REJECTION MIXING: FAST SEMANTIC PROPAGATION OF MASK TOKENS FOR EFFICIENT DLLM INFERENCE

Yushi Ye[1,*], **Feng Hong**[1,*], **Huangjie Zheng, Xu Chen**[1]
Zhiyong Chen[1], **Yanfeng Wang**[2], **Jiangchao Yao**[1,†]
[1]Cooperative Medianet Innovation Center, Shanghai Jiao Tong University
[2]School of Artificial Intelligence, Shanghai Jiao Tong University
{stephen-ye, Sunarker}@sjtu.edu.cn

ABSTRACT

Diffusion Large Language Models (DLLMs) promise fast non-autoregressive inference but suffer a severe quality-speed trade-off in parallel decoding. This stems from the "combinatorial contradiction" phenomenon, where parallel tokens form semantically inconsistent combinations. We address this by integrating continuous representations into the discrete decoding process, as they preserve rich interposition dependency. We propose ReMix (Rejection Mixing), a framework that introduces a novel Continuous Mixing State as an intermediate between the initial masked state and the final decoded token state. This intermediate state allows a token's representation to be iteratively refined in a continuous space, resolving mutual conflicts with other tokens before collapsing into a final discrete sample. Furthermore, a rejection rule reverts uncertain representations from the continuous state back to the masked state for reprocessing, ensuring stability and preventing error propagation. ReMix thus mitigates combinatorial contradictions by enabling continuous-space refinement during discrete diffusion decoding. Extensive experiments demonstrate that ReMix, as a training-free method, achieves a 2-8× inference speedup without any quality degradation.[1]

1 INTRODUCTION

The proliferation of Large Language Models (LLMs) (Zhao et al., 2023; Touvron et al., 2023; Chowdhery et al., 2023) marks a pivotal moment in artificial intelligence. Yet their predominant autoregressive (AR) architecture, which generates outputs in a token-by-token sequence, severely limits inference speed. To address this, Diffusion Large Language Models (DLLMs) (Li et al., 2025b; Nie et al., 2025; Ye et al., 2025; Gong et al., 2025; Tian et al., 2025) have emerged as a compelling non-autoregressive paradigm. By modelling entire sequences of tokens in parallel, DLLMs bypass the inherent dependencies of AR models, offering the potential for a substantial leap in generative throughput and establishing them as a critical frontier for developing high-efficiency language and multimodal systems (Yang et al., 2025; Li et al., 2025a; You et al., 2025).

This parallel promise, however, is undermined by a critical dilemma: a severe quality-speed trade-off. They achieve optimal performance when decoding a single token at a time, but the quality degrades significantly when attempting to decode multiple tokens in parallel (Nie et al., 2025). This trade-off is a direct symptom of a fundamental problem we term the "combinatorial contradiction", where tokens sampled within the same decoding step can be mutually contradictory. The issue is exemplified by Song & Zhou (2025): "The list of poker hands that consist of two English words are: _ _." While valid subsequent pairs include "high card", "two pair", "full house", or "straight flush", a direct parallel approach might independently select the most probable token for each position, resulting in a semantically incorrect combination like "high house". Although sophisticated parallel decoding schedules (Wu et al., 2025; Ben-Hamu et al., 2025; Wei et al., 2025; Bao et al., 2025; Chen et al., 2025) have improved the trade-off between quality and speed, they do not fully resolve the core consistency problem. WINO (Hong et al., 2025) mitigates this issue through a revocable strategy, but

[*]Equal contribution. [†]Corresponding author.
[1]Code: https://github.com/Serpientw/ReMix-DLLM.

at the cost of introducing an additional verification block and its associated computational overhead. APD (Israel et al., 2025), on the other hand, addresses this challenge by defining a multiplicative mixture between the DLLM marginal probabilities and the joint probability of sequences under a small auxiliary AR model, a method that, while effective, requires additional computation and enforces a left-to-right generation order.

In existing DLLMs, "combinatorial contradiction" arises from the discrete decoding nature: tokens generated within the same step cannot perceive one another and are sampled independently (Liu et al., 2025a). Continuous representations offer a way out of this limitation: they enable the model to capture dependencies across different positions in the sequence and preserve richer information throughout the decoding process (Zheng et al., 2025; Zhou et al., 2025; Pynadath et al., 2025). To address combinatorial contradiction, we propose a novel approach that introduces continuous representations into the decoding process. By retaining continuous representations of unfilled positions throughout the decoding process, rather than remasking them and discarding valuable information, we preserve the interdependencies between positions in the sequence. This allows the model to iteratively refine the continuous representations over multiple decoding steps, resolving mutual contradictions and ensuring combinatorial consistency.

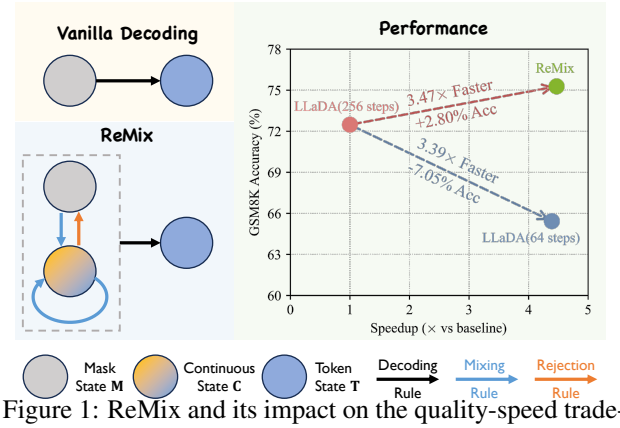


Figure 1: ReMix and its impact on the quality-speed trade-off. (Left) By introducing an intermediate Continuous State ($\mathbf{C}$), ReMix extends the standard $\mathbf{M} \to T$ transition by introducing a flexible loop involving Mixing $\mathbf{M} \to \mathbf{C}$, self-transition $\mathbf{C} \circlearrowleft$ and Rejection $\mathbf{C} \to \mathbf{M}$, enabling in-process refinement. (Right) On GSM8K, ReMix demonstrates superior performance, turning the baseline's 7.05% accuracy loss (when accelerating 3.39x) into a 2.80% accuracy gain while achieving similar acceleration.

Building on this intuition, we propose ReMix (Rejection Mixing), a method that integrates continuous states into the discrete diffusion decoding process to address the "combinatorial contradiction" (see Fig. 1). In the standard decoding process, each token position transitions from a [MASK] state to a semantic token. Traditional methods use predefined rules to control this transition. ReMix introduces an intermediate continuous state, allowing token representations to be refined before finalizing them as discrete tokens. To ensure stability, we also incorporate a rejection mechanism that reverts uncertain representations back to the [MASK] state for reprocessing, preventing early uncertainty from affecting the decoding process. Our main contributions are summarized as follows:

- We analyze a fundamental bottleneck in the parallel decoding of DLLMs, which we term the "combinatorial contradiction". We posit that this issue stems from the purely discrete nature of the decoding process and propose a novel approach to resolve it by integrating continuous representations into the discrete diffusion loop.
- We propose ReMix (Rejection Mixing), a novel, training-free decoding framework. ReMix introduces a Continuous Mixing State as an intermediate step, enabling iterative refinement and ensuring stability through a rejection mechanism. This approach allows for resolving mutual dependencies in a continuous space before discretization, improving the overall decoding process.
- We conduct extensive experiments on both language and multimodal generation benchmarks. Our results demonstrate that ReMix successfully breaks the severe quality-speed trade-off, achieving a 2-8× inference speedup over strong baselines without any quality degradation, and in several cases, even improving output quality.

