DMax: Aggressive Parallel Decoding for dLLMs

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, Xinchao Wang*
National University of Singapore
zigeng99@u.nus.edu, xinchao@nus.edu.sg

Abstract

We present DMax, a new paradigm for efficient diffusion language models (dLLMs). It mitigates error accumulation in parallel decoding, enabling aggressive decoding parallelism while preserving generation quality. Unlike conventional masked dLLMs that decode through a binary mask-to-token transition, DMax reformulates decoding as a progressive self-refinement from mask embeddings to token embeddings. At the core of our approach is On-Policy Uniform Training, a novel training strategy that efficiently unifies masked and uniform dLLMs, equipping the model to recover clean tokens from both masked inputs and its own erroneous predictions. Building on this foundation, we further propose Soft Parallel Decoding. We represent each intermediate decoding state as an interpolation between the predicted token embedding and the mask embedding, enabling iterative self-revising in embedding space. Extensive experiments across a variety of benchmarks demonstrate the effectiveness of DMax. Compared with the original LLaDA-2.0-mini, our method improves TPF on GSM8K from 2.04 to 5.48 while preserving accuracy. On MBPP, it increases TPF from 2.71 to 5.86 while maintaining comparable performance. On two H200 GPUs, our model achieves an average of 1,338 TPS at batch size 1. Code is available at: https://github.com/czg1225/DMax

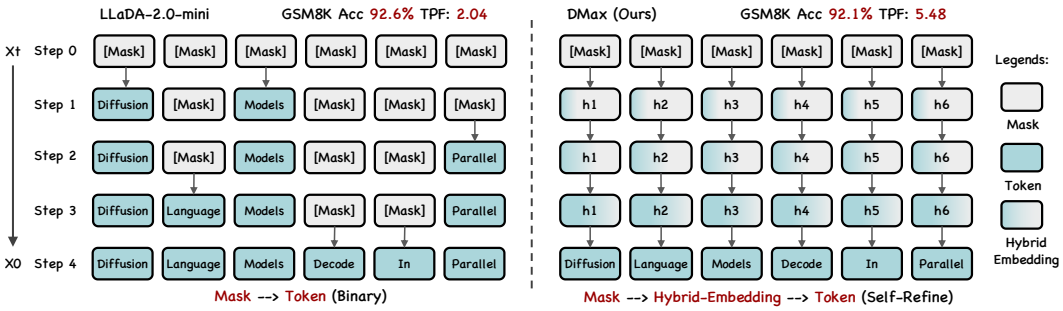


Figure 1: Comparison between the original LLaDA-2.0-mini and our proposed DMax. Unlike the original binary mask-to-token decoding process, DMax introduces a self-revising mask-to-hybrid-embedding-to-token process, enabling highly parallel decoding without accuracy dropping.

1 Introduction

Recently, Diffusion Language Models (dLLMs) [96, 101, 44, 58, 111, 98] have emerged as a compelling alternative to the long-standing dominance of Autoregressive Language Models (AR-LLM) [1, 6, 26] in text generation. The primary allure of dLLMs lies in their capacity for parallel decoding, which holds great promise for improving inference efficiency

*Corresponding Author

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

Despite this promise, the practical decoding parallelism of existing dLLMs [60, 112, 95, 17, 10, 93] remains limited, as their performance drops sharply under aggressive parallel decoding. Some prior work has attempted to improve this trade-off through improved decoding [38, 84, 43, 28, 8, 33, 35, 92] or distillation strategies [16, 64, 104, 40, 72]. Nevertheless, these methods do not address the fundamental bottleneck underlying parallel decoding in current dLLM paradigm: error accumulation.

In current mask-based dLLMs, decoding is a binary, one-way mask-to-token process. Once a masked position is decoded into a token, that token is fixed and propagated as context to subsequent decoding steps, with no opportunity for revision. Under highly parallel decoding, erroneous predictions are inevitable. Once such errors are committed, they contaminate future predictions and trigger cascading error accumulation, ultimately leading to semantic collapse. Unlike speculative decoding [41, 11, 45], dLLMs lack a mechanism to recover from incorrect predictions, which fundamentally restricts their performance under highly parallel decoding. Addressing this challenge requires a new dLLM paradigm with an intrinsic capability to revise its own predictions during decoding.

Building on this insight, we propose DMax, a novel paradigm that reformulates the binary mask-to-token decoding process into a self-revising transformation in the embedding space. Central to our approach is On-Policy Uniform Training (OPUT), a training recipe that efficiently extends a pretrained masked diffusion language model into a self-corrective uniform diffusion language model while preserving its original mask denoising capability. Unlike conventional uniform diffusion training that constructs noisy sequences by randomly sampling tokens from the vocabulary, OPUT samples noisy inputs on-policy from the model's own predictive distribution. This substantially bridges the train-inference gap and enables the model to effectively learn to correct its own potential prediction errors. Building upon OPUT, we further present Soft Parallel Decoding (SPD) for inference. Instead of treating decoded tokens as discrete and irrevocable commitments, SPD represents each intermediate decoding state as a hybrid soft embedding, formed by interpolating between the predicted token embedding and the mask embedding according to the model's prediction confidence. This simple design provides the model with confidence priors from previous steps, enabling more robust self-correction.

Using LLaDA-2.0-mini [10], a state-of-the-art open-source dLLM, as the base model, we validate the effectiveness of our method across multiple widely used benchmarks. On the mathematical reasoning benchmark GSM8K [20], our method increases tokens per forward (TPF) from 2.04 to 5.48 with only minimal accuracy degradation relative to the original model. On the code generation benchmark MBPP [5], it improves TPF from 2.71 to 5.86 while maintaining comparable performance.

In summary, we propose DMax, a novel paradigm that enables highly parallel decoding for dLLMs while preserving strong performance. Our central idea is to mitigate the error accumulation issue caused by the conventional one-way mask-to-token decoding. To realize this, we introduce two key designs: on-policy uniform training and soft parallel decoding. Extensive experiments demonstrate the effectiveness and superiority of our approach. This work establishes a new strong baseline for future research on parallel decoding in dLLMs.

2 Preliminaries

We begin by briefly reviewing the diffusion language modeling paradigms, and then highlight the central challenge for highly parallel decoding and introduce our key motivation.

Masked Diffusion Language Models (MDLMs). MDLMs [73, 4, 68, 108, 53] formulate text generation as a discrete denoising process over token sequences, where clean tokens are progressively replaced by a special `[MASK]` symbol during corruption. Let $x_0 = (x_0^1, \ldots, x_0^L) \in \mathcal{V}^L$ denote a clean sequence of length L, where $\mathcal{V}$ is the vocabulary. Given a corrupted sequence x_t at noise level $t \in [0, 1]$, the denoising model is trained to recover the original tokens only at masked positions. The standard MDLM objective is

$$\mathcal{L}_{\text{MDLM}}(\theta) = -\mathbb{E}_{x_0,\, t,\, x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}(x_t^i = \texttt{[MASK]})\log p_\theta(x_0^i \mid x_t)\right]. \tag{1}$$

At inference time, MDLMs start from a fully masked sequence and iteratively decode masked positions in parallel, with an optional remasking step to enable further refinement.

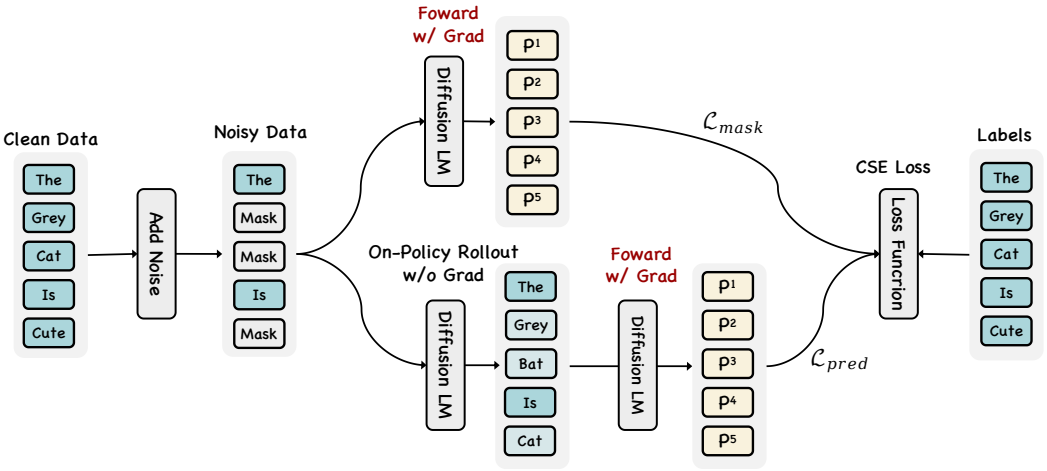


Figure 2: Overview of the proposed On-Policy Uniform Training.

Uniform Diffusion Language Models (UDLMs). UDLMs [70, 69, 71] generalize the corruption process by replacing tokens with uniformly sampled vocabulary tokens rather than a dedicated `[MASK]` symbol. As a result, the model is trained to recover clean tokens from arbitrary noisy token inputs, instead of only from masked positions. A standard UDLM training objective is

$$\mathcal{L}_{\text{UDLM}}(\theta) = -\mathbb{E}_{x_0,\, t,\, x_t}\left[\sum_{i=1}^{L} \log p_\theta(x_0^i \mid x_t)\right]. \tag{2}$$

During inference, UDLMs typically start from a fully noisy sequence sampled uniformly from the vocabulary and iteratively update all positions.

