dVOTING: FAST VOTING FOR dLLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion Large Language Models (dLLMs) represent a new paradigm beyond autoregressive modeling, offering competitive performance while naturally enabling a flexible decoding process. Specifically, dLLMs can generate tokens at arbitrary positions in parallel, endowing them with significant potential for parallel test-time scaling, which was previously constrained by severe inefficiency in autoregressive modeling. In this work, we introduce dVOTING, a fast voting technique that boosts reasoning capability without training, with only an acceptable extra computational overhead. dVOTING is motivated by the observation that, across multiple samples for the same prompt, token predictions remain largely consistent, whereas performance is determined by a small subset of tokens exhibiting cross-sample variability. Leveraging the arbitrary-position generation capability of dLLMs, dVOTING performs iterative refinement by sampling, identifying uncertain tokens via consistency analysis, regenerating them through voting, and repeating this process until convergence. Extensive evaluations demonstrate that dVOTING consistently improves performance across various benchmarks. It achieves gains of 6.22%-7.66% on GSM8K, 4.40%-7.20% on MATH500, 3.16%-14.84% on ARC-C, and 4.83%-5.74% on MMLU.

1 INTRODUCTION

Diffusion large language models (dLLMs) (Yi et al., 2024; Zhang et al., 2025; Nie et al., 2025; Ye et al., 2025; Yu et al., 2025b) have recently emerged as a competitive alternative to autoregressive LLMs (Achiam et al., 2023; Bai et al., 2023; Dubey et al., 2024), demonstrating strong performance and extending beyond open-source settings to closed-source models such as Gemini-Diffusion, Seed-Diffusion (Song et al., 2025), and Mercury (Khanna et al., 2025). Through an iterative unmask-and-remask process, dLLMs enable parallel decoding in a flexible order beyond left-to-right token generation in autoregressive LLMs, offering substantial flexibility and potential at test time.

In dLLMs, recent efforts (Zhao et al., 2025a; Tang et al., 2025; Zhao et al., 2025b; Wang et al., 2025b; Yang et al., 2025a; Huang et al., 2025) for reasoning enhancement mainly focus on training time, primarily through reinforcement learning (RL) (Ouyang et al., 2022; Shao et al., 2024; Feng et al., 2025b). These RL approaches introduce novel training schemes and achieve notable performance improvements. Complementary to these training-time approaches, another line of work explores enhancing reasoning by operating directly on the decoding process at test time.

Inspired by the success of test-time scaling in LLMs (Muennighoff et al., 2025; Snell et al., 2024; Wang et al., 2025a; Sun et al., 2024; Xu et al., 2025) and recent findings that RL primarily improves sampling efficiency rather than intrinsic capability (Yue et al., 2025; Chen et al., 2025b), we instead focus on inference. To date, a few works have explored test-time scaling for reasoning enhancement in dLLMs. HEX (Lee et al., 2025) activates implicit semi-autoregressive experts by varying block sizes, while RFG (Chen et al., 2025a) guides generation at the logit level using an additional fine-tuned model. Although these methods can boost performance, they typically involve increased test-time computation. Our goal is to reduce the substantial redundancy inherent in the voting paradigm.

Based on the consistency analysis, we further identify a simple yet crucial observation that repeated tokens frequently appear across multiple samples for the same input, as illustrated in Figure 2(b) and quantified in Table 1. Furthermore, we connect this observation with the remasking mechanism of dLLMs which allows the model to mask and regenerate an arbitrary number of decoded tokens at