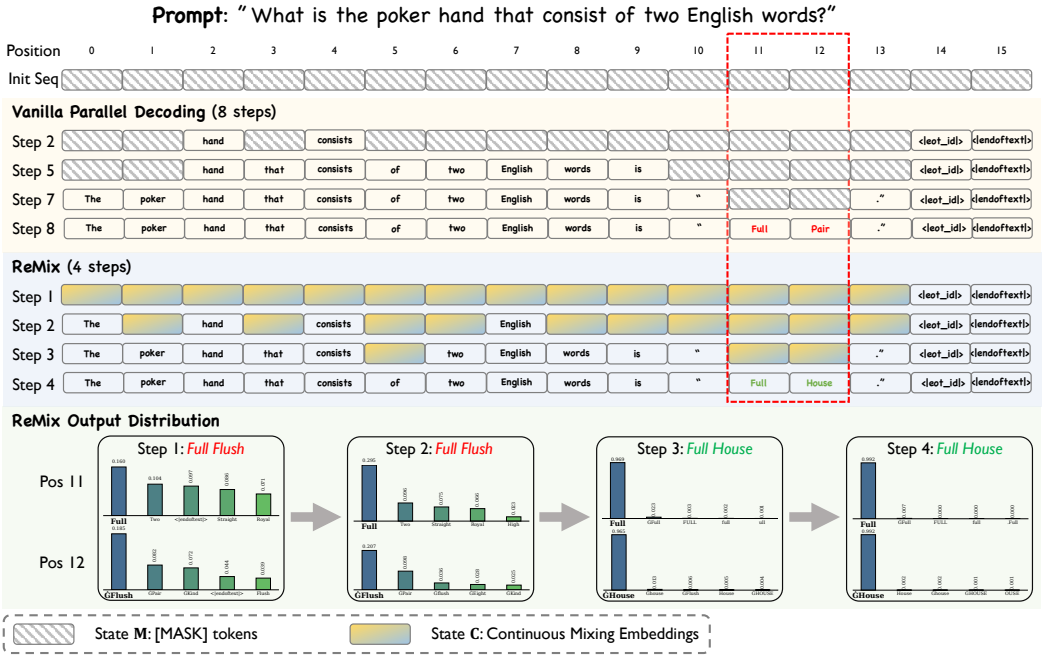


Figure 2: ReMix resolves combinatorial contradictions by allowing continuous-space refinement. Using the poker hand example, this figure compares the decoding processes. (Top) Vanilla Parallel Decoding: Succumbs to the contradiction. It independently decodes "Full" and "Pair" at positions 11 and 12, resulting in semantically incorrect "Full Pair" in 8 steps. (Middle) ReMix: Correctly generates "Full House" in only 4 steps. (Bottom) ReMix Output Distribution: This visualizes the internal mechanism. In Steps 1 & 2, both positions are in the Continuous Mixing State (State **C**, gradient). Their output distributions are iteratively refined, allowing the model to recognize the "Full" ↔ "House" dependency. This mutual coordination before discretization prevents the error. The Ġ symbol (*e.g.* in ĠFlush) indicates a token that begins with a whitespace.

2 Related Work

2.1 Diffusion-based Large Language Model

Diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2021; Ho et al., 2020), initially gaining prominence for image generation (Dhariwal & Nichol, 2021; Rombach et al., 2022; Nichol et al., 2022), have recently emerged as a competitive alternative to AR language models in text generation. This transition from continuous to discrete data domains necessitates new modeling approaches. The foundational work by Sohl-Dickstein et al. (2015) first explored this direction, which was later rigorously formalized by Austin et al. (2021a) in the D3PM framework. D3PM conceptualizes the diffusion forward process as a discrete-state Markov chain with a series of transition matrices, which is later extended by Campbell et al. (2022) using the theory of continuous-time Markov chains.

A particularly influential thread of research, known as masked diffusion models (MDMs), builds upon the absorbing-state diffusion concept in D3PM. As MDMs have evolved, they have demonstrated compelling performance across scales, from compact architectures like SEDD (Lou et al., 2024), MDLM (Sahoo et al., 2024) and RADD (Ou et al., 2025) to large-scale implementations such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025). Most recently, this line of work has been expanded to multimodal learning. MMaDA (Yang et al., 2025), for instance, introduces a family of multimodal large diffusion models featuring a shared probabilistic formulation and a modality-agnostic architecture, underscoring the versatility of the diffusion framework.

2.2 Inference Acceleration Technique of DLLMs

The acceleration of inference for DLLMs is an area of active research, as reducing latency and computational overhead is crucial for real-world applications. Broadly, two primary domains have emerged for accelerating inference: dynamic caching mechanism and optimized sampling strategy.

Dynamic Caching Mechanism. The caching mechanism for diffusion models differs significantly from that of AR models (Jiang et al., 2025), primarily due to the bidirectional attention mechanism employed in diffusion processes. Several approaches have optimized caching in this context: dLLM-Cache (Liu et al., 2025b) introduces a dynamic key-value cache strategy that updates tokens with significant feature changes; Fast-dLLM (Wu et al., 2025) accelerates this by using chunked KV caching and confidence-aware parallel decoding; dKV-Cache (Ma et al., 2025) employs delayed caching to reuse key-value states after token decoding, ensuring stable acceleration; and Sparse-dLLM (Song et al., 2025) reduces overhead by pruning less important tokens in the cache.

Optimized Sampling Strategy. Beyond dynamic caching, optimizing the sampling process itself represents a pivotal direction for accelerating inference in DLLMs. Research has explored diverse strategies: APD (Israel et al., 2025) dynamically adjusts the number of tokens sampled in parallel with a small AR verification model; WINO (Hong et al., 2025) introduces revocable decoding to enhance both efficiency and effectiveness in DLLMs; similarly, FreeDave (Wu & Zhang, 2025) enables lossless parallel decoding via self-drafting and self-verification; COVER (Xiang et al., 2026) performs context-preserving verification via KV-cache override to reduce flip-flop revisions in revocable diffusion decoding. and Learn2PD (Bao et al., 2025) trains a lightweight filter model to predict token-wise convergence, reducing redundant decoding.

3 Preliminary

Decoding Process of DLLMs. Let $\mathcal{V}$ be the vocabulary, in which [MASK] $\in \mathcal{V}$ is a special token. Given a prompt X, a DLLM parameterized with θ is designed to generate a response $Y = [y_1, y_2, \ldots, y_L]$ with a predefined response length L. The response sequence is initialized as all special mask tokens, $Y \leftarrow [\text{[MASK]}, \text{[MASK]}, \ldots, \text{[MASK]}]$. The decoding process iteratively refines the response sequence Y in each decoding step. We model the decoding process as a state transition, where each masked token (state $\mathbf{M}$) is transformed into a specific semantic token (state $\mathbf{T}$) under certain rules.

