supplementary

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars because evaluation is deterministic: for a fixed model, prompt, and decoding method, outputs are identical across runs. We evaluate on the full test split of each benchmark rather than sampled subsets, eliminating test-set sampling variance. As a result, repeated runs show negligible variation, and the reported differences reflect deterministic decoding behavior under identical conditions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work aims to advance diffusion llms research and we do not identify any specific societal impacts requiring additional discussion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Section 5

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.