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

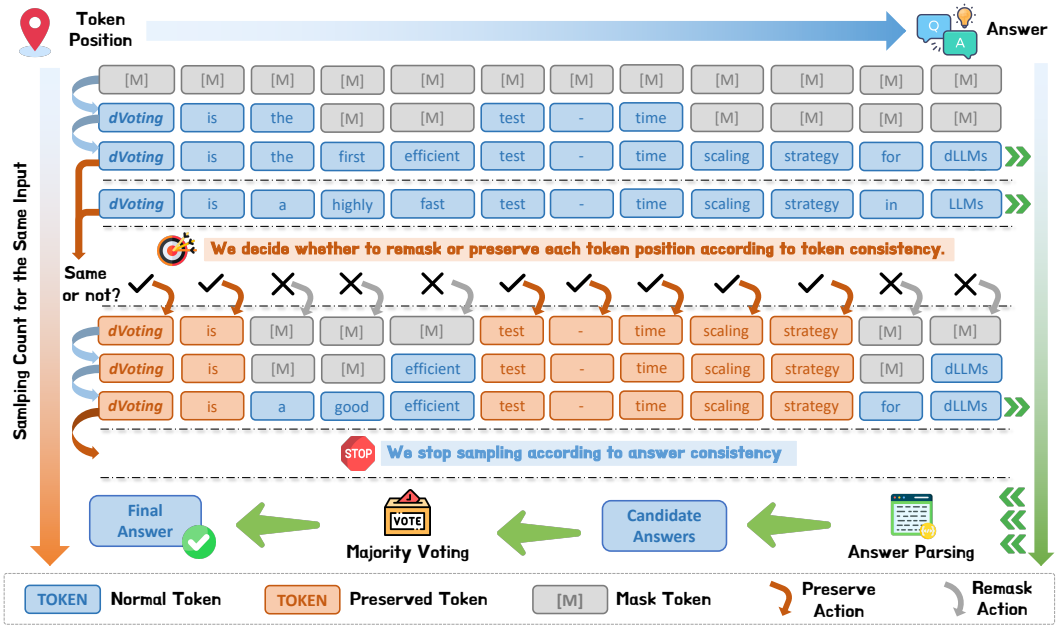


Figure 1: Overview of DVOTING. For each prompt, our DVOTING preserves consistent tokens in previous generations and remasks the remaining tokens to initiate subsequent sampling, and terminates the process early when candidate answers satisfy consistent criteria.

arbitrary positions within a sequence[1], making it well-suited to reduce the redundancy revealed by this observation. Accordingly, we propose a simple yet fast voting strategy (Figure 1). Specifically, we identify uncertain tokens based on token consistency, iteratively remask and regenerate them for refinement, and finally aggregate candidate answers via voting.

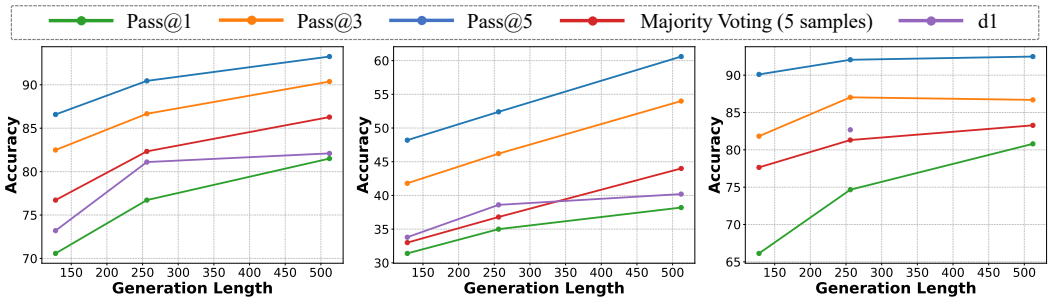
We conduct extensive experiments across various reasoning benchmarks on two popular dLLMs (LLaDA (Nie et al., 2025) & Dream (Ye et al., 2025)) to evaluate the effectiveness of our method. The results demonstrate that our method boosts the performance across all benchmarks covering mathematical, scientific, and general reasoning. For instance, our method yields performance gains of 6.22%–7.66% on GSM8K (Cobbe et al., 2021) and 4.40%–7.20% on MATH500 (Lightman et al., 2023) on LLaDA. Our method further achieves a leading performance–efficiency trade-off. Additionally, we present the robustness of our method under various configurations.