Specifically, at each decoding step, the model estimates the probability distribution over the response sequence as $p_\theta(\hat{Y}|X, Y)$. Tokens that meet the $\mathbf{M} \to \mathbf{T}$ transition condition are unmasked.

$$y_i \leftarrow \arg\max_{v \in \mathcal{V}} p_\theta(\hat{y_i} = v|X, Y), \text{ if } i \in \text{rule}_{\mathbf{M}\to\mathbf{T}}.$$

Here, $\text{rule}_{\mathbf{M}\to\mathbf{T}}$ denotes the set of positions that satisfy the $\mathbf{M} \to \mathbf{T}$ transition condition. For example, under the commonly used high-confidence greedy decoding, we have

$$\text{rule}_{\mathbf{M}\to\mathbf{T}} = \{\underset{i \in \{i|y_i = \text{[MASK]}\}}{\arg\max} (\max_{v \in \mathcal{V}} p_\theta(\hat{y_i} = v|X, Y)\}.$$

Decoding concludes once every element in Y has transitioned to state $\mathbf{T}$ as the final generated result.

4 Method

4.1 Motivation

Current DLLMs still face the speed-performance trade-off, with optimal performance achieved when decoding one token at a time. Parallel decoding, while faster, often results in performance drop due to “combinatorial contradiction," where sampling multiple tokens from different plausible completion paths leads to contradictions. We attribute this to the strictly *discrete* nature of parallel decoding, where simultaneous sampling of multiple positions lacks coordination, causing semantic conflicts. To address this, we propose introducing a continuous intermediate state between the two discrete states, state $\mathbf{M}$ and state $\mathbf{T}$, allowing positions to accumulate and align decoding intent before finalizing the output, thereby resolving inconsistencies and improving performance (see Fig. 2).

4.2 Rejection Mixing of Continuous States

Building on the motivation above, we propose Rejection Mixing (ReMix), a method designed to accelerate semantic propagation in DLLM inference by introducing a continuous intermediate state $\mathbf{C}$ into the standard discrete $\mathbf{M} \to \mathbf{T}$ transition. This state allows for mutual perception and iterative

adjustment of inter-positional inconsistencies before collapsing into a final discrete state T. Operationally, each position starts as a masked token in state $\mathbf{M}$. At each decoding step, if the decoding rule is satisfied, the position is directly decoded to the token state $\mathbf{T}$; otherwise, it shifts from $\mathbf{M}$ to the continuous state $\mathbf{C}$, where it is iteratively updated by the mixing rule, and either be accepted to $\mathbf{T}$ once the decoding rule holds or rejected back from $\mathbf{C}$ to $\mathbf{M}$ according to the rejection rule (detailed in Algorithm 1 in the Appendix).

We now detail the specific rules governing the transitions of each position among states $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{T}$ during the decoding process.

Initialization ($\mathbf{M}$). As introduced in Section 3, the decoding process begins with the entire response sequence initialized with mask tokens $\mathbf{M}$ at every position.

$$y_i \leftarrow \text{[MASK]},\ \forall i \in \{1, \ldots, L\}. \tag{Initialization}$$

Decoding Rule ($(\mathbf{M}, \mathbf{C}) \rightarrow \mathbf{T}$). For adaptive parallel decoding, we employ a commonly used confidence-based thresholding strategy. Specifically, when the confidence of the output distribution at a given position surpasses a predefined threshold τ_{conf}, that position is decoded into a concrete semantic token, corresponding to a transition from state $\mathbf{M}$ (or $\mathbf{C}$) to state $\mathbf{T}$.

$$y_i \leftarrow \arg\max_{v \in \mathcal{V}} p_\theta(\hat{y}_i = v | X, Y),\ \text{if } \max_{v \in \mathcal{V}} p_\theta(\hat{y}_i = v | X, Y) > \tau_{\text{conf}}. \tag{$(\mathbf{M}, \mathbf{C}) \rightarrow \mathbf{T}$}$$

Mixing Rule ($\mathbf{M} \rightarrow \mathbf{C}$ ↻). The Mixing Rule defines the transition from state $\mathbf{M}$ to the continuous state $\mathbf{C}$, as well as the self-transition process within $\mathbf{C}$. Since $\mathbf{C}$ is a continuous state rather than a discrete token, the transition to $\mathbf{C}$ is represented as an update to the embedding y^{emb}. For discrete tokens, let the embedding layer be parameterized by a weight matrix $W \in \mathbb{R}^{|V| \times d}$, where $|V|$ is the vocabulary size and d is the embedding dimension. Because the mapping between a token and its embedding is one-to-one, updating the token y can be equivalently viewed as updating its embedding y^{emb}. Our Mixing Rule is designed to preserve the output distribution information from the previous step to enhance consistency in subsequent decoding.

$$y_i^{\text{emb}} \leftarrow \beta W^\top p_\theta(\hat{y}_i | X, Y) + (1 - \beta)\text{Emb}_{\text{[MASK]}},\ \text{if position } i \text{ not decoded}. \tag{$\mathbf{M} \rightarrow \mathbf{C}$ ↻}$$

Here, $\text{Emb}_{\text{[MASK]}}$ denotes the embedding of the [MASK] token, and $\beta \in (0, 1)$ represents the mixing coefficient between the output distribution and $\text{Emb}_{\text{[MASK]}}$. In addition, when converting the output distribution into an embedding update (*i.e.*, computing $W^\top p_\theta(\hat{y}_i | X, Y)$), we apply the commonly used top-p (nucleus) sampling for stability and efficiency, as detailed in Section A.2.

Rejection Rule ($\mathbf{C} \rightarrow \mathbf{M}$). As a training-free method, although Eq. ($\mathbf{M} \rightarrow \mathbf{C}$ ↻) mixes output distribution information for the [MASK] tokens at decoding positions, it may introduce inference-training inconsistencies and distributional shifts, since all training is conducted on discrete tokens. To mitigate this issue, we introduce a rejection rule: when a decoding position exhibits significant instability in its output, it is reset to the initial state M. Specifically, when the output distributions of two consecutive decoding steps are too far apart, the position is reset to [MASK].

$$y_i \leftarrow \text{[MASK]},\ \text{if position } i \text{ not decoded, and } \mathbb{D}_{\text{JS}}(p_\theta(\hat{y}_i | X, Y) \parallel p_\theta(\hat{y}_i | X, Y_{\text{old}})) > \tau_{\text{rej}}, \tag{$\mathbf{C} \rightarrow \mathbf{M}$}$$

where $\mathbb{D}_{\text{JS}}(\cdot \parallel \cdot)$ denotes the Jensen-Shannon (JS) divergence, and τ_{rej} is the hyperparameter threshold used to determine instability.

In summary, ReMix orchestrates a dynamic decoding process where positions flexibly transition between masked, continuous, and discrete states. By enabling continuous-space refinement ($\mathbf{M} \rightarrow \mathbf{C}$ ↻) and providing a stability fallback mechanism ($\mathbf{C} \rightarrow \mathbf{M}$), ReMix effectively addresses combinatorial contradictions and accelerates semantic propagation without additional training.

4.3 Discussion

The proposed ReMix method offers a novel perspective on stabilizing parallel decoding in DLLMs. By leveraging continuous representations, we bypass the rigid combinatorial constraints of purely discrete sampling. The Mixing Rule serves as a form of "soft" lookahead, allowing undecoded positions to communicate their tentative semantic intent to neighbors without fully committing to a potentially erroneous discrete token. This facilitates a more globally coherent decoding trajectory.