Error accumulation in MDLMs. Existing dLLMs based on the MDLM paradigm degrade sharply under highly parallel decoding, which limits the practical speedup. The main reason behind it is error accumulation. MDLM decoding follows a binary mask-to-token process: each position is either a mask token or a committed token. Once a masked position is decoded, its prediction is treated as fixed context for subsequent steps. Early mistakes cannot be revised, and instead propagate through later denoising steps as erroneous context.

UDLMs as a Promising Solution. In contrast, UDLMs are trained to denoise from arbitrary vocabulary tokens rather than only from `[MASK]`, so all positions can be re-evaluated at every decoding step. This token-to-token denoising mechanism naturally enables self-correction and improves robustness to prediction errors. However, UDLM decoding typically starts from a fully random sequence, which makes denoising harder and leads to very unstable generation.

Unify the Strengths of MDLMs and UDLMs. Motivated by this trade-off, we propose to unify the strengths of both paradigms. Specifically, we retain a fully masked sequence as the initialization of UDLM decoding to preserve stability, while continuing to re-predict all tokens that have been decoded from `[MASK]` at every subsequent step. This design combines the stable initialization with the self-revising capability, enabling a more robust parallel decoding process.

3 Methodology

3.1 On-Policy Uniform Training

A practical way to achieve this goal is to extend a pretrained MDLM into a UDLM. Accordingly, our first objective is to endow a pretrained MDLM with the self-revision capability of UDLMs while preserving its original mask denoising ability.

Extending MDLM toward UDLM is Nontrivial. This is nontrivial because the training objective of UDLMs differs substantially from that of MDLMs and is considerably harder to optimize. In the standard UDLM training paradigm, a clean sequence is first corrupted by randomly selecting a subset

of positions and replacing the selected tokens with tokens sampled uniformly from the vocabulary. The resulting noisy sequence is then used as model input, and the model is trained to recover the original clean sequence.

However, this training strategy is often unstable in practice and tends to yield suboptimal performance. A key reason is that uniformly sampled tokens lie far outside the natural language manifold, producing highly unnatural corrupted inputs. As a result, the model must spend substantial capacity merely learning to map these corrupted sequences back toward plausible language, rather than directly acquiring effective language modeling and self-correction behaviors. More importantly, this corruption process introduces a severe train–inference mismatch. Unlike conventional UDLMs, our paradigm first predicts tokens from masked positions in parallel and then iteratively refines its own predictions. Consequently, the noisy sequences encountered at inference time are sampled from the model's own output distribution rather than from a uniform vocabulary distribution. This mismatch hinders self-correction and leads to ineffective training.

On-Policy Uniform Training. To address these issues, we propose On-Policy Uniform Training (OPUT), a simple yet effective method for equipping MDLMs with self-corrective denoising capability. The core idea is to construct training inputs using noisy sequences sampled on-policy from the model's own predictive distribution, rather than from a uniform vocabulary distribution, thereby bridging the train–inference gap. The overview of the training procedure is shown in Figure 2.

Training Procedure. Let M_θ denote a pretrained diffusion language model built on the MDLM paradigm, parameterized by θ. We further adapt M_θ on a training dataset $\mathcal{D}$ of clean sequences $x_0 = (x_0^1, \ldots, x_0^L)$. At each training iteration, we first sample a corruption level $t \sim \mathrm{Uniform}(t_l, t_h)$, where t_l and t_h denote the lower and upper bounds of the noise level, respectively. Given a clean sequence $x_0 \sim \mathcal{D}$, we construct a masked noisy sequence $x_t^{(m)}$ by independently replacing each token with `[MASK]` with probability t.

We feed $x_t^{(m)}$ into M_θ and predict all masked positions in parallel. By sampling from the model's predictive distribution at masked positions, we obtain a predicted noisy sequence $x_t^{(p)}$, defined as

$$x_t^{(p),i} = \begin{cases} x_t^{(m),i}, & \text{if } x_t^{(m),i} \neq \texttt{[MASK]}, \\ \hat{x}^i, \quad \hat{x}^i \sim p_\theta(\cdot \mid x_t^{(m)}), & \text{if } x_t^{(m),i} = \texttt{[MASK]}. \end{cases} \tag{3}$$

Importantly, $x_t^{(p)}$ is sampled using the current model parameters at each iteration, making this a strictly on-policy rollout process.

Next, we perform two forward passes, using the masked noisy sequence $x_t^{(m)}$ and the predicted noisy sequence $x_t^{(p)}$ as inputs, respectively:

$$p_\theta^{(m)}(\cdot \mid x_t^{(m)}) = M_\theta(x_t^{(m)}), \qquad p_\theta^{(p)}(\cdot \mid x_t^{(p)}) = M_\theta(x_t^{(p)}). \tag{4}$$

We then supervise both outputs against the original clean sequence x_0 using cross-entropy loss over *all* token positions, regardless of whether a position is masked:

$$\mathcal{L}_{\text{mask}} = -\sum_{i=1}^{L} \log p_\theta^{(m)}(x_0^i \mid x_t^{(m)}), \qquad \mathcal{L}_{\text{pred}} = -\sum_{i=1}^{L} \log p_\theta^{(p)}(x_0^i \mid x_t^{(p)}). \tag{5}$$

The final training objective is

$$\mathcal{L}_{\text{on-policy}} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{pred}}, \tag{6}$$

By reducing the train–inference mismatch, the proposed OPUT strategy enables a pretrained MDLM to efficiently learn self-correction through limited post-training, while retaining its original mask denoising ability. As a result, the model can correct self-generated errors and effectively mitigate error accumulation under highly parallel decoding. On LLaDA-2.0-mini, our method improves GSM8K accuracy from 78% to 90% under confidence-threshold decoding with a threshold of 0.5, while also delivering faster decoding.

3.2 Soft Parallel Decoding

Although OPUT substantially mitigates error accumulation, it still struggles when many erroneous predictions arise simultaneously within a block. When many positions are decoded in parallel,

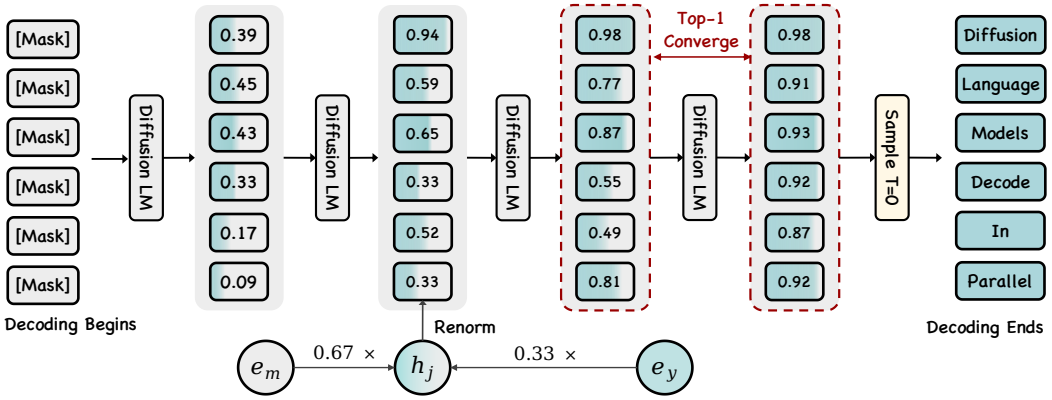


Figure 3: Overview of the Soft Parallel Decoding procedure in DMax.

correlated errors can appear at once, making them difficult to fully correct through iterative refinement. For example, for OPUT-trained LLaDA-2.0-mini, if we decode all masked positions in a block at once using a confidence threshold of 0, and then iteratively refine them, the accuracy on GSM8K drops to only 68%.

Soft Parallel Decoding. To further enhance self-revising in iterative refinement, we propose soft parallel decoding. The central idea is to preserve predictive uncertainty from earlier iterations and explicitly propagate it to later refinement steps. Concretely, instead of treating intermediate decoding states as discrete tokens, we represent each decoded token as a soft embedding interpolated between the predicted token embedding and the mask embedding. Because the mask embedding naturally encodes maximal uncertainty, this interpolation serves as an explicit carrier of uncertainty across iterations. This enables the model to better distinguish confident predictions from unreliable ones, allowing it to focus on refining low-confidence tokens while avoiding interference from noisy signals.

Decoding Procedure. An overview of the decoding process is shown in Figure 3. It follows a block-wise semi-autoregressive process. For each block, we partition its positions into two sets: *mask positions* and *token positions*. At initialization, all positions in the block are mask positions. At each decoding step, we use an aggressive confidence threshold τ_{dec} to promote some mask positions into token positions. Specifically, we scan the masked region from left to right and promote only its longest contiguous prefix whose confidence exceeds τ_{dec}. Once the first mask position with confidence below τ_{dec} is encountered, all mask positions to its right remain masked. If no mask position satisfies this criterion, we still promote the leftmost mask position to ensure decoding progress. This design keeps the masked region contiguous and prevents unreliable future tokens on the right from interfering with mask predictions on the left.

At decoding step t, every mask position uses the mask embedding as model input:

$$\mathbf{h}_j^{(t)} = \mathbf{e}_{\text{mask}}, \qquad j \in \mathcal{M}^{(t)}. \tag{7}$$

where $\mathcal{M}^{(t)}$ denotes the set of mask positions at step t.