In conclusion, we propose DVOTING, a training-free, simple yet effective voting strategy that boosts performance. Based on the empirical study, we identify and quantify a key observation that many tokens are repeatedly generated across multiple sampling runs, and relate this to the remasking mechanism of dLLMs. Accordingly, we introduce the remask sampling strategy, which iteratively remasks and regenerates selected tokens to obtain multiple candidate generations and aggregates via voting to enhance performance. Extensive evaluations demonstrate the effectiveness of DVOTING. Together, DVOTING establishes the first baseline and provides a foundation for efficient test-time scaling in dLLMs, further unlocking their potential at test time.

2 Related Work

Overview of Diffusion Language Models. Diffusion models (Ho et al., 2020; Song & Ermon, 2019; Song et al., 2020) have shown strong generative capabilities in continuous domains (*e.g.*, image (Rombach et al., 2022; Peebles & Xie, 2023), video (Ho et al., 2022; Brooks et al., 2024), and audio (Liu et al., 2023; Evans et al., 2024)), while their extension to language modeling remains challenging due to the discrete nature of text. Recent works (Austin et al., 2021; Sahoo et al., 2024; Lou et al., 2023; Zheng et al., 2024; Cheng et al., 2025) address this issue by formulating diffusion processes directly over token spaces, typically through masked token prediction, which enables parallel generation and relaxes the strict autoregressive constraint. Building on this paradigm,

[1] *e.g.*, "dLLMs are very efficient at test time." → "dLLMs [MASK] [MASK] [MASK] at [MASK] time." → "dLLMs can be fast at inference time."



(a) GSM8K (b) MATH500 (c) ARC-C

Figure 2: Empirical study on LLaDA-8B-Instruct. We report the performance of five strategies: (1) Pass@1; (2) Pass@3; (3) Pass@5; (4) Majority voting (5 samples), which denotes the results of standard test-time scaling strategies; and (5) d1, which represents the performance of RL-enhanced models. We report the results of GSM8K, MATH500, and ARC-C in (a), (b), and (c), respectively.

dLLMs (Nie et al., 2025; Ye et al., 2025; Khanna et al., 2025; Song et al., 2025; Bie et al., 2025) have demonstrated competitive performance compared to autoregressive models at the billion-parameter scale, indicating the practical viability of diffusion for language generation. Moreover, diffusion language models have been increasingly explored in advanced settings such as reasoning (Zhu et al., 2025; Zhao et al., 2025a; Tang et al., 2025; Lin et al., 2025), multimodal generation Yang et al. (2025b); Li et al. (2025b); Yu et al. (2025b); You et al. (2025), and code synthesis (Gong et al., 2025; Khanna et al., 2025; Peng et al.), reflecting the growing scope and maturity of this research direction (Yu et al., 2025a; Li et al., 2025c).

Test-Time Scaling in Language Models. Test-time scaling (Welleck et al., 2024; Snell et al., 2024; Muennighoff et al., 2025; OpenAI, 2024) has emerged as an effective alternative to training-time scaling, aiming to elicit stronger reasoning capabilities by allocating additional computation during inference. In autoregressive language models, test-time scaling has been extensively studied through techniques such as chain-of-thought (CoT) prompting (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2022), best-of-N sampling (Sun et al., 2024; Wang et al., 2025a; Xu et al., 2025), and self-consistency (Wang et al., 2022; 2024; Aggarwal et al., 2023). However, test-time scaling in dLLMs remains relatively underexplored. Recent studies provide initial evidence of its potential: HEX (Lee et al., 2025) shows that aggregating diverse masking schedules during inference can substantially improve performance, while RFG (Chen et al., 2025a) introduces guidance from an additional fine-tuned model by operating at the logits level. PG-DLM (Dang et al., 2025) iteratively refines denoising trajectories to improve performance. These methods provide valuable references for test-time scaling in dLLMs. However, test-time scaling efficiency (Sun et al., 2024; Xu et al., 2025; Feng et al., 2025a; Ma et al., 2024; Zhu et al., 2024) is equally critical, as it directly addresses the substantial inference cost and determines the practicality of such approaches in real-world settings. To date, efficient test-time scaling for dLLMs remains unexplored; this gap is the primary focus of our work.