Furthermore, the Rejection Rule acts as a critical regularizer. It implicitly acknowledges the limitations of using continuous states in a model trained for discrete inputs. By resetting unstable positions

Table 1: Performance and inference speedup comparison on diverse language benchmarks.

Benchmark	Method	Accuracy	Steps	Step Reduction	Latency	Latency Speedup
GSM8K	LLaDA	73.01	256	1.00 ×	13.89	1.00 ×
Math Reasoning	ReMix	75.66 (+2.65)	51.55 (-204.45)	4.97 ×	3.00 (-10.89)	4.63 ×
MATH-500	LLaDA	32.20	256	1.00 ×	13.96	1.00 ×
Math Reasoning	ReMix	35.00 (+2.80)	66.33 (-189.67)	3.86 ×	3.93 (-10.03)	3.55 ×
HumanEval	LLaDA	43.90	256	1.00 ×	19.22	1.00 ×
Code Generation	ReMix	44.50 (+0.60)	89.40 (-166.60)	2.86 ×	7.13 (-12.09)	2.70 ×
MBPP	LLaDA	36.20	256	1.00 ×	14.11	1.00 ×
Code Generation	ReMix	37.00 (+0.80)	84.28 (-171.72)	3.04 ×	5.05 (-9.06)	2.79 ×
Countdown	LLaDA	25.78	256	1.00 ×	14.39	1.00 ×
Logical Reasoning	ReMix	29.68 (+3.90)	101.44 (-154.56)	2.52 ×	6.08 (-8.31)	2.37 ×
Sudoku	LLaDA	14.25	256	1.00 ×	19.34	1.00 ×
Logical Reasoning	ReMix	18.55 (+4.30)	84.06 (-171.94)	3.05 ×	6.76 (-12.58)	2.86 ×
ARC-E	LLaDA	59.68	256	1.00 ×	13.70	1.00 ×
Commonsense Reasoning	ReMix	70.54 (+10.86)	50.69 (-205.31)	5.05 ×	2.98 (-10.72)	4.60 ×
ARC-C	LLaDA	52.17	256	1.00 ×	13.67	1.00 ×
Commonsense Reasoning	ReMix	66.22 (+14.05)	61.30 (-194.70)	4.18 ×	3.49 (-10.18)	3.92 ×

Table 2: Performance and inference speedup comparison across diverse multimodal understanding and reasoning benchmarks. We use CIDEr for Flickr30k and accuracy for other benchmarks.

Benchmark	Method	Performance	Steps	Step Reduction	Latency	Latency Speedup
Flickr30k-lite	MMaDA	57.52	256	1.00 ×	108.02	1.00 ×
Captioning	ReMix	59.59 (+2.07)	30.13 (-225.87)	8.50 ×	14.37 (-93.65)	7.52 ×
AI2D-lite	MMaDA	62.40	256	1.00 ×	109.27	1.00 ×
Chart Understanding	ReMix	62.60 (+0.20)	37.82 (-218.18)	6.77 ×	19.24 (-90.03)	5.68 ×
MathVision-mini	MMaDA	12.83	256	1.00 ×	111.12	1.00 ×
Math Reasoning	ReMix	12.83 (+0.00)	52.71 (-203.29)	4.86 ×	27.56 (-83.56)	4.03 ×
MathVista-mini	MMaDA	31.60	256	1.00 ×	110.38	1.00 ×
Math Reasoning	ReMix	34.60 (+3.00)	43.61(-212.39)	5.87 ×	20.73(-89.65)	5.32 ×
MMMU-val	MMaDA	23.33	256	1.00 ×	113.76	1.00 ×
Multi-discipline Reasoning	ReMix	24.67 (+1.34)	58.04 (-197.96)	4.41 ×	30.36 (-83.40)	3.75 ×
ScienceQA-IMG	MMaDA	46.70	256	1.00 ×	111.60	1.00 ×
Multi-discipline Reasoning	ReMix	47.35 (+0.65)	43.86 (-212.14)	5.84 ×	21.35 (-69.27)	5.23 ×

to [MASK], we effectively prune divergent decoding paths early, preventing error cascading. This training-free nature is a significant advantage, as it avoids the substantial computational cost of re-training or fine-tuning large-scale diffusion models while still improving the decoding consistency.

5 Experiments

5.1 Main Results

Performance gain and Speedup in the Language Domain. We evaluate the performance and efficiency of LLaDA w/ and w/o ReMix across eight language benchmarks, as shown in Table 1. Across eight diverse reasoning and coding benchmarks, incorporating ReMix consistently improves both accuracy and efficiency over LLaDA, demonstrating that our method delivers generalizable gains rather than task-specific spikes. On the accuracy side, ReMix yields improvements on every benchmark, ranging from modest boosts in code generation tasks (e.g., +0.60 on HumanEval, +0.80 on MBPP) to substantial jumps on challenging reasoning datasets (e.g., +3.90 on Countdown, +4.30 on Sudoku, and up to +14.05 on ARC-C). Notably, even in math reasoning tasks where LLaDA is relatively strong, ReMix still achieves significant improvements. Efficiency gains are equally consistent. ReMix reduces the number of sampling steps by 2.5×–5.0×, corresponding to absolute reductions of 150–205 steps compared to the 256-step LLaDA baseline. This directly translates into substantially lower generation latency: latency drops by 8–13 seconds, producing 2.4×–4.6× end-to-end speedup depending on the dataset. Importantly, these efficiency improvements do not come at the cost of accuracy. In all cases, accuracy increases alongside faster inference.

Performance Gain and Speedup in the Multimodal Domain. We further evaluate ReMix in the multimodal setting by integrating it into the MMADA framework and conducting experiments on six vision–language benchmarks, as reported in Table 2. ReMix again shows consistent improvements over MMaDA, confirming that the benefits of our approach extend beyond text-only reasoning into the multimodal domain. On performance, Remix delivers accuracy gains on nearly all datasets, with improvements ranging from +0.20 on AI2D-lite to +3.00 on MathVista-mini, and notably +2.07