For each token position $j \in \mathcal{T}^{(t)}$, where $\mathcal{T}^{(t)}$ is the set of token positions, we construct a hybrid embedding from the top-1 prediction at the previous step $t-1$ as the model input. Let $y_j^{(t-1)}$ denote the top-1 predicted token at position j, and let $\pi_j^{(t-1)}$ be its predicted probability. We assign the remaining probability mass to the mask embedding:

$$\pi_{j,\text{mask}}^{(t-1)} = 1 - \pi_j^{(t-1)}. \tag{8}$$

The unnormalized hybrid embedding is then

$$\tilde{\mathbf{h}}_j^{(t)} = \pi_j^{(t-1)}\, \mathbf{e}\left(y_j^{(t-1)}\right) + \pi_{j,\text{mask}}^{(t-1)}\, \mathbf{e}_{\text{mask}}, \qquad j \in \mathcal{T}^{(t)}. \tag{9}$$

Directly adding high-dimensional embeddings may distort their magnitude and lead to norm collapse. To avoid this issue, we renormalize the hybrid embedding so that its norm matches the probability-

Algorithm 1 Soft Parallel Decoding (Block-Wise)

Require: Block positions $\mathcal{B}$, decoding threshold τ_{dec}, acceptance threshold τ_{acc}

$\mathcal{M} \leftarrow \mathcal{B}$, $\mathcal{T} \leftarrow \emptyset$, $\mathbf{h}_j \leftarrow \mathbf{e}_{\text{mask}}\ \forall j \in \mathcal{B}$ ▷ initialize with fully masked block

repeat

  $p_j(\cdot) \leftarrow p_\theta(\cdot \mid \{\mathbf{h}_j\}_{j\in\mathcal{B}}),\ \forall j \in \mathcal{B}$

  $\hat{y}_j \leftarrow \arg\max_y p_j(y),\ \ c_j \leftarrow p_j(\hat{y}_j),\ \forall j \in \mathcal{B}$

  $\mathcal{P} \leftarrow$ longest contiguous prefix in $\mathcal{M}$ such that $c_j > \tau_{\text{dec}}$ for all $j \in \mathcal{P}$

  if $\mathcal{P} = \emptyset$ **then**

    $\mathcal{P} \leftarrow \{$leftmost position in $\mathcal{M}\}$

  end if

  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{P}, \qquad \mathcal{M} \leftarrow \mathcal{B} \setminus \mathcal{T}$

  $\mathbf{h}_j \leftarrow \mathbf{e}_{\text{mask}},\ \forall j \in \mathcal{M}$ ▷ Eq. (7)

  $\mathbf{h}_j \leftarrow \text{HybridEmb}(\hat{y}_j, c_j, \mathbf{e}_{\text{mask}}),\ \forall j \in \mathcal{T}$ ▷ Eqs. (8)–(10)

until $\hat{y}_j^{(t)} = \hat{y}_j^{(t-1)},\ \forall j \in \mathcal{B}$ **or** $\min_{j\in\mathcal{B}} c_j > \tau_{\text{acc}}$ ▷ block converges

return $\hat{y}_j,\ \forall j \in \mathcal{B}$ ▷ commit the block

weighted sum of the component norms:

$$\mathbf{h}_j^{(t)} = \frac{\tilde{\mathbf{h}}_j^{(t)}}{\left\|\tilde{\mathbf{h}}_j^{(t)}\right\|_2} \left(\pi_j^{(t-1)} \left\|\mathbf{e}\left(y_j^{(t-1)}\right)\right\|_2 + \pi_{j,\text{mask}}^{(t-1)} \left\|\mathbf{e}_{\text{mask}}\right\|_2\right). \tag{10}$$

This hybrid embedding serves as a soft intermediate state between decoding steps, explicitly carrying forward the uncertainty of previous predictions.

We regard a block as having converged to a stable state if either of the following conditions holds: (1) the top-1 predictions at all positions remain unchanged for two consecutive decoding steps, or (2) the confidence of every position in the block exceeds a high acceptance threshold τ_{acc}. Once a block converges, we commit all token positions in the block according to the final predictions and move on to the next block.

By interpolating prediction-mask embeddings as intermediate states, the model receives an explicit uncertainty prior before every forward pass, leading to substantially more robust parallel decoding. On OPUT-trained LLaDA-2.0-mini, under the highly aggressive setting of $\tau_{\text{dec}} = 0$, soft parallel decoding improves GSM8K accuracy from 68% to 90% while achieving a higher speedup.

OPUT as a Prerequisite. Notably, soft parallel decoding must be used together with OPUT-trained models. OPUT trains the model to recover the correct target not only from masked inputs, but also from its own sampled predictions. As a result, the model learns a consistent mapping from both mask embeddings and self-predicted token embeddings toward the correct output, which makes interpolation between them meaningful. In contrast, applying soft parallel decoding to a standard diffusion language model without OPUT leads to catastrophic performance collapse.

4 Experiments

4.1 Experimental Setup

Implementation Details. We build our method on LLaDA-2.0-mini [10], a state-of-the-art open-source diffusion language model. During training, we use OPUT with a fixed mask ratio of 0.75. We perform full-parameter fine-tuning for 2 epochs with a batch size of 8, an initial learning rate of 2×10^{-6}, and a cosine learning rate schedule. Training follows the block-diffusion setting with a block size of 32. To avoid extra memory overhead, the masked noisy sequence and the predicted noisy sequence are optimized in separate iterations within the same epoch, rather than jointly in a single iteration. Under this setup, we train two models: DMax-Math for mathematical reasoning and DMax-Coder for code generation tasks. All training runs are conducted on 8 H200 GPUs. At inference time, we adopt the proposed SPD decoding strategy under semi-autoregressive block diffusion with a block size of 32. The acceptance threshold for determining whether a block has converged to a stable state is set to $\tau_{\text{acc}} = 0.9$.

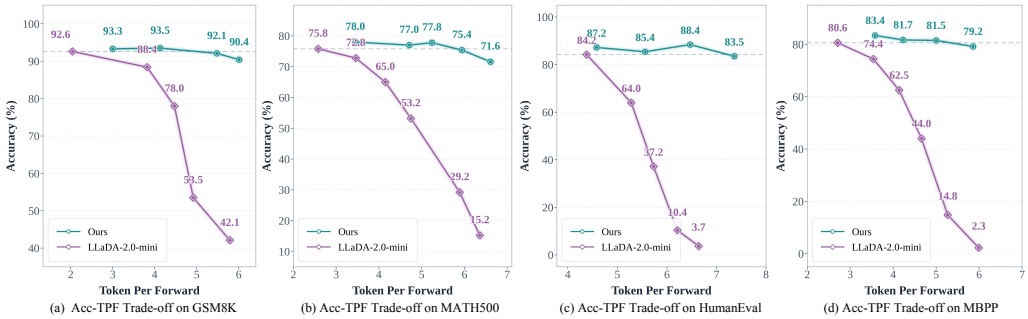
Table 1: Comparison with the original model and different baselines. For our DMax-Math model, we set the decoding threshold to 0.5; for the DMax-Coder model, we set it to 0.65. In addition to TPF, TPS, and accuracy, we also report the AUP score to provide a more comprehensive evaluation of parallel decoding performance. All evaluations are under zero-shot and a batch size of 1.

Benchmark	Method	TPF ↑	TPS ↑	Acc. ↑	AUP Score ↑
		Math & Reasoning Benchmarks			
GSM8K	LLaDA-2.0-mini	2.04	512	92.6%	340
	Hierarchical Decoding	2.44	577	91.6%	357
	dParallel SFT	2.79	721	92.3%	395
	Uniform Diffusion Training	2.26	493	68.7%	0
	DMax-Math	**5.48**	**1258**	92.1%	**557**
MATH500	LLaDA-2.0-mini	2.58	626	75.8%	257
	Hierarchical Decoding	3.01	669	73.0%	268
	dParallel SFT	3.42	823	75.8%	310
	Uniform Diffusion Training	2.43	530	33.6%	0
	DMax-Math	**5.94**	**1286**	75.4%	**507**
Minerva-Algebra	LLaDA-2.0-mini	3.01	755	91.4%	363
	Hierarchical Decoding	3.40	787	90.6%	382
	dParallel SFT	3.91	943	91.4%	430
	Uniform Diffusion Training	2.55	551	42.7%	0
	DMax-Math	**7.03**	**1492**	91.5%	**658**
ASDIV	LLaDA-2.0-mini	2.03	512	92.8%	354
	Hierarchical Decoding	2.43	528	92.5%	366
	dParallel SFT	2.72	663	93.0%	459
	Uniform Diffusion Training	2.51	515	80.8%	0
	DMax-Math	**5.62**	**1172**	92.5%	**556**
		Code Generation Benchmarks			
HumanEval-Instruct	LLaDA-2.0-mini	4.38	1044	84.2%	369
	Hierarchical Decoding	4.67	1014	81.1%	379
	dParallel SFT	5.12	1229	76.8%	394
	Uniform Diffusion Training	2.93	628	15.2%	0
	DMax-Coder	**7.36**	**1557**	83.5%	**637**
MBPP-Instruct	LLaDA-2.0-mini	2.71	662	80.6%	276
	Hierarchical Decoding	2.88	685	76.6%	241
	dParallel SFT	3.66	880	74.7%	273
	Uniform Diffusion Training	2.84	608	23.4%	0
	DMax-Coder	**5.86**	**1264**	79.2%	**482**

Training Data. We construct all training data through self-distillation. Specifically, we take prompts from public datasets and use LLaDA-2.0-mini to generate responses as training targets. For math, prompts are collected from GSM8K trainset [20], PRM12K [46], a subset of Numina-Math [42], and a subset of OpenThoughts [27]. For code, prompts are drawn from a subset of OpenCodeInstruct [2]. Responses are generated with a confidence threshold of 0.95, a block size of 32, and a maximum generation length of 2048 tokens. We discard incomplete generations that do not finish within the length budget. This yields 0.7M math samples and 1.0M code samples. Notably, we do not use any external high-quality responses, all supervision is obtained from the model's own generations.