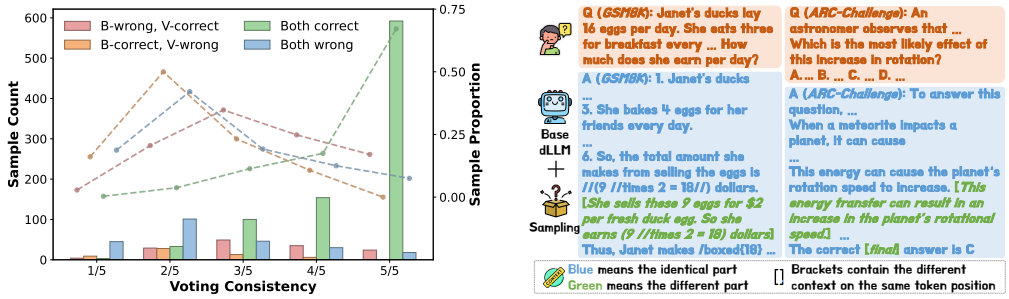
3 Preliminaries

We anchor our preliminaries in continuous-time diffusion language models defined over discrete vocabularies, with masked diffusion language models (MDLMs) serving as the main instantiation. MDLMs generate text by iteratively reconstructing a partially corrupted sequence rather than producing tokens in an autoregressive order. Starting from an initial clean sequence x_0, the model constructs a noised version x_t by independently masking each position with intensity $t \in [0, 1]$. Formally, the corruption distribution is

$$q(x_t \mid x_0) = \prod_{i=1}^{L} \left[(1-t)\,\delta(x_t^i = x_0^i) + t\,\delta(x_t^i = \text{[MASK]}) \right].$$

Once a corrupted sequence is obtained, a denoising network p_θ attempts to infer the original tokens at the masked locations. Since all masked positions are conditionally independent given x_t, the recovery model factorizes as

$$p_\theta(x_0 \mid x_t) = \prod_{i : x_t^i = \text{[MASK]}} p_\theta(x_0^i \mid x_t).$$



(a) Distribution of voting consistency level

(b) Cases on token consistency

Figure 3: (a) We report the distribution of voting consistency levels across different sample categories, defined by the correctness of the baseline and voting predictions. (b) We present cases illustrating token-level redundancy in dLLM sampling, drawn from GSM8K and ARC-C, respectively.

Training focuses on masked tokens, with the model learning to reconstruct original content from corrupted sequences. The resulting reconstruction loss serves as a surrogate objective that upper-bounds the negative log-likelihood.

Properties of dLLMs Fit Parallel Test-Time Scaling. During inference, dLLMs update all masked positions in parallel, producing token predictions for the entire sequence at each iteration while selectively remasking uncertain positions. In contrast to prior work that focuses on single-run decoding (Chen et al., 2025c; Wei et al., 2025), our work primarily targets parallel test-time scaling, where multiple outputs are generated in parallel and subsequently aggregated into a final prediction. Across multiple sampling runs, the remasking mechanism enables the model to selectively reuse reliable context from previous iterations, refining earlier decisions. This property makes dLLMs a natural and scalable backbone for parallel test-time scaling. Additionally, the inherently parallel decoding process of dLLMs further reduces inference latency.

Parallel Test-Time Scaling Can Work as an Alternative to RL in dLLMs. RL can be viewed as selecting higher-quality trajectories from multiple samples and increasing their sampling probability, thereby improving sampling efficiency rather than fundamentally enhancing the model's capacity. This perspective is also supported by recent work (Yue et al., 2025) in LLMs. As a major strategy of parallel test-time scaling, voting performs multiple sampling runs and selects the most consistent one as the final prediction. From this viewpoint, ***voting can be regarded as a training-free alternative that approximates the effect of RL***. As shown in Figure 2, we conduct experiments and further validate this claim in dLLMs: scaling test-time computation achieves performance comparable to advanced RL-based methods across various tasks. However, these significant performance gains come at the cost of increased inference overhead, which motivates us to improve efficiency.

4 METHODS

In this section, we introduce DVOTING to demonstrate the potential of simple voting strategies as a cost-effective alternative to RL in dLLMs. We first present the critical observation and then introduce the key designs of our DVOTING.