Table 3: Experiment results on different generation lengths and block lengths

Benchmark	Gen Length	Block Length	Method	Accuracy	Steps	Step Reduction	Latency	Latency Speedup
Different Generation Lengths								
GSM8K	128	128	LLaDA	59.21	128	1.00 ×	15.15	1.00 ×
			ReMix	59.74 (+0.53)	36.63	3.49 ×	4.63	3.27 ×
	256		LLaDA	72.48	256	1.00 ×	38.56	1.00 ×
			ReMix	75.28 (+2.80)	58.54	4.37 ×	8.63	4.47 ×
	512		LLaDA	74.68	512	1.00 ×	114.13	1.00 ×
			ReMix	77.63 (+2.95)	74.76	6.85 ×	17.66	6.46 ×
MathVista -mini	128	128	MMaDA	31.80	128	1.00 ×	44.60	1.00 ×
			ReMix	33.80 (+2.00)	35.87	3.57 ×	16.15	2.76 ×
	256		MMaDA	31.60	256	1.00 ×	110.38	1.00 ×
			ReMix	34.60 (+3.00)	43.61	5.87 ×	20.73	5.32 ×
	512		MMaDA	31.30	512	1.00 ×	242.9	1.00 ×
			ReMix	33.40 (+2.10)	71.31	7.18 ×	38.81	6.26 ×
Different Block Lengths								
GSM8K	256	32	LLaDA	81.73	256	1.00 ×	35.32	1.00 ×
			ReMix	81.88 (+0.15)	66.43	3.85 ×	9.66	3.66 ×
		64	LLaDA	82.56	256	1.00 ×	38.77	1.00 ×
			ReMix	82.48 (-0.08)	63.04	4.06 ×	9.49	4.09 ×
		128	LLaDA	72.48	256	1.00 ×	38.56	1.00 ×
			ReMix	75.28 (+2.80)	58.54	4.37 ×	8.63	4.47 ×
MathVista -mini	256	32	MMaDA	33.00	256	1.00 ×	109.39	1.00 ×
			ReMix	34.40 (+1.40)	55.08	4.65 ×	25.11	4.36 ×
		64	MMaDA	31.70	256	1.00 ×	109.84	1.00 ×
			ReMix	33.90 (+2.20)	54.49	4.70 ×	25.69	4.28 ×
		128	MMaDA	31.60	256	1.00 ×	110.38	1.00 ×
			ReMix	34.60 (+3.00)	43.61	5.87 ×	20.73	5.32 ×

on Flickr30k-lite. Efficiency improvements are substantial across all benchmarks. Remix reduces sampling steps by 4.4× to 8.5×. As a result, inference becomes 3.75× to 7.52× faster, saving 70–94 seconds per query depending on the task. In particular, captioning and chart-understanding benchmarks achieve the largest speedups (7.5× and 5.7×).

5.2 Ablation Study and Further Analysis

Performance Across Different Generation and Block Lengths. In the first half of Table 3, we evaluate the performance of ReMix across various generation lengths, keeping the block length fixed to 128. ReMix consistently shows a significant imp rovement in accuracy for all generation lengths tested, while also substantially reducing the number of decoding steps. Remarkably, the performance gains are more pronounced with longer generation lengths. For instance, when the generation length is set to 512, ReMix achieves accuracy improvements of 2.95% and 2.10% on GSM8K and MathVista-mini, while accelerating decoding efficiency by 6.46× and 6.26×, respectively. Similarly, we present the results of ReMix with different block lengths in the lower half of Table 3, where the generation length is fixed to 256. In this case, ReMix consistently delivers comparable or better accuracy across all three block length settings, accompanied by significant reductions in inference time. In summary, ReMix proves robustness w.r.t. both generation length and block length. It maintains high accuracy while consistently achieving stable acceleration in model inference, regardless of variations in these parameters.

Analysis on the Mitigation of Combinatorial Contradictions. To demonstrate the effectiveness of ReMix in mitigating "combinatorial contradiction", we use GPT-4o-mini to evaluate the generated responses on GSM8K, scoring them from 1 to 5, with lower scores indicating poorer performance. As shown in Fig. 3, our method achieves a significantly higher score of 4.32 compared to 3.23 from the baseline, with fewer decoding steps, and approaches the score of 4.33 achieved by standard decoding. The distribution further highlights the improvement: in the baseline, samples are mainly distributed between scores of 2 and 4, with fewer than 200 samples rated 5. In contrast, ReMix results in a distribution skewed towards scores of 4 and 5, with over 600 samples receiving a perfect score of 5. The significant improvement, as shown by both the higher average score and the better distribution of results, demonstrates the effectiveness of ReMix in alleviating the degradation caused by the combinatorial contradiction during parallel decoding.

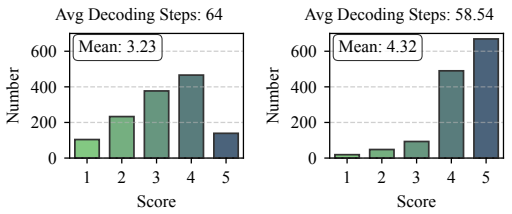


Figure 3: Comparison of GSM8K responses (LLaDA-based) score distributions addressing combinatorial contradictions on naive parallel decoding (left) and ReMix (right)

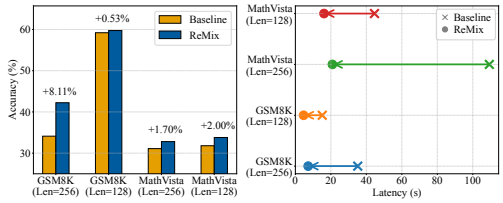


Figure 4: Experiment results on fully diffusion-based decoding (gen length = block length).

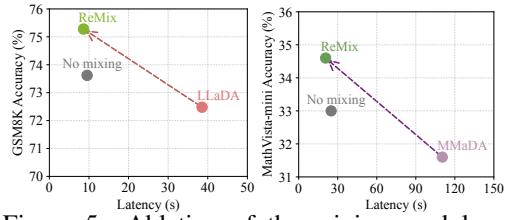
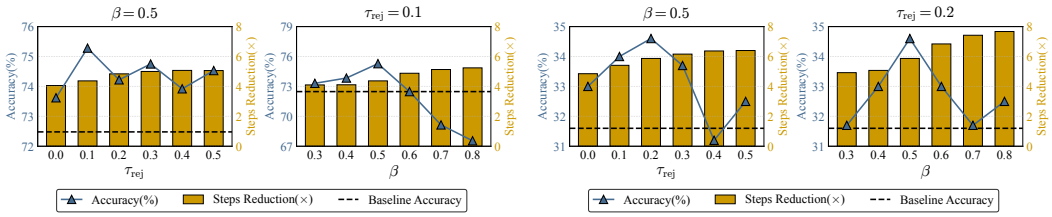


Figure 5: Ablation of the mixing module on GSM8K (left) and MathVista-mini (right).



(a) ReMix (LLaDA-based) on GSM8K

(b) ReMix (MMaDA-based) on MathVista-mini

Figure 6: Ablation study on the mixing ratio β and rejection threshold τ_{rej}

Performance on Fully Diffusion-Based Decoding In Fig. 4, we examine the full-diffusion scenario, where the generation length matches the block length. The results demonstrate that ReMix consistently achieves both accuracy improvements and speedup on both baselines. It's worth mentioning that LLaDA baseline(Len=256) experiences severe performance degradation compared to the semi-autoregressive decoding paradigm shown in Table 1, whereas ReMix compensates for this loss with far fewer decoding steps. This highlights that our method effectively enhances baseline full-diffusion decoding by incorporating continuous representations.

Ablation on Mixing Module. We perform an ablation study on the mixing module of our method by setting the mixing ratio to 0, thereby disabling the intermediate continuous state $\mathbf{C}$. This results in a degeneration of our method to confidence-aware parallel decoding strategy (Wu et al., 2025). As illustrated in Fig. 5, our approach outperforms this variant, achieving significantly higher accuracy while maintaining decoding speedup. These findings provide compelling evidence that the incorporation of continuous embeddings enhances the model generation capability.