Evaluation Details. We evaluate our method on multiple benchmarks. For mathematical reasoning, we use GSM8K [20], MATH500 [46], Minerva-Algebra [30], and ASDIV [57], and prompt the model to produce chain-of-thought [83] reasoning. For code generation, we use the instruction versions of HumanEval [13] and MBPP [5]. All evaluations are conducted with the dInFer [56] framework on 2 H200 GPUs using tensor parallelism. Besides TPF, TPS, and accuracy, we also report AUP Score [64] to measure parallel decoding performance. The generation length for all benchmarks is 2048.

Baselines. We compare our method against four baselines in terms of both decoding efficiency and generation accuracy: (1) LLaDA-2.0-mini, the base model, evaluated with its default confidence-threshold-based parallel decoding strategy using a threshold of 0.95; (2) Hierarchical Decoding, an advanced inference strategy that improves parallel decoding via a divide-and-conquer procedure [63]. The low threshold is set as 0.2; (3) dParallel-SFT, for which we use the LLaDA-2.0-mini-CAP model [10], where the certainty-forcing loss proposed in dParallel [16] is incorporated into large-scale



(a) Acc-TPF Trade-off on GSM8K (b) Acc-TPF Trade-off on MATH500 (c) Acc-TPF Trade-off on HumanEval (d) Acc-TPF Trade-off on MBPP

Figure 4: Comparison of accuracy-TPF trade-off curves between original LLaDA-2.0-mini model and our method. We present curves on GSM8K, MATH500, HumanEval and MBPP benchmarks.

Table 2: Our proposed new paradigm also improves the model's accuracy at low parallelism.

Benchmarks	LLaDA-2.0-mini		DMax	
	TPF ↑	Acc. ↑	TPF ↑	Acc. ↑
GSM8K	2.04	92.6%	3.54 (+1.50)	93.4% (+0.8%)
MATH500	2.58	75.8%	3.45 (+0.87)	78.0% (+2.2%)
Minerva-Algebra	3.01	91.4%	4.96 (+1.95)	93.6% (+2.2%)
ASDIV	2.03	92.8%	3.10 (+1.07)	93.5% (+0.7%)
HumanEval-Instruct	4.38	84.2%	4.58 (+0.20)	87.2% (+3.0%)
MBPP-Instruct	2.71	80.6%	3.58 (+0.87)	83.4% (+2.8%)

supervised fine-tuning to improve decoding parallelism; and (4) Uniform Diffusion Training, which continues training the base model using the conventional UDLM objective. In addition to masked noisy sequences, this baseline also replaces tokens with random vocabulary samples to construct uniformly corrupted noisy sequences, while keeping all other training settings identical to those of DMax. During inference, it updates all tokens within a block at every step until convergence.

4.2 Experimental Results

Aggressive Parallelism While Preserving Accuracy. As shown in Table 1, compared with the original LLaDA-2.0-mini, our method substantially increases decoding parallelism, improving the average TPF from 2.8 to 6.2 while preserving the original accuracy. In contrast, the other baselines provide only limited gains in parallel decoding. This advantage is further reflected in the AUP Score, where DMax consistently outperforms both the original model and all baselines by a large margin. These results demonstrate that our paradigm enables a much stronger parallel decoding capability than conventional MDLMs. Moreover, on two H200 GPUs, our model achieves a practical inference throughput of over 1000 tokens per second.

On-Policy Training as the Cornerstone. Table 1 also compares our method with conventional uniform diffusion training. The latter neither improves decoding speed nor preserves model quality, instead causing a noticeable performance drop. We find that this failure stems from the large mismatch between the randomly sampled noisy sequences used in training and the model's actual decoding trajectories at inference time. Consequently, the model struggles to revise erroneous predictions while unnecessarily perturbing correct ones, resulting in unstable oscillations within each block. By contrast, our on-policy training samples noisy sequences from the model's own outputs, effectively bridging this train–inference gap and substantially improving self-revision under parallel decoding.

Superior Efficiency–Performance Trade-off. Figure 4 compares the accuracy–TPF trade-off curves of our method and the original model on GSM8K, MATH500, HumanEval, and MBPP. As TPF increases, the original model suffers a sharp accuracy drop, whereas our method maintains stable performance. For instance, on MATH500, at around 6.5 TPF, our method still retains over 71.6% accuracy, while the original model falls to 15.2%. The gap is even larger on code benchmarks: on MBPP, at a similar TPF, our method achieves 79.2%, whereas the original model drops to only 2.3%. This superior trade-off stems from the self-revision capability of our paradigm, which effectively mitigates error accumulation under aggressive parallel decoding.

Table 3: Ablation on different training and inference strategies with different decoding parallelism.

Train	Inference		$\tau_{\text{dec}} = 0.95$		$\tau_{\text{dec}} = 0.50$		$\tau_{\text{dec}} = 0.0$	
On-Policy Rollout	Contiguous Prefix	Hybrid Embedding	TPF ↑	Acc. ↑	TPF ↑	Acc. ↑	TPF ↑	Acc. ↑
			2.04	92.6%	4.47	78.0%	7.86	0.9%
	✓	✓	1.04	0.0%	1.73	0.0%	5.39	0.0%
✓			2.95	92.6%	5.14	90.1%	5.89	68.2%
✓	✓		2.85	93.0%	5.28	91.3%	5.89	68.2 %
✓		✓	3.25	92.8%	5.64	91.4%	6.01	90.4%
✓	✓	✓	3.00	93.3%	5.48	92.1%	6.01	90.4%

Improved Performance at Low Parallelism. By enabling dLLMs to revise their own predictions, our method not only mitigates error accumulation under aggressive parallel decoding, but also improves performance in the low-parallelism regime. Through iterative re-evaluation of earlier predictions, the model can recover from reasoning errors that would otherwise remain on the original decoding path. As shown in Table 2, our method consistently improves accuracy by 0.8%–3.0% across multiple benchmarks at low parallelism. Importantly, these gains are obtained using only the model's own generated responses as training data, without introducing any external supervision.

5 Ablation Study

Ablation Study on Training and Inference Strategies. Table 3 presents a comprehensive ablation study of both our training and inference designs. We compare different combinations of training and decoding strategies on GSM8K under three decoding thresholds, $\tau_{\text{dec}} \in \{0.95, 0.5, 0.0\}$. On-policy rollout is the core of our training method. Even with OPUT alone, the model acquires the ability to revise its own errors, yielding substantial accuracy gains over the original model at $\tau_{\text{dec}} = 0.5$ and 0.0. Our proposed SPD further improves robustness when many erroneous predictions emerge simultaneously, allowing the model to remain stable under highly parallel decoding and to preserve strong performance even in the extreme case of $\tau_{\text{dec}} = 0.0$. The key ingredient of SPD is to use soft embeddings, rather than discrete tokens, as intermediate decoding states. Maintaining the non-masked region as a contiguous prefix further improves performance. Another important result is that OPUT is a prerequisite for SPD. As shown in Table 3, directly applying SPD to the original model causes generation to collapse. This is because OPUT trains the model to recover clean tokens from both mask tokens and predicted tokens, making interpolation between their embeddings a meaningful and effective input for denoising.

Ablation Study on Convergence Criteria. We further study in Table 4 how different block-level convergence criteria affect the efficiency–performance trade-off. We consider two criteria: (1) *consistency*, where decoding is considered converged if the model produces the same top-1 prediction for the block in two consecutive steps; and (2) *confidence*, where decoding is considered converged if the confidence of every token in the block exceeds 0.9. As shown in Table 4, consistency serves as the primary convergence signal, with most blocks terminating once this condition is met. Adding the confidence criterion can further improve TPF by allowing decoding to stop before two consecutive identical predictions are observed, thereby saving the final forward pass. Importantly, neither criterion affects the accuracy.

Table 4: Ablation study on block-level convergence criteria. The decoding threshold is set to 0.5 and 0.65.

Convergence		GSM8K		MBPP	
Consistency	Confidence	TPF ↑	Acc. ↑	TPF ↑	Acc. ↑
✓		5.13	92.1%	5.16	79.9%
	✓	2.28	92.2%	3.36	80.1%
✓	✓	5.48	92.1%	5.86	79.2%

6 Related Work

Diffusion Language Models. Diffusion models [32, 74] have become dominant in visual generation [66, 62, 67, 102], and recent work has explored their application to text generation. Among existing paradigms, masked diffusion language models (MDLMs) [73, 4, 68, 108, 53] have emerged as a promising alternative to AR-LLMs by modeling language in discrete space through masked token

prediction. Building on this formulation, LLaDA [60] and Dream [95] scale MDLMs to the billion-parameter regime with large-scale pretraining, demonstrating their practical potential. LLaDA-2.0 [10], LLaDA-MoE [113] and dMoE[22] further show that MDLMs can be effectively scaled with mixture-of-experts architectures. Beyond these developments, dLLMs are also attracting increasing attention in reasoning [112, 61, 89, 65, 77, 59, 106], multimodal tasks [97, 99, 93, 94, 50, 85, 100, 18], code generation [90, 25, 21], long-context modeling [49, 29, 109], and agent [107, 105].