4.1 KEY OBSERVATIONS

We conduct an empirical study using majority voting with entropy-threshold parallel decoding on LLaDA with GSM8K. Based on the results, we present the key observation that motivates us to leverage the remasking capability of dLLMs to further improve efficiency.

Observation A. We partition samples into four categories based on whether the baseline and voting predictions are correct. We propose a metric called *voting consistency level*, defined as the fraction of votes received by the most frequent answer. As shown in Figure 3(a), we observe that samples correctly solved by both the baseline and voting methods are highly concentrated at high consistency levels, with 84.58% falling into the $4/5$ or $5/5$ bins. In contrast, the remaining categories are predominantly associated with low consistency levels, with 66.07% of samples receiving only $1/5$

or 2/5 votes. In summary, easier questions tend to exhibit higher voting consistency, indicating that sampling redundancy predominantly arises in such cases and can be reduced with minimal impact on performance.

Observation B. We further observe that for a given question, many token positions remain identical across multiple samples, as illustrated in Figure 3(b). This token-level redundancy indicates substantial overlap among sampled sequences and reveals additional opportunities to reuse confident predictions while focusing resampling on a small subset of uncertain token positions.

We propose a new metric termed *Non-Unique Position Rate at k (NUPR@k)* to quantify how frequently this phenomenon occurs. Given K sampled answers of equal length N for the same question, NUPR@k measures the fraction of token positions at which at least k out of K samples share an identical token. Formally, a token position is considered non-unique if at least k out of the K samples share the same token at that position, and NUPR@k is computed as the average fraction of such positions across all tokens and questions. As shown in Table 1, we report results on GSM8K, MATH500, and ARC-C under different generation lengths. We observe that NUPR@2 is consistently around 50%, while NUPR@3 remains around 20%, indicating substantial token-level redundancy.

Table 1: Results of NUPR@k on various benchmarks under two generation lengths. To be specific, we record the tokens of 5 samples and calculate the metric for each question.

Metric / Seq. Len.	GSM8K		MATH500		ARC-C	
	128	256	128	256	128	256
NUPR@2	0.6077	0.5473	0.4894	0.4383	0.4612	0.4812
NUPR@3	0.2109	0.1953	0.1327	0.1142	0.1562	0.2134

4.2 Remask Sampling

Based on these observations (A & B), we naturally connect the empirical findings with the inherent remasking capability of dLLMs and propose a simple yet effective remask sampling strategy (as shown in Figure 1). We present the detailed implementation pipeline in Appendix A.1.

Remask Sampling Strategy. Inspired by observation A, we stop sampling on time based on answer consistency (*e.g.*, the first several generations yield the same answer). Furthermore, inspired by observation B, we propose our core design to further improve efficiency. Unlike AR reasoning models, dLLMs rely more heavily on confidence for decoding quality (Chen et al., 2025c; Nie et al., 2025). Therefore, when a span of tokens remains consistent across multiple outputs, it suggests that the model assigns relatively high confidence to that span. Although such consistency does not guarantee correctness, it reliably indicates redundancy, as further resampling these tokens is unlikely to provide substantial benefit. Based on this intuition, instead of generating a full response at each sampling action, we selectively remask tokens and continue sampling based on the consistency analysis. In particular, we retain tokens that exhibit agreement across samples, as well as tokens from samples with highly consistent predicted answers, and perform subsequent sampling conditioned on the preserved tokens. This allows additional computation to be allocated to positions that are more worthwhile to explore, thereby improving its efficiency.

Additionally, in dLLMs, parallel decoding has been extensively explored and is a natural advantage of this paradigm. Following prior work (Ben-Hamu et al., 2025; Chen et al., 2025c; Wei et al., 2025), we adopt a simple entropy-threshold parallel decoding scheme that commits all token positions whose entropy falls below a predefined threshold α at each denoising step. In our main experiments, we set $\alpha = 0.3$ and provide an ablation study on this threshold. Finally, we stop sampling once the voting-based answer converges and adopt it as the final prediction.