Tuning of Mixing Ratio and Rejection Threshold. In Fig. 6, we present the results of ReMix with varying mixing ratio β and rejection threshold τ_{rej}, two key parameters that govern the speed-quality trade-off of the method. For both tasks, accuracy initially increases as β grows. A moderate level of mixing strengthens the contribution of the continuous embedding, enabling the model to capture richer contextual signals and improving quality. However, when β becomes too large (e.g., $\beta > 0.6$ in GSM8K or $\beta > 0.5$ in MathVista), performance begins to degrade sharply, which reflects a shift toward over-reliance on continuous updates, destabilizing the hybrid decoding process. A similar pattern emerges with τ_{rej}. Increasing the threshold initially boosts accuracy by allowing the model to accept more continuous updates when they are confident, improving the refinement of intermediate predictions. Beyond a certain point, however, overly permissive thresholds introduce low-confidence or noisy continuous embeddings, which in turn reduce accuracy. Across all tested values, step reduction remains large and stable, and ReMix surpasses the baseline in nearly all configurations, demonstrating resilience to a wide range of τ_{rej} values.

6 Conclusion

This work identifies combinatorial contradiction as a fundamental obstacle limiting the quality–speed trade-off in parallel DLLM decoding, which arises due to the rigidity of purely discrete updates that fail to preserve cross-position dependencies when multiple tokens are decoded simultaneously. We thus introduce ReMix, a simple, training-free decoding framework that augments masked discrete diffusion with a continuous mixing state and a lightweight rejection mechanism. ReMix consistently delivers 2–8× inference acceleration without sacrificing output quality and often improves it, narrowing the performance gap with AR models while retaining DLLMs' advantages. These results highlight the promise of integrating continuous and discrete diffusion and point toward a new family of efficient generative models.

7 ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China (No. 62306178) and STCSM (No. 22DZ2229005).

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *NeurIPS*, pp. 17981–17993, 2021a.

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

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

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *CoRR*, abs/2505.24857, 2025.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *NeurIPS*, 2022.

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

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms. *CoRR*, abs/2509.26488, 2025.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

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

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *NeurIPS*, pp. 8780–8794, 2021.

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. *CoRR*, abs/2506.20639, 2025.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *NeurIPS*, 2020.

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

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *CoRR*, abs/2506.00413, 2025.

Haoyun Jiang, Haolin Li, Jianwei Zhang, Fei Huang, Qiang Hu, Minmin Sun, Shuai Xiao, Yong Li, Junyang Lin, and Jiangchao Yao. Catekv: On sequential consistency for long-context LLM inference acceleration. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth A dozen images. *CoRR*, abs/1603.07396, 2016.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. *CoRR*, abs/2505.16839, 2025a.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. *CoRR*, abs/2508.10875, 2025b.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. In *ICLR*. OpenReview.net, 2025a.

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. *CoRR*, abs/2506.06295, 2025b.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*. OpenReview.net, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *CoRR*, abs/2505.15781, 2025.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16784–16804. PMLR, 2022.

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. *CoRR*, abs/2502.09992, 2025.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, July 2024.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *ICLR*. OpenReview.net, 2025.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models. *CoRR*, abs/2510.22510, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10674–10685. IEEE, 2022.

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *NeurIPS*, 2024.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei (eds.), *ICML*, volume 37 of *JMLR*, pp. 2256–2265. JMLR.org, 2015.

Jiaming Song and Linqi Zhou. Ideas in inference-time scaling can benefit generative pre-training algorithms. *CoRR*, abs/2503.07154, 2025.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*. OpenReview.net, 2021.

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. *CoRR*, abs/2508.02558, 2025.

Yuchuan Tian, Yuchen Liang, Jiacheng Sun, Shuo Zhang, Guangwen Yang, Yingte Shu, Sibo Fang, Tianyu Guo, Kai Han, Chao Xu, Hanting Chen, Xinghao Chen, and Yunhe Wang. From next-token to next-block: A principled adaptation path for diffusion llms. *CoRR*, abs/2512.06776, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, pp. 4566–4575. IEEE Computer Society, 2015.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. In *NeurIPS*, 2024.

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

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *CoRR*, abs/2505.22618, 2025.

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

Yanzheng Xiang, Lan Wei, Yizhen Yao, Qinglin Zhu, Hanqi Yan, Chen Jin, Philip Alexander Teare, Dandan Zhang, Lin Gui, Amrutha Saseendran, and Yulan He. Stop the flip-flop: Context-preserving verification for fast revocable diffusion decoding. *CoRR*, abs/2602.06161, 2026.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *CoRR*, abs/2505.15809, 2025.

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

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. *CoRR*, abs/2505.16933, 2025.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Trans. Assoc. Comput. Linguistics*, 2:67–78, 2014.

Xiang Yue, Yuansheng Ni, Tianyu Zheng, Kai Zhang, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, pp. 9556–9567. IEEE, 2024.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Reality check on the evaluation of large multimodal models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *NAACL*, pp. 881–916. Association for Computational Linguistics, 2025.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

Huangjie Zheng, Shansan Gong, Ruixiang Zhang, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously augmented discrete diffusion model for categorical generative modeling. *CoRR*, abs/2510.01329, 2025.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi S. Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner. *CoRR*, abs/2510.03206, 2025.

A Implementation Detail

A.1 Pseudocode

Algorithm 1 ReMix

Require: Prompt X, Model p_θ, Hyperparameters
Initialize $S \leftarrow [\mathbf{M}, \ldots, \mathbf{M}]$ (State tracker)
Initialize Y^{emb} with $\text{Emb}_{[\text{MASK}]}$
$P_{\text{old}} \leftarrow \mathbf{0}$
while $\exists i$ s.t. $S_i \neq \mathbf{T}$ **do**
  $P \leftarrow p_\theta(\hat{Y} | X, Y^{\text{emb}})$ ▷ Model forward pass
  for each position i s.t. $S_i \neq \mathbf{T}$ **do**
    if Condition (Eq. ($(\mathbf{M}, \mathbf{C}) \rightarrow \mathbf{T}$)) **then**
      $S_i \leftarrow \mathbf{T}$
      Update Y_i^{emb} (Eq. ($(\mathbf{M}, \mathbf{C}) \rightarrow \mathbf{T}$))
    else ▷ Not decoded
      if $S_i = \mathbf{C}$ **and** Condition (Eq. ($\mathbf{C} \rightarrow \mathbf{M}$)) **then**
        $S_i \leftarrow \mathbf{M}$ ▷ Apply $\mathbf{C} \rightarrow \mathbf{M}$
        $Y_i^{\text{emb}} \leftarrow \text{Emb}_{[\text{MASK}]}$
      else
        $S_i \leftarrow \mathbf{C}$ ▷ Apply $\mathbf{M} \rightarrow \mathbf{C}$ or $\mathbf{C}$ ⟳
        Update Y_i^{emb} (Eq. ($\mathbf{M} \rightarrow \mathbf{C}$ ⟳))
      end if
    end if
  end for
  $P_{\text{old}} \leftarrow P$ ▷ Store for next step's Eq. ($\mathbf{C} \rightarrow \mathbf{M}$)
end while
return Final Tokens from Y^{emb}

A.2 Adaptive Top-p Sampling in Mixing Rule

As mentioned in Section 4, we employ top-p (nucleus) sampling during the embedding update to improve inference stability. Rather than using a fixed nucleus threshold p across all positions, we adopt a dynamic strategy that adapts to the model's prediction confidence at each position. Specifically, let $p_{\max}$ denote the highest probability in the output distribution at the current position. We define the adaptive nucleus threshold as:

$$p_{\text{dyn}} = \min\left(2 \cdot p_{\max},\ 0.9\right).$$

The residual probability mass, i.e., the portion not included in this top set, is assigned to the [MASK] token. To enhance numerical and behavioral stability during iterative refinement, this design prevents low-confidence, low-probability tokens from contributing to the embedding update; instead, their weight is consolidated into the `[MASK]` embedding, which serves as a stable anchor in the continuous state $\mathbf{C}$.