Accelerating Diffusion Language Models. dLLMs are viewed as promising due to their potential for low-cost inference, yet their efficiency remains largely underexplored. Existing efforts improve efficiency from several perspectives. Some methods reduce the cost of each decoding step through techniques including KV caching [55, 51, 87, 36, 47], token dropping [15, 37, 75, 88], and sparse attention [82, 19]. Others design more effective decoding strategies [38, 84, 43, 28, 8, 33, 35, 92, 52, 12, 80, 63, 23] to improve generation efficiency. A separate line of work [76, 64, 104, 7, 14, 34] learns better decoding trajectories so that fewer decoding steps are required. dParallel [16] employs certainty-forcing distillation to accelerate confidence convergence and enable higher parallel decoding. Other methods [86, 17, 81, 3, 54, 78, 48, 24] interpolate between diffusion and autoregressive language models to better balance speed and accuracy. [9, 79, 103] implement uniform training, which trains the model to recover clean tokens from random noisy tokens, thereby enabling token correction during generation. SM [31] and EvoToken [110] introduce soft embeddings into the decoding process, but neither method translates this design into improved decoding efficiency. PRISM [39] introduces a lightweight adapter to learn per-token quality scores to enable inference-time remasking to correct tokens. Further efforts [91] leverage compression techniques to construct lightweight dLLMs.

7 Limitations and Future Work

Limitations. Our empirical study is conducted at a moderate scale due to computational constraints. Specifically, we evaluate our method on approximately 1M training samples (on the order of billions of tokens) using a 16B-scale diffusion language model. While this setup is sufficient to demonstrate the effectiveness of our approach across multiple benchmarks, we do not explore larger model scales (100B or beyond). As a result, the full potential of the proposed paradigm under large-scale regimes remains to be explored.

Future Work. An important direction for future work is to scale the proposed approach to significantly larger models and substantially larger training corpora, where scaling laws may further amplify the benefits of improved parallel decoding. In addition, extending the maximum sequence length and exploring long-context settings could unlock new efficiency gains and broaden the applicability of diffusion language models.

More broadly, we are interested in understanding how the proposed paradigm interacts with large-scale pretraining, including its potential when integrated into training-from-scratch pipelines. We believe such exploration may further improve both model quality and decoding efficiency, and we leave this as future work.

8 Conclusion

In this paper, we present DMax, a novel paradigm for efficient diffusion language models that mitigates error accumulation for parallel decoding. DMax enables aggressive decoding parallelism while preserving the accuracy of the original model. We introduce two key components of our approach, namely On-Policy Uniform Training and Soft Parallel Decoding, and demonstrate their effectiveness through extensive experiments on diverse benchmarks. Our results establish a strong new baseline for parallel decoding in dLLMs and suggest a promising new direction for dLLMs.

Acknowledgement

This project is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2026, WBS: A-8004345-00-00), the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity R&D Programme and CyberSG R&D Cyber Research Programme Office (Award: CRPO-GC1-NTU-002), as well as NUS IT's High Performance Cluster.

References

[1] 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.

[2] Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. Opencodeinstruct: A large-scale instruction tuning dataset for code llms. *arXiv preprint arXiv:2504.04030*, 2025.

[3] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

[4] 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.

[5] 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*, 2021.

[6] 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.

[7] Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via adaptive parallel decoding. In *The Fourteenth International Conference on Learning Representations*, 2025.

[8] Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 2025.

[9] Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

[10] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

[11] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

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

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

[14] Shirui Chen, Jiantao Jiao, Lillian J Ratliff, and Banghua Zhu. dultra: Ultra-fast diffusion language models via reinforcement learning. *arXiv preprint arXiv:2512.21446*, 2025.

[15] Xinhua Chen, Sitao Huang, Cong Guo, Chiyue Wei, Yintao He, Jianyi Zhang, Hai Li, Yiran Chen, et al. Dpad: Efficient diffusion language models with suffix dropout. *arXiv preprint arXiv:2508.14148*, 2025.

[16] Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms. *International Conference on Learning Representations*, 2026.

[17] Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

[18] Shuang Cheng, Yuhua Jiang, Zineng Zhou, Dawei Liu, Wang Tao, Linfeng Zhang, Biqing Qi, and Bowen Zhou. Sdar-vl: Stable and efficient block-wise diffusion for vision-language understanding. *arXiv preprint arXiv:2512.14068*, 2025.

[19] Alexandros Christoforos and Chadbourne Davis. Moe-diffuseq: Enhancing long-document diffusion models with sparse attention and mixture of experts. *arXiv preprint arXiv:2512.20604*, 2025.

[20] 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.

[21] Chenghao Fan, Wen Heng, Bo Li, Sichen Liu, Yuxuan Song, Jing Su, Xiaoye Qu, Kai Shen, and Wei Wei. Stable-diffcoder: Pushing the frontier of code diffusion large language model. *arXiv preprint arXiv:2601.15892*, 2026.

[22] Sicheng Feng, Zigeng Chen, Gongfan Fang, Xinyin Ma, and Xinchao Wang. dmoe: dllms with learnable block experts. In *Neural Information Processing Systems*, 2026.

[23] Sicheng Feng, Zigeng Chen, Xinyin Ma, Gongfan Fang, and Xinchao Wang. dvoting: Fast voting for dllms. *arXiv preprint arXiv:2602.12153*, 2026.

[24] Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, et al. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed. *arXiv preprint arXiv:2512.14067*, 2025.

[25] 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. *arXiv preprint arXiv:2506.20639*, 2025.

[26] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[27] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

[28] Daehoon Gwak, Minseo Jung, Junwoo Park, Minho Park, ChaeHun Park, Junha Hyung, and Jaegul Choo. Reward-weighted sampling: Enhancing non-autoregressive characteristics in masked diffusion llms. *arXiv preprint arXiv:2509.00707*, 2025.

[29] Guangxin He, Shen Nie, Fengqi Zhu, Yuankang Zhao, Tianyi Bai, Ran Yan, Jie Fu, Chongxuan Li, and Binhang Yuan. Ultrallada: Scaling the context length to 128k for diffusion large language models. *arXiv preprint arXiv:2510.10481*, 2025.

[30] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[31] Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. *arXiv preprint arXiv:2510.17206*, 2025.

[32] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[33] 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.

[34] Yanzhe Hu, Yijie Jin, Pengfei Liu, Kai Yu, and Zhijie Deng. Lightningrl: Breaking the accuracy-parallelism trade-off of block-wise dllms via reinforcement learning. *arXiv preprint arXiv:2603.13319*, 2026.

[35] Yuezhou Hu, Harman Singh, Monishwaran Maheswaran, Haocheng Xi, Coleman Hooper, Jintao Zhang, Aditya Tomar, Michael W Mahoney, Sewon Min, Mehrdad Farajtabar, et al. Residual context diffusion language models. *arXiv preprint arXiv:2601.22954*, 2026.

[36] Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S Abdelfattah, Jae-sun Seo, Zhiru Zhang, and Udit Gupta. Accelerating diffusion language model inference via efficient kv caching and guided diffusion. *arXiv preprint arXiv:2505.21467*, 2025.

[37] Jianuo Huang, Yaojie Zhang, Yicun Yang, Benhao Huang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. Mask tokens as prophet: Fine-grained cache eviction for efficient dllm inference. *arXiv preprint arXiv:2510.09309*, 2025.

[38] Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *arXiv preprint arXiv:2506.00413*, 2025.

[39] Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction. *arXiv preprint arXiv:2510.01384*, 2025.

[40] Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. Cdlm: Consistency diffusion language models for faster sampling. *arXiv preprint arXiv:2511.19269*, 2025.

[41] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[42] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

[43] Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Variable-length denoising for diffusion large language models. *arXiv e-prints*, pages arXiv–2508, 2025.

[44] Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*, 2025.

[45] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[46] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[47] Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. Wedlm: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025.

[48] Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. Tidar: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*, 2025.

[49] Xiaoran Liu, Yuerong Song, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Longllada: Unlocking long context capabilities in diffusion llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32186–32194, 2026.

[50] Yang Liu, Pengxiang Ding, Tengyue Jiang, Xudong Wang, Wenxuan Song, Minghui Lin, Han Zhao, Hongyin Zhang, Zifeng Zhuang, Wei Zhao, et al. Mmada-vla: Large diffusion vision-language-action model with unified multi-modal instruction and generation. *arXiv preprint arXiv:2603.25406*, 2026.

[51] Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

[52] Lingkun Long, Yushi Huang, Shihao Bai, Ruihao Gong, Jun Zhang, Ao Zhou, and Jianlei Yang. Focus-dllm: Accelerating long-context diffusion llm inference via confidence-guided context focusing. *arXiv preprint arXiv:2602.02159*, 2026.

[53] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. 2023.

[54] Linrui Ma, Yufei Cui, Kai Han, and Yunhe Wang. Diffusion in diffusion: Breaking the autoregressive bottleneck in block diffusion models. *arXiv preprint arXiv:2601.13599*, 2026.

[55] Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *Neural Information Processing Systems*, 2025.

[56] Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, et al. dinfer: An efficient inference framework for diffusion language models. *arXiv preprint arXiv:2510.08666*, 2025.

[57] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers, 2021.

[58] Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*, 2025.