5 Experiments

5.1 Experimental Setups

Baselines. We compare against four sampling baselines: the original results, majority voting, and two related methods, HEX (Lee et al., 2025) and RFG (Chen et al., 2025a). Specifically, for the original performance, we adopt the semi-autoregressive decoding strategy following the original paper; for majority voting (5 samples), we follow the configuration used in HEX. For HEX and RFG, we report either our reimplementations or the results from the original papers; the two methods require 25 and

Table 2: Results on mathematical reasoning benchmarks with LLaDA-8B-Instruct. $*$ represents the results of reimplementation. † indicates that RFG requires an extra instruction-tuned or RL-enhanced model as the policy model. We report Pass@1 accuracy along with the corresponding step count. The involved RL-enhanced methods require in-domain training data.

Method / Gen. Len.	Training Necessity	GSM8K (0-shot)			MATH500 (0-shot)		
		128	256	512	128	256	512
LLaDA-8B-Instruct	Pre-training	70.58% / 128.0	76.72% / 256.0	81.50% / 512.0	31.40% / 128.0	35.00% / 256.0	38.20% / 512.0
RL-Enhanced Models (in-domain)							
d1	SFT + RL	73.20% / 64.0	81.10% / 128.0	82.10% / 256.0	33.80% / 64.0	38.60% / 128.0	40.20% / 256.0
wd1	RL	-	80.80% / 128.0	82.30% / 256.0	-	34.40% / 128.0	39.00% / 256.0
IGPO	RL	-	83.60% / 256.0	-	-	42.80% / 256.0	-
Test-Time Scaling Strategies							
Majority Voting	Training-free	76.72% / 320.0	82.33% / 640.0	86.28% / 1280.0	33.00% / 320.0	36.80% / 640.0	44.00% / 1280.0
HEX*	Training-free	80.14% / 1600.0	85.75% / 3200.0	88.78% / 6400.0	39.60% / 1600.0	43.60% / 3200.0	47.40% / 6400.0
RFG	FT *or* RL†	-	81.30% / 512.0	-	-	39.60% / 512.0	-
DVOTING (ours)	Training-free	78.24% / 170.4	83.78% / 237.1	87.72% / 289.6	34.80% / 292.1	40.20% / 473.7	45.40% / 701.2

Table 3: Results on scientific and general reasoning benchmarks with LLaDA-8B-Instruct.

Method / Gen. Len.	Training Necessity	ARC-C (0-shot)		GPQA (0-shot)		MMLU (0-shot)	
		128	256	128	256	128	256
LLaDA-8B-Instruct	Pre-training	66.13% / 128.0	74.66% / 256.0	25.00% / 128.0	23.66% / 256.0	57.13% / 128.0	58.04% / 256.0
Test-Time Scaling Strategies							
Majority Voting	Training-free	77.65% / 320.0	81.31% / 640.0	22.54% / 320.0	25.45% / 640.0	62.27% / 320.0	63.11% / 640.0
HEX*	Training-free	82.67% / 1600.0	83.87% / 3200.0	25.67% / 1600.0	28.34% / 3200.0	64.28% / 1600.0	64.75% / 3200.0
DVOTING (ours)	Training-free	80.97% / 265.4	83.36% / 415.6	28.57% / 219.3	28.39% / 555.7	62.87% / 222.1	62.87% / 391.4

2 samples, respectively. Additionally, we include the original results reported in the paper for three RL methods: d1 (Zhao et al., 2025a), wd1 (Tang et al., 2025), and IGPO (Zhao et al., 2025b).

Inference Details. We follow the official inference settings to evaluate the original models from both the LLaDA series (Nie et al., 2025; Zhu et al., 2025) and Dream (Ye et al., 2025). For the baselines, we report the results from the original paper or our reimplementation, with the latter following the configurations specified in the original paper. We apply full denoising steps (N) for evaluating the original models and half denoising steps ($N/2$) for baselines (*e.g.*, majority voting and HEX) to generate a sequence of length N. We set the temperature to 0.6 for our proposed DVOTING and conduct experiments under standard generation lengths (128, 256, 512), with corresponding block sizes of 8, 16, and 32, respectively, following the semi-autoregressive strategy for models from the LLaDA series. For the Dream model, we implement a semi-autoregressive decoding strategy, as it is not supported by the original generation pipeline, while all other settings follow those used for the LLaDA series. We set an upper bound $n = 5$ on the total number of samples for our DVOTING.