B Experiment Setup

Datasets and Baselines. We evaluate ReMix across diverse tasks and domains. In the language domain, we compare ReMix with the standard LLaDA decoding on eight tasks: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b), Countdown (Zhao et al., 2025), Sudoku (Zhao et al., 2025), ARC-E (Clark et al., 2018), and ARC-C (Clark et al., 2018), encompassing four text-generation categories: mathematical reasoning, code generation, logical reasoning, and commonsense reasoning. For the vision-language domain, we compare ReMix with the standard MMaDA (Yang et al., 2025) decoding on six multimodal understanding tasks: Flickr30k (Young et al., 2014), AI2D (Kembhavi et al., 2016), MathVision (Wang et al., 2024), MathVista (Lu et al., 2024), MMMU (Yue et al., 2024) and ScienceQA (Lu et al., 2022), covering four types of multimodal tasks: captioning, chart understanding, math reasoning and multi-discipline reasoning. For clarity, we run experiments on the lite version of Flickr30k, AI2D from LMMs-Lab (Zhang et al., 2025), the official testmini subset of MathVision, MathVista, the validation set of MMMU and the ScienceQA-IMG version of ScienceQA.

Evaluation Details. All benchmarks are evaluated zero-shot, except Sudoku, which uses a 4-shot setting. We report CIDEr Vedantam et al. (2015) for Flickr30k and accuracy for all other benchmarks. For multimodal understanding tasks, we choose GPT-4o mini (OpenAI, 2024) as evaluator to assess the correctness and quality of generated answers. To quantify inference efficiency, we measure the required decoding steps and latency(in seconds) for ReMix and baseline on each task, averaging across all samples in the corresponding benchmark.

Implementation Details. Our evaluation pipeline is implemented on top of the WINO codebase (Hong et al., 2025), adapting its evaluation scripts to our setting while retaining the default preprocessing and protocol unless otherwise noted. We use LLaDA-8B-Instruct[2] for language evaluation and MMaDA-8B-MixCoT[3] for vision–language evaluation. All experiments are conducted on 8 NVIDIA GeForce RTX 3090 GPUs, except for the main language experiments, which are performed on 8 NVIDIA A100 GPUs. We employ semi-autoregressive sampling strategy that partitions the sequence into blocks and generates them left to right. In our experiments, we set the generation length to 256 and block length to 128. For the hyperparameters in ReMix, we set the confidence threshold τ_{conf} to 0.8. The mixing ratio β is selected from $\{0.4, 0.5, 0.6\}$ to balance the trade-off between speed and quality, while the rejection threshold τ_{rej} varies between 0.1 and 0.4.

C Additional Experiments

C.1 Comparison with State-of-the-Art Methods

We compare ReMix with three SOTA accelerating methods on GSM8K: WINO, Fast-dLLM, and Learn2PD. WINO and ReMix are evaluated under default settings, while Fast-dLLM and Learn2PD follow their optimal configuration with block length 32 (Learn2PD uses the official adapter[4]). As shown in Table 4, ReMix achieves the best speed-quality trade-off, outperforming baselines in throughput without sacrificing accuracy.

Table 4: Accuracy and throughput (TPS) comparison on GSM8K.

Method	Block Length = 128		Block Length = 32	
	Acc.	TPS	Acc.	TPS
LLaDA	72.55	22.15	81.88	22.20
WINO	75.28(+2.73)	74.82(×3.38)	–	–
Fast-dLLM	–	–	80.29(-1.59)	75.45(×3.40)
Learn2PD	–	–	78.17(-3.71)	102.72(×4.63)
ReMix	75.36(+2.81)	102.47(×4.63)	81.20(-0.68)	80.29(×3.62)

C.2 Analysis of Computational Overhead

We profile the average overhead on GSM8K in Table 5, where ReMix-specific computations account for only 9.12% of the overall runtime. This efficiency stems from the lightweight design of both modules: the Mixing update contains highly parallelizable projection restricted to undecoded positions, while the Rejection rule involves only efficient distribution-wise divergence calculations. Since neither operation requires extra backbone forward passes, the overhead remains negligible compared to the dominant transformer computation.

Table 5: Runtime breakdown of ReMix overhead on GSM8K.

Component	Time (s)	Share (%)
End-to-end generation	2.52	100.00
Mixing update	0.14	5.56
Rejection	0.09	3.56
ReMix-specific total	0.23	9.12

[2] https://huggingface.co/GSAI-ML/LLaDA-8B-Instruct
[3] https://huggingface.co/Gen-Verse/MMaDA-8B-MixCoT
[4] https://github.com/ims-kdks/Learning-to-Parallel-Decoding

D CASE STUDY

To facilitate a comprehensive comparison of the decoding processes between the baseline methods and ReMix, we present two generation examples: one from GSM8K and another from MMMU-val. As demonstrated, the standard decoding procedure can prematurely predict incorrect tokens, which significantly disrupts the generation of subsequent tokens. In contrast, ReMix effectively mitigates this error propagation by incorporating continuous information into the decoding process, thereby preventing such errors from occurring in the first place.

Case: GSM8K, Generation Length=256, Block Length=128

Question: Randy has 9 oatmeal cookies, 4 chocolate chip cookies, and 5 sugar cookies. He ate 3 cookies for an early day snack, one of each flavor. He ate 2 oatmeal cookies for lunch. He gives 2 sugar cookies to his friends. Then, he bakes 4 of each flavor for dinner. How many cookies does he have now?
GT: 23

LLaDA:

Step: 239 (Decoding 2nd block, error occurs):
First, let's determine the initial number of each type of cookie:
- Oatmeal cookies: 9
- Chocolate chip cookies: 4
- Sugar cookies: 5
Next, we account for the cookies Randy ate and gave away:
- He ate 3 cookies for an early day snack, one of each flavor:
- Oatmeal cookies: 9 - 1 = 8
- Chocolate chip cookies: 4 - 1 = 3
- Sugar cookies: 5 - 1 = 4
- He ate 2 oatmeal cookies for lunch:
- Oatmeal cookies: 8 - 2 = 6
- He gave 2 sugar cookies to his friends:
- Sugar cookies: 4 - 2 = 2
, he bakes 4 of each flavor for dinner:
- Oatmeal cookies: 6 + 4 = 10
- Chocolate chip cookies: 3
- Sugar cookies: 2 + 4 = 6
</reasoning>
<answer>
$\boxed{1}$

Step: 239 (Decoding 2nd block, error accumulates):
First, let's determine the initial number of each type of cookie:
- Oatmeal cookies: 9
- Chocolate chip cookies: 4
- Sugar cookies: 5
Next, we account for the cookies Randy ate and gave away:
- He ate 3 cookies for an early day snack, one of each flavor:
- Oatmeal cookies: 9 - 1 = 8
- Chocolate chip cookies: 4 - 1 = 3
- Sugar cookies: 5 - 1 = 4
- He ate 2 oatmeal cookies for lunch:
- Oatmeal cookies: 8 - 2 = 6
- He gave 2 sugar cookies to his friends:
- Sugar cookies: 4 - 2 = 2
, he bakes 4 of each flavor for dinner:
- Oatmeal cookies: 6 + 4 = 10
- Chocolate chip cookies: 3