[59] Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Why arbitrary order limits reasoning potential in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

[60] 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. *arXiv preprint arXiv:2502.09992*, 2025.

[61] Leyi Pan, Shuchang Tao, Yunpeng Zhai, Zheyu Fu, Liancheng Fang, Minghua He, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, et al. d-treerpo: Towards more reliable policy optimization for diffusion language models. *arXiv preprint arXiv:2512.09675*, 2025.

[62] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[63] Xiaojing Qi, Lun Du, Xinyuan Zhang, Lanning Wei, Tao Jin, and Da Zheng. Hierarchy decoding: A training-free parallel decoding strategy for diffusion large language models. In *The Fourteenth International Conference on Learning Representations*.

[64] Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*, 2026.

[65] Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. *arXiv preprint arXiv:2510.08554*, 2025.

[66] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[67] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

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

[69] Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *arXiv preprint arXiv:2506.10892*, 2025.

[70] Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Alexander Rush, Volodymyr Kuleshov, Hugo Dalla-Torre, Sam Boshar, Bernardo P de Almeida, and Thomas Pierrot. Simple guidance mechanisms for discrete diffusion models. In *... International Conference on Learning Representations*, volume 2025, page 44153, 2025.

[71] Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukic. Scaling beyond masked diffusion language models. *arXiv e-prints*, pages arXiv–2602, 2026.

[72] Qiuhong Shen, Xingyi Yang, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Speed: Sharpened-teacher distillation for parallel decoding of diffusion language models. In *International Conference on Machine Learning*, 2026.

[73] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[74] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[75] Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dllm: Accelerating diffusion llms with dynamic cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33038–33046, 2026.

[76] Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

[77] Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*, 2025.

[78] Yuchuan Tian, Yuchen Liang, Shuo Zhang, Yingte Shu, Guangwen Yang, Wei He, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, et al. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*, 2025.

[79] Dimitri Von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. *arXiv preprint arXiv:2503.04482*, 2025.

[80] Kangyu Wang, Zhiyun Jiang, Haibo Feng, Weijia Zhao, Lin Liu, Jianguo Li, Zhenzhong Lan, and Weiyao Lin. Creditdecoding: Accelerating parallel decoding in diffusion large language models with trace credits. *arXiv preprint arXiv:2510.06133*, 2025.

[81] Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*, 2025.

[82] Zeqing Wang, Gongfan Fang, Xinyin Ma, Xingyi Yang, and Xinchao Wang. Sparsed: Sparse attention for diffusion language models. In *International Conference on Learning Representations*, 2026.

[83] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[84] Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast: The three golden principles. *arXiv preprint arXiv:2506.10848*, 2025.

[85] Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. Llada-vla: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025.

[86] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025.

[87] 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.

[88] Zhongyu Xiao, Zhiwei Hao, Jianyuan Guo, Yong Luo, Jia Liu, Jie Xu, and Han Hu. Streaming-dllm: Accelerating diffusion llms via suffix pruning and dynamic decoding. *arXiv e-prints*, pages arXiv–2601, 2026.

[89] Shaoan Xie, Lingjing Kong, Xiangchen Song, Xinshuai Dong, Guangyi Chen, Eric P Xing, and Kun Zhang. Step-aware policy optimization for reasoning in diffusion large language models. *arXiv preprint arXiv:2510.01544*, 2025.

[90] Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025.

[91] Chen Xu and Dawei Yang. Dllmquant: Quantizing diffusion-based large language models. *arXiv preprint arXiv:2508.14090*, 2025.

[92] Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, et al. Lopa: Scaling dllm inference via lookahead parallel decoding. *arXiv preprint arXiv:2512.16229*, 2025.

[93] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

[94] Jiacheng Ye, Shansan Gong, Jiahui Gao, Junming Fan, Shuang Wu, Wei Bi, Haoli Bai, Lifeng Shang, and Lingpeng Kong. Dream-vl & dream-vla: Open vision-language and vision-language-action models with diffusion language model backbone. *arXiv preprint arXiv:2512.22615*, 2025.

[95] 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.

[96] Qiuhua Yi, Xiangfan Chen, Chenwei Zhang, Zehai Zhou, Linan Zhu, and Xiangjie Kong. Diffusion models in text generation: a survey. *PeerJ Computer Science*, 10:e1905, 2024.

[97] Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

[98] Runpeng Yu, Qi Li, and Xinchao Wang. Discrete diffusion in large language and multimodal models: A survey. *Transactions on Machine Learning Research*, 2026.

[99] Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion parallel generation for large multimodal modal. 2026.

[100] Lunbin Zeng, Jingfeng Yao, Bencheng Liao, Hongyuan Tao, Wenyu Liu, and Xinggang Wang. Diffusionvl: Translating any autoregressive models into diffusion vision language models. *arXiv preprint arXiv:2512.15713*, 2025.

[101] Lingzhe Zhang, Liancheng Fang, Chiming Duan, Minghua He, Leyi Pan, Pei Xiao, Shiyu Huang, Yunpeng Zhai, Xuming Hu, Philip S Yu, et al. A survey on parallel text generation: From parallel decoding to diffusion language models. *arXiv preprint arXiv:2508.08712*, 2025.

[102] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023.

[103] Shuibai Zhang, Fred Zhangzhi Peng, Yiheng Zhang, Jin Pan, and Grigorios G Chrysos. Corrective diffusion language models. *arXiv preprint arXiv:2512.15596*, 2025.

[104] Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Vladimir Pavlovic, et al. T3d: Few-step diffusion language models via trajectory self-distillation with direct discriminative optimization. *arXiv preprint arXiv:2602.12262*, 2026.

[105] Jiahao Zhao, Shaoxuan Xu, Zhongxiang Sun, Fengqi Zhu, Jingyang Ou, Yuling Shi, Chongxuan Li, Xiao Zhang, and Jun Xu. Dllm-searcher: Adapting diffusion large language model for search agents. *arXiv preprint arXiv:2602.07035*, 2026.

[106] 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.

[107] Huiling Zhen, Weizhe Lin, Renxi Liu, Kai Han, Yiming Li, Yuchuan Tian, Hanting Chen, Xiaoguang Li, Xiaosong Li, Chen Chen, et al. Dllm agent: See farther, run faster. *arXiv preprint arXiv:2602.07451*, 2026.

[108] Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

[109] Liang Zheng, Bowen Shi, Yitao Hu, Jiawei Zhang, Ruofan Li, Sheng Chen, Wenxin Li, and Keqiu Li. Mosaic: Unlocking long-context inference for diffusion llms via global memory planning and dynamic peak taming. *arXiv preprint arXiv:2601.06562*, 2026.

[110] Linhao Zhong, Linyu Wu, Bozhen Fang, Tianjian Feng, Chenchen Jing, Wen Wang, Jiaheng Zhang, Hao Chen, and Chunhua Shen. Beyond hard masks: Progressive token evolution for diffusion language models. *arXiv preprint arXiv:2601.07351*, 2026.

[111] Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling. *arXiv preprint arXiv:2602.22661*, 2026.

[112] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

[113] Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, et al. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025.

Appendix

A More Implementation Details

Training Configuration. All experiments are conducted within the dFactory framework, with minor modifications to accommodate our training algorithm. We adopt a block diffusion training paradigm. For each sample, the clean sequence is concatenated after the noisy sequence, and both share identical positional encodings. The attention mask is carefully designed such that each noisy block can attend to all tokens within the same block as well as all tokens in preceding blocks. We apply the training loss to all tokens in the noisy sequence, regardless of whether they are masked, while no loss is imposed on the prompt or the clean sequence. The block size is set to 32, and the maximum sequence length is 2048. The masking ratio is fixed at 0.75 throughout training. We use the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The base learning rate is 2×10^{-6}, with a cosine annealing schedule and a linear warmup phase comprising 3% of the total training steps. We set the weight decay to 0.01 and apply gradient clipping with a maximum norm of 1.0 to ensure training stability. Training is performed on $8\times$ NVIDIA H200 GPUs using the FSDP2 parallelization strategy with gradient checkpointing enabled. The global batch size is 8, with a per-GPU micro-batch size of 1. The model is trained for a total of two epochs.

AUP Score. To jointly evaluate generation quality and decoding efficiency, we adopt the *Accuracy Under Parallelism* (AUP) metric [64], which summarizes a model's performance along the accuracy–parallelism trade-off curve. Concretely, by varying a decoding control parameter (e.g., confidence or entropy threshold), one obtains a set of operating points $\{(\rho_i, y_i)\}_{i=1}^{m}$, where ρ_i denotes parallelism (measured by tokens per forward pass, TPF) and y_i denotes task accuracy. Rather than relying on a single operating point, AUP integrates performance over this curve, while explicitly penalizing regimes with significant accuracy degradation. Formally, we first discard points with $y_i < y_{\min}$, where $y_{\min}$ is defined relative to the highest-achieved accuracy to exclude low-quality regimes. The AUP score is then computed as a weighted area under the remaining curve:

$$\mathrm{AUP} = \rho_1 y_1 + \sum_{i=2}^{m} (\rho_i - \rho_{i-1}) \cdot \frac{y_i W(y_i) + y_{i-1} W(y_{i-1})}{2},$$

where $W(y)$ is a monotonically increasing weighting function that down-weights low-accuracy regions, typically defined as $W(y) = \min\Big(\exp\Big(-\alpha\left(1 - \frac{y}{y_{\max}}\right)\Big), 1\Big)$, with $y_{\max}$ being the best accuracy on the task and α controlling the penalty strength. Intuitively, AUP assigns substantial credit only when parallelism improves without sacrificing accuracy, while gains achieved through aggressive but unreliable decoding contribute marginally. This design makes AUP a robust, hardware-agnostic metric for assessing the overall effectiveness of parallel decoding strategies in diffusion language models, capturing both efficiency and fidelity in a unified manner.