Evaluation Details. We conduct extensive experiments across various benchmarks covering a broad range of reasoning tasks, including mathematical reasoning, scientific reasoning, and general high-level reasoning, using GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), ARC-C (Clark et al., 2018), MMLU (Hendrycks et al., 2021), and GPQA (Rein et al., 2024). Additionally, for models from the LLaDA (LLaDA-8B-I and LLaDA-1.5) and Dream (Dream-7B-I), we follow the simple-eval framework[2] for zero-shot evaluation and prompt the models to generate step-by-step reasoning. We implement voting within the same framework to process all candidate answers.

5.2 MAIN RESULTS

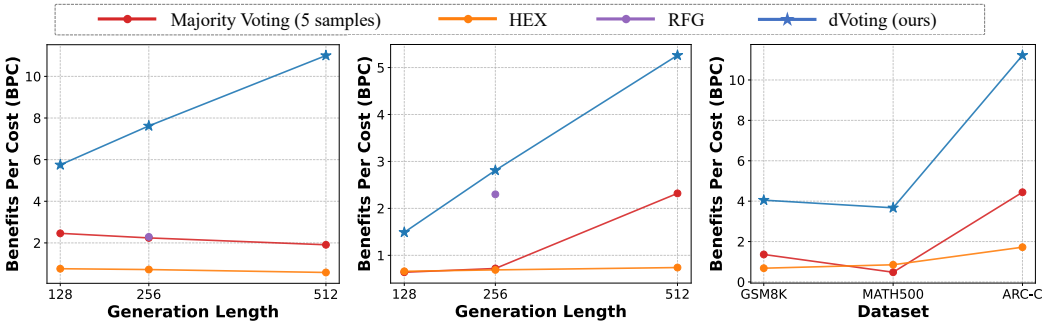
Results on the Native dLLM (LLaDA). We report the results of LLaDA-8B-Instruct in Tables 2 and 3. Compared to the original model and most in-domain RL methods, our DVOTING achieves consistent and substantial performance gains across different benchmarks and generation lengths: 6.22%–7.66% on GSM8K, 4.40%–7.20% on MATH500, 3.16%-14.84% on ARC-C, 3.57%-4.73% on GPQA, and 4.83%-5.74% on MMLU. Against test-time scaling baselines, DVOTING attains strong performance with the fewest steps, delivering 1.1–4.4× speedup over majority voting, 1.1–2.2× over RFG, and 5.5–22.1× over HEX.

Results on the AR-initialized dLLM (Dream). As shown in Table 4, our DVOTING also consistently achieves significant performance improvements across all benchmarks while using the fewest steps

[2] https://github.com/openai/simple-evals

Table 4: Results on Dream-7B-Instruct. $*$ represents the results of reimplementation. † indicates that RFG requires an extra instruction-tuned or RL-enhanced model as the policy model. We set top-p to 0.6 for the majority voting baseline.

Method / Gen. Len.	Training Necessity	GSM8K (0-shot)		MATH500 (0-shot)		ARC-C (0-shot)	MMLU (0-shot)
		128	256	128	256	128	128
Dream-7B-Instruct*	-	68.92% / 128.0	81.80% / 256.0	33.00% / 128.0	42.20% / 256.0	66.63% / 128.0	64.83% / 128.0
Test-Time Scaling Strategies							
Majority Voting	Training-free	72.33% / 320.0	84.53% / 640.0	34.20% / 320.0	44.00% / 640.0	77.73% / 320.0	68.51% / 320.0
HEX*	Training-free	77.48% / 1600.0	87.04% / 3200.0	43.60% / 1600.0	50.60% / 3200.0	88.14% / 1600.0	69.87% / 1600.0
RFG	FT *or* RL†	-	82.10% / 512.0	-	46.40% / 512.0	-	-
DVOTING (ours)	Training-free	75.44% / 206.2	86.96% / 290.4	42.20% / 320.7	48.80% / 498.5	86.09% / 221.9	69.80% / 119.6



(a) GSM8K on LLaDA (b) MATH500 on LLaDA (c) Benchmarks on Dream

Figure 4: Comparison of performance-efficiency trade-off between DVOTING and other test-time scaling baselines. We present the results on the LLaDA model for GSM8K and MATH500 in (a) and (b), respectively. We present the results on the Dream model for GSM8K, MATH500, and ARC-C under 128 generation length in (c). We mark our method with a star.