- Sugar cookies: 2 + 4 = 6
</reasoning>
<answer>
$\boxed{19}$

Step: 255 (Final result):
First, let's determine the initial number of each type of cookie:
- Oatmeal cookies: 9
- Chocolate chip cookies: 4
- Sugar cookies: 5
Next, we account for the cookies Randy ate and gave away:
- He ate 3 cookies for an early day snack, one of each flavor:
- Oatmeal cookies: 9 - 1 = 8
- Chocolate chip cookies: 4 - 1 = 3
- Sugar cookies: 5 - 1 = 4
- He ate 2 oatmeal cookies for lunch:
- Oatmeal cookies: 8 - 2 = 6
- He gave 2 sugar cookies to his friends:
- Sugar cookies: 4 - 2 = 2
Finally, he bakes 4 of each flavor for dinner:
- Oatmeal cookies: 6 + 4 = 10
- Chocolate chip cookies: 3
- Sugar cookies: 2 + 4 = 6
Total cookies: 10 + 3 + 6 = 1
</reasoning>
<answer>
$\boxed{19}$

ReMix:

Step: 47 (Decoding 2nd block, no error):
First, let's determine the initial number of each type of cookie:
- Oatmeal cookies: 9
- Chocolate chip cookies: 4
- Sugar cookies: 5
Next, we account for the cookies Randy ate:
- He ate 3 cookies for an early day snack, one of each flavor:
Oatmeal cookies: 9 - 1 = 8
Chocolate chip cookies: 4 - 1 = 3
Sugar cookies: 5 - 1 = 4
- He ate 2 oatmeal cookies for lunch:
Oatmeal cookies: 8 - 2 = 6
- He 2 sugar cookies to his friends:
Sugar cookies: 4 - 2 = 2
, he bakes 4 of each flavor for dinner:
- Oatmeal cookies: 6 + 4 = 10
- Chocolate chip cookies: 3 cookies
</reasoning>
<answer>
$\boxed{2}$

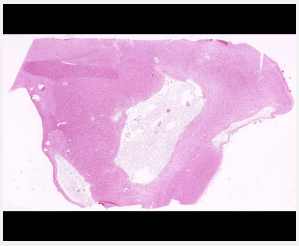
Step: 48 (Decoding 2nd block, correct induction):
First, let's determine the initial number of each type of cookie:
- Oatmeal cookies: 9
- Chocolate chip cookies: 4
- Sugar cookies: 5
Next, we account for the cookies Randy ate:

- He ate 3 cookies for an early day snack, one of each flavor:
Oatmeal cookies: 9 - 1 = 8
Chocolate chip cookies: 4 - 1 = 3
Sugar cookies: 5 - 1 = 4
- He ate 2 oatmeal cookies for lunch:
Oatmeal cookies: 8 - 2 = 6
- He 2 sugar cookies to his friends:
Sugar cookies: 4 - 2 = 2
, he bakes 4 of each flavor for dinner:
- Oatmeal cookies: 6 + 4 = 10
- Chocolate chip cookies: 3 + 4 = 7
- Sugar cookies: 2 + 4 =
</reasoning>
<answer>
$\boxed{2}$

Step: 64 (Final result):
First, let's determine the initial number of each type of cookie:
- Oatmeal cookies: 9
- Chocolate chip cookies: 4
- Sugar cookies: 5
Next, we account for the cookies Randy ate:
- He ate 3 cookies for an early day snack, one of each flavor:
Oatmeal cookies: 9 - 1 = 8
Chocolate chip cookies: 4 - 1 = 3
Sugar cookies: 5 - 1 = 4
- He ate 2 oatmeal cookies for lunch:
Oatmeal cookies: 8 - 2 = 6
- He gave 2 sugar cookies to his friends:
Sugar cookies: 4 - 2 = 2
Then, he bakes 4 of each flavor for dinner:
- Oatmeal cookies: 6 + 4 = 10
- Chocolate chip cookies: 3 + 4 = 7
- Sugar cookies: 2 + 4 = 6
Adding these together:
Total cookies = 10 + 7 + 6 = 23 cookies
</reasoning>
<answer>
$\boxed{23}$

Case: MMMU-val, Generation Length=256, Block Length=128

Question: Which statement is false?
(A) The differential diagnosis on this case is remote infarct vs. remote contusion
(B) There is a well developed gliotic wall associated with this remote injury
(C) The presence of macrophages is not helpful in distinguishing this from a subacute process
(D) This is very unlikely to have resulted from venous occlusion
Image:



GT: A

MMaDA:

- -

Step: 119 (Decoding 1st block, error occurs):
<think>
To determine which statement is false, letś analyze each option:
1. "The differential diagnosis on this case is remote infarct vs. remote contusion."
- This is because the image clearly shows a distant infarct a.
2. "There is a well developed gliotic wall associated with this remote injury."
- This is because the image shows a well gliotic wall, which is in the.
3. "The presence of macrophages is not helpful in distinguishing this from a subacute process."
- This is false

- -

Step: 121 (Decoding 1st block, error accumulates):
<think>
To determine which statement is false, letś analyze each option:
1. "The differential diagnosis on this case is remote infarct vs. remote contusion."
- This is true because the image clearly shows a distant infarct a.
2. "There is a well developed gliotic wall associated with this remote injury."
- This is true because the image shows a well gliotic wall, which is in the.
3. "The presence of macrophages is not helpful in distinguishing this from a subacute process."
- This is false

- -

Step: 256 (Final result):
<think>
To determine which statement is false, letś analyze each option:
1. “The differential diagnosis on this case is remote infarct vs. remote contusion."
- This is true because the image clearly shows a distant infarct from a distant location.
2. “There is a well developed gliotic wall associated with this remote injury."
- This is true because the image shows a well developed gliotic wall, which is visible in the foreground.
3. “The presence of macrophages is not helpful in distinguishing this from a subacute process."
- This is false because the image shows a subacute process with macrophages visible.
Therefore, the false statement is C.
</think>
C

ReMix:

- -

Step: 12 (Decoding 1st block, no error):
To determine which statement is false, letś analyze each option:
1. "The differential diagnosis on this case is vs. remote contusion."
This is false because the image clearly shows a.
2. "There is a well developed gliotic wall associated with this remote injury."

This is because a well gliotic wall is a feature in the image.
3. "The presence of macrophages is not helpful in distinguishing this from a subacute process."
This is'

Step: 47 (Final result):
<think>
To determine which statement is false, letś analyze each option:
1. “The differential diagnosis on this case is remote injury vs. is remote contusion."
This is false because the image clearly shows a distant regional contusion, not a distant infarct.
2. “There is a well developed gliotic wall associated with this remote injury."
This is true because a well developed gliotic wall is a clear feature in the image.
3. “The presence of macrophages is not helpful in distinguishing this from a subacute process."
This is true because the presence of macrophages in subacute processes is a characteristic feature of subacute processes.
4. “This is very unlikely to have resulted from venous occlusion."
This is true because the image clearly shows a distant infarct, not venous occlusion.
Therefore, the false statement is option A.
</think>
A