In our experiments, we adopt the default setting $\alpha = 3.0$. For mathematical reasoning tasks, we set the minimum accuracy threshold as $y_{\min} = y_{\max} - 5$, while for code generation tasks, we use a more relaxed threshold $y_{\min} = y_{\max} - 10$.

B Controlled Comparison with Self-Distillation

To further investigate the source of DMax's performance improvements, we conduct a controlled comparison with a self-distillation baseline. Since our post-training data consist entirely of responses generated by the original LLaDA-2.0-mini model, an important question is whether the observed improvements arise primarily from additional training on self-generated data or from the proposed On-Policy Uniform Training (OPUT) and Soft Parallel Decoding (SPD).

Experimental Setup. We initialize both methods from the same pretrained LLaDA-2.0-mini checkpoint and perform post-training on an identical dataset comprising 0.7M self-generated mathematical reasoning examples and 1.0M self-generated code examples. Both methods use the same training configurations and number of epochs, ensuring comparable training budgets. We consider two settings: (1) **Self-Distill**, which continues training using the standard masked diffusion language modeling (MDLM) objective and performs inference with conventional confidence-threshold decoding; and (2) **DMax**, which employs our proposed OPUT objective for post-training and SPD for inference.

Table 5: Comparison between DMax and the self-distillation baseline under matched post-training data and comparable training budgets. Both methods are initialized from LLaDA-2.0-mini and trained on the same self-generated dataset. TPF measures decoding parallelism. The best accuracy in each setting is highlighted in bold.

Benchmark	τ_{dec}	Method	TPF ↑	Acc. ↑
MATH500	0.90	Self-Distill	4.07	76.0%
		DMax	4.26	**78.4%**
	0.50	Self-Distill	5.96	62.4%
		DMax	6.15	**76.4%**
GSM8K	0.90	Self-Distill	3.95	92.2%
		DMax	3.88	**93.4%**
	0.50	Self-Distill	5.98	82.5%
		DMax	5.75	**92.9%**
MBPP	0.90	Self-Distill	4.46	78.5%
		DMax	4.66	**82.7%**
	0.65	Self-Distill	5.78	49.7%
		DMax	6.13	**78.5%**

We evaluate both methods on MATH500, GSM8K, and MBPP under different decoding thresholds τ_{dec}. We report both tokens per forward pass (TPF) and task accuracy to assess the trade-off between decoding parallelism and generation quality.

Results and Analysis. As shown in Table 5, DMax consistently outperforms the self-distillation baseline in accuracy across all evaluated benchmarks and decoding thresholds. Importantly, this advantage becomes substantially more pronounced as decoding becomes increasingly aggressive.

On MATH500, DMax improves accuracy from 76.0% to 78.4% at $\tau_{\text{dec}} = 0.9$, while achieving slightly higher decoding parallelism. When the threshold is reduced to 0.5, the self-distillation baseline suffers a substantial accuracy degradation to 62.4%, whereas DMax maintains 76.4% accuracy at a comparable TPF of approximately 6. Similarly, on GSM8K, DMax achieves 92.9% accuracy at $\tau_{\text{dec}} = 0.5$, outperforming the self-distillation baseline by 10.4 percentage points while operating at similar decoding parallelism.

The improvement is particularly pronounced on code generation. On MBPP, reducing the decoding threshold to 0.65 causes the self-distillation baseline to degrade to 49.7% accuracy. In contrast, DMax retains 78.5% accuracy while achieving a higher TPF of 6.13, corresponding to an improvement of 28.8 percentage points.

These results suggest that conventional self-distillation alone is insufficient to address the error accumulation problem under aggressive parallel decoding. Although additional training on self-generated responses can improve model behavior, the standard MDLM objective does not explicitly train the model to recover from its own erroneous predictions. Consequently, once incorrect tokens are committed during conventional decoding, they may propagate errors into subsequent predictions.

In contrast, OPUT explicitly trains the model to denoise its own sampled predictions, while SPD enables iterative refinement through uncertainty-aware intermediate embeddings. Their combination substantially improves robustness to erroneous predictions, allowing DMax to maintain high generation quality at considerably greater decoding parallelism.

C Quantitative Analysis of Self-Correction

A central motivation of DMax is to overcome the error accumulation problem inherent in conventional masked diffusion language models. Unlike standard mask-to-token decoding, where decoded tokens are treated as fixed context, DMax allows previously predicted tokens to be repeatedly revised during the decoding process. To directly investigate this mechanism, we conduct additional experiments

Table 6: Quantitative analysis of self-correction on MATH500 under different decoding thresholds. Correction counts measure the total number of token prediction changes per sample, while correction positions count the number of unique positions revised at least once. Disabling self-correction substantially degrades accuracy, particularly under aggressive decoding. The best accuracy at each threshold is highlighted in bold.

τ_{dec}	Method	Correction Counts / Sample	Correction Positions / Sample	Acc. ↑
0.90	Original LLaDA-2.0-mini	0	0	75.8%
	DMax w/o Self-Correction	0	0	75.4%
	DMax	3.26	2.33	**78.4%**
0.50	Original LLaDA-2.0-mini	0	0	52.2%
	DMax w/o Self-Correction	0	0	63.6%
	DMax	25.37	20.79	**76.4%**
0.00	Original LLaDA-2.0-mini	0	0	4.0%
	DMax w/o Self-Correction	0	0	6.0%
	DMax	359.73	228.30	**71.2%**

measuring token-level revisions and evaluating the effect of disabling self-correction at inference time.

Experimental Setup. We evaluate DMax on MATH500 under three decoding thresholds, $\tau_{\text{dec}} \in \{0.9, 0.5, 0.0\}$, representing increasingly aggressive decoding parallelism. To quantify the self-correction behavior, we record two statistics for each generated sample: (1) **Self-Correction Counts**, defined as the total number of changes in token predictions across consecutive decoding steps; and (2) **Self-Correction Positions**, defined as the number of distinct token positions whose predictions change at least once during decoding. The distinction between these metrics is important, a single token position may be revised multiple times, contributing multiple correction events but only one unique corrected position.

We further compare three decoding configurations: (1) **Original LLaDA-2.0-mini**, which uses the pretrained model with conventional masked diffusion decoding; (2) **DMax w/o Self-Correction**, which uses the OPUT-trained DMax checkpoint but disables token revision and adopts conventional mask-to-token decoding; and (3) **DMax**, which employs the complete OPUT-trained model with the proposed SPD decoding procedure. This comparison allows us to examine both the frequency of token revisions and their contribution to generation accuracy.

Self-Correction Behavior. Table 6 provides direct quantitative evidence that DMax actively revises its predictions during decoding. At the relatively conservative threshold of $\tau_{\text{dec}} = 0.9$, DMax performs an average of 3.26 token revisions across 2.33 distinct positions per sample. As the threshold decreases to 0.5, these values increase to 25.37 revisions across 20.79 positions.

Under the most aggressive setting of $\tau_{\text{dec}} = 0.0$, DMax performs an average of 359.73 token revisions spanning 228.30 unique positions per sample. This substantial increase indicates that highly parallel decoding introduces considerably more prediction changes, making iterative refinement particularly important. Moreover, the number of revision events consistently exceeds the number of unique revised positions. This confirms that some positions undergo multiple revisions rather than being updated only once, illustrating the iterative nature of the proposed self-refinement process.

Effect of Disabling Self-Correction. To further examine the contribution of self-correction, we disable token revision at inference time while retaining the OPUT-trained model. As shown in Table 6, this modification causes substantial accuracy degradation, especially at lower decoding thresholds.

At $\tau_{\text{dec}} = 0.9$, DMax achieves 78.4% accuracy, compared with 75.4% without self-correction. The gap widens considerably as decoding becomes more aggressive. At $\tau_{\text{dec}} = 0.5$, disabling self-correction reduces accuracy from 76.4% to 63.6%, corresponding to a drop of 12.8 percentage points.

The difference becomes particularly striking at $\tau_{\text{dec}} = 0.0$, where all masked positions can be decoded aggressively. Conventional masked decoding achieves only 4.0% accuracy with the original LLaDA-2.0-mini model and 6.0% with the OPUT-trained DMax checkpoint. In contrast, full DMax maintains 71.2% accuracy, outperforming its non-corrective counterpart by 65.2 percentage points.

These results reveal two complementary effects. First, OPUT improves robustness even when self-correction is disabled, as reflected by the accuracy improvement from 52.2% to 63.6% at $\tau_{\text{dec}} = 0.5$. Second, and more importantly, enabling iterative self-correction yields substantial additional improvements, particularly in the high-parallelism regime. This observation supports our central hypothesis that the inability to revise erroneous predictions is a major bottleneck in conventional masked diffusion decoding.

D Discussion

Is it beneficial to introduce soft embeddings during training? During decoding, our method employs soft parallel decoding, where the previous-step confidence is injected as a prior by interpolating between the `[MASK]` embedding and the predicted token embedding according to the confidence gap. A natural question is whether exposing the model to such soft embeddings during training could further improve performance.

We conduct a comprehensive study by representing the predicted tokens from on-policy rollouts as soft embeddings during training, while keeping the learning objective unchanged. Surprisingly, this strategy does not improve performance; instead, it reduces decoding parallelism. We hypothesize that soft embeddings introduce excessive mask-like noise into the context, making the model treat the entire context as unreliable. As a result, the model becomes less confident when predicting pure `[MASK]` tokens. Empirically, we observe a substantial drop in mask-token prediction confidence, which sharply reduces the number of tokens exceeding the confidence threshold at each step. Consequently, even with a threshold of $\tau_{\text{dec}} = 0.5$, this variant achieves only 3.5 TPF on GSM8K, whereas the original DMax reaches 5.5 TPF.

E Case Study

We present qualitative case studies to further illustrate the effectiveness of our approach. Figure 5 showcases representative examples on mathematical reasoning tasks, while Figure 6 provides examples on code generation tasks. Compared to the original LLaDA-2.0-mini model, our method achieves substantially higher decoding parallelism. Importantly, this improvement in efficiency does not come at the cost of generation quality. The outputs remain coherent, logically consistent, and correct. These examples highlight that our approach effectively mitigates error accumulation under aggressive parallel decoding, enabling both high efficiency and reliable generation.

F Social Impacts

Our work improves the efficiency of diffusion language models by enabling highly parallel decoding while preserving generation quality, which can substantially reduce inference latency and computational cost. This may facilitate broader accessibility of large language models in resource-constrained environments and enable real-time applications at scale.

From an environmental perspective, improved decoding efficiency can reduce energy consumption during inference, contributing positively to sustainability. Nevertheless, this benefit may be offset if the technology leads to significantly increased overall usage. We emphasize that responsible deployment, including content moderation, usage restrictions, and alignment with human values, remains essential when applying such systems in real-world scenarios.

Prompt: Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? /nLet's think step by step/n

LLaDA-2.0-mini **# TPF: 1.84**

Let's break this down step by step:

1. **Eggs laid per day**:
   Janet’s ducks lay **16 eggs per day**.

2. **Eggs Janet eats**:
   - She eats **3 eggs for breakfast** every morning.
   - She uses **4 eggs to bake muffins** every day.
   So, total eggs used per day = 3 + 4 = **7 eggs**.

3. **Eggs sold at the farmers' market**:
   Total eggs laid - eggs used = 16 - 7 = **9 eggs**.

4. **Price per egg**:
   She sells each egg for **$2**.

5. **Money made per day**:
   9 eggs × $2 per egg = **$18**.

   So, Janet makes **$18 every day** at the farmers' market.

Answer: $18.

DMax (Ours) **# TPF: 5.45**

Let's break this down step by step:

1. **Eggs laid per day**:
   Janet’s ducks lay **16 eggs per day**.

2. **Eggs eaten for breakfast**:
   She eats **3 eggs every morning**.

3. **Eggs used for muffins**:
   She bakes **4 eggs every day** for her friends.

4. **Total eggs used per day**:
   3 (breakfast) + 4 (muffins) = **7 eggs per day**

5. **Eggs left to sell at the farmers' market**:
   16 (total) - 7 (used) = **9 eggs per day**

6. **Price per egg**:
   She sells each egg for **$2**.

7. **Daily earnings at the farmers' market**:
   9 eggs × $2 per egg = **$18**

   So, Janet makes **$18 every day** at the farmers' market.

Prompt: What is the degree of the polynomial $(4 +5x^3 +100 +2\pi x^4 + \sqrt{10}x^4 +9)$? /nLet's think step by step/n

LLaDA-2.0-mini **# TPF: 2.96**

We are given the polynomial:

$$
4 + 5x^3 + 100 + 2\pi x^4 + \sqrt{10}x^4 + 9
$$

Step 1: Combine like terms
First, combine the constant terms:
$$
4 + 100 + 9 = 113
$$

Now look at the terms involving x:
- $5x^3$
- $2\pi x^4$
- $\sqrt{10}x^4$

Combine the x^4 terms:
$$
2\pi x^4 + \sqrt{10}x^4 = (2\pi + \sqrt{10})x^4
$$

So the simplified polynomial is:
$$
(2\pi + \sqrt{10})x^4 + 5x^3 + 113
$$

Step 2: Determine the degree
The **degree** of a polynomial is the highest power of x with a non-zero coefficient.

Here, the highest power is x^4, so the degree is:

$$
\boxed{4}
$$

DMax (Ours) **# TPF: 7.31**

We are given the polynomial:

$$
4 + 5x^3 + 100 + 2\pi x^4 + \sqrt{10}x^4 + 9
$$

Step 1: Combine like terms
First, combine the constant terms:
$$
4 + 100 + 9 = 113
$$

Now look at the terms with x:
- $5x^3$
- $2\pi x^4$
- $\sqrt{10}x^4$

Combine the x^4 terms:
$$
2\pi x^4 + \sqrt{10}x^4 = (2\pi + \sqrt{10})x^4
$$

So the polynomial becomes:
$$
(2\pi + \sqrt{10})x^4 + 5x^3 + 113
$$

Step 2: Determine the degree
The **degree** of a polynomial is the highest power of x with a non-zero coefficient.

Here, the highest power of x is x^4, and its coefficient is $(2\pi + \sqrt{10}) \neq 0$.

Final Answer:
$$
\boxed{4}
$$

Figure 5: Case study on math reasoning tasks.

Prompt: Write a python function to remove first and last occurrence of a given character from the string. \n\nPlease enclose your code within delimiters as follows:\n```python\n# YOUR CODE HERE\n```\n\n

LLaDA-2.0-mini **# TPF: 2.75**

```python
def remove_first_last_occurrence(string, char):
    """
    Remove the first and last occurrence of a given character from the string.

    Args:
        string (str): The input string
        char (str): The character to remove

    Returns:
        str: The string with first and last occurrence of char removed
    """
    if not string or not char:
        return string

    # Find the first occurrence
    first_index = string.find(char)

    # Find the last occurrence
    last_index = string.rfind(char)

    # If character not found, return original string
    if first_index == -1:
        return string

    # If first and last occurrence are the same (only one occurrence)
    if first_index == last_index:
        return string[:first_index] + string[first_index + 1:]

    # Remove both occurrences
    # Remove last occurrence first (to avoid index shifting)
    temp_string = string[:last_index] + string[last_index + 1:]
    result = temp_string[:first_index] + temp_string[first_index + 1:]
```

DMax (Ours) **# TPF: 7.14**

```python
def remove_first_last_occurrence(s, char):
    """
    Remove the first and last occurrence of a given character from the string.

    Args:
        s (str): The input string
        char (str): The character to remove

    Returns:
        str: The string with first and last occurrence of the character removed
    """
    # Find the index of the first occurrence
    first_index = s.find(char)

    # Find the index of the last occurrence
    last_index = s.rfind(char)

    # If character doesn't exist, return the original string
    if first_index == -1:
        return s

    # If character appears only once, remove it
    if first_index == last_index:
        return s[:first_index] + s[first_index + 1:]

    # Remove both first and last occurrence
    # Remove last occurrence first to avoid index shifting
    result = s[:last_index] + s[last_index + 1:]
    result = result[:first_index] + result[first_index + 1:]

    return result
```

Prompt: Write a function to sort a given matrix in ascending order according to the sum of its rows. \n\nPlease enclose your code within delimiters as follows:\n```python\n# YOUR CODE HERE\n```\n\n

LLaDA-2.0-mini **# TPF: 3.11**

```python
def sort_matrix_by_row_sum(matrix):
    """
    Sort a given matrix in ascending order according to the sum of its rows.

    Args:
        matrix: A 2D list representing the matrix

    Returns:
        A new matrix with rows sorted by their sums in ascending order
    """
    if not matrix:
        return []

    # Sort the matrix based on the sum of each row
    sorted_matrix = sorted(matrix, key=sum)
    return sorted_matrix
```

DMax (Ours) **# TPF: 7.37**

```python
def sort_matrix_by_row_sum(matrix):
    """
    Sort a matrix in ascending order according to the sum of its rows.

    Args:
        matrix: A 2D list representing the matrix

    Returns:
        A new matrix with rows sorted by their sum in ascending order
    """
    if not matrix:
        return []

    # Sort the matrix based on the sum of each row
    sorted_matrix = sorted(matrix, key=sum)
    return sorted_matrix

# Example usage:
# matrix = [[3, 1, 2], [1, 5, 1], [2, 2, 2]]
# result = sort_matrix_by_row_sum(matrix)
# print(result)  # [[1, 5, 1], [2, 2, 2], [3, 1, 2]]
```

Figure 6: Case study on code generation tasks.

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: The main claims made in the abstract and introduction accurately reflect our paper's contributions and scope.

   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: We discuss the limitations of the work in the appendix.

   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: The paper does not include theoretical results.

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: The paper fully discloses 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.

   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: We provide open access to the data and code with sufficient instructions to faithfully reproduce the main experimental results.

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: The paper specifies all the training and test details

   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: [Yes]

   Justification: The paper reports the statistical significance of the experiments.

   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: The paper provides sufficient information on the computer resources

   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: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   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: [Yes]

    Justification: We discuss societal impacts in the appendix.

    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: The paper poses no such risks.

    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: All original creators or owners of the assets used in this paper have been properly credited, the license is clearly stated in the original credited asset, and the terms of use have been fully respected.

    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: [Yes]

    Justification: New assets introduced in the paper are well-documented, and the documentation is provided alongside the assets

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: [Yes]

    Justification: The core contribution is a novel parallel decoding algorithm for LLMs.

    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.