Table 5: Comparison on end-to-end latency. We set the generation length to 256. Avg. Speedup is computed relative to HEX.

Method	GSM8K	MATH500	ARC-C	Avg. Latency	Avg. Speedup
	Acc. / Latency	Acc. / Latency	Acc. / Latency		
LLaDA-8B-Instruct	76.72% / 38.7s	35.00% / 39.2s	74.66% / 22.2s	33.4s	12.75×
Test-Time Scaling Strategies					
Majority Voting	82.33% / 96.8s	36.80% / 98.0s	81.31% / 55.5s	83.4s	5.10×
HEX	85.75% / 508.1s	43.60% / 491.9s	83.87% / 276.3s	425.4s	1.00×
DVOTING (ours)	83.78% / 38.3s	40.20% / 44.2s	83.36% / 38.3s	40.3s	10.57×

among all test-time scaling baselines. Specifically, DVOTING delivers a 1.0–2.7× speedup over majority voting, a 1.0–1.8× speedup over RFG, and a 5.0–13.4× speedup over HEX.

DVOTING Pushes Performance–Efficiency Pareto Frontier Forward. We propose a unified metric, denoted as *benefits per cost (BPC)*, to quantify the performance–efficiency trade-off in the context of test-time scaling. Specifically, the metric measures the performance gain per unit of additional test-time cost, computed as the improvement over the original model normalized by the corresponding increase in the total number of denoising steps[3]. As shown in Figure 4, our DVOTING achieves superior performance–efficiency trade-off across all settings. Based on the LLaDA results, our DVOTING consistently achieves larger BPC improvements than other test-time scaling baselines as the generation length increases, demonstrating its clear advantage in long-generation length settings.

DVOTING Reduces End-to-End Latency. We then report the end-to-end latency and corresponding speedup to support our claims in Table 5. We use HEX as the baseline for speedup calculation. We observe that our method achieves significant end-to-end speedup (nearly 10× against the HEX baseline) while maintaining strong performance gains. The reduced latency further supports the effectiveness of our DVOTING in real-world deployment.

[3]We present an example (GSM8K on LLaDA-8B-Instruct under 128 generation length): $\frac{78.24\%-70.58\%}{170.4\ /\ 128.0} = 5.75$.

Table 6: Results on RL-enhanced model (LLaDA-1.5). $*$ represents the results of reimplementation. † indicates that RFG requires an extra instruction-tuned or RL-enhanced model as the policy model. We report Pass@1 accuracy along with the corresponding step count.

Method / Gen. Len.	Training Necessity	GSM8K (0-shot)		MATH500 (0-shot)		ARC-C (0-shot)	MMLU (0-shot)
		128	256	128	256	128	128
LLaDA-1.5*	-	72.48% / 128.0	78.01% / 256.0	30.60% / 128.0	33.40% / 256.0	68.52% / 128.0	58.27% / 128.0
Test-Time Scaling Strategies							
Majority Voting	Training-free	78.47% / 320.0	84.69% / 640.0	32.00% / 320.0	41.00% / 640.0	79.69% / 320.0	62.91% / 320.0
HEX (Lee et al., 2025)*	Training-free	82.41% / 1600.0	86.50% / 3200.0	39.60% / 1600.0	43.40% / 3200.0	86.69% / 1600.0	64.47% / 1600.0
RFG (Chen et al., 2025a)	FT *or* RL†	-	82.10% / 512.0	-	-	-	-
DVOTING (ours)	Training-free	79.15% / 158.8	84.53% / 223.1	34.40% / 280.1	41.20% / 443.1	82.63% / 267.7	63.10% / 224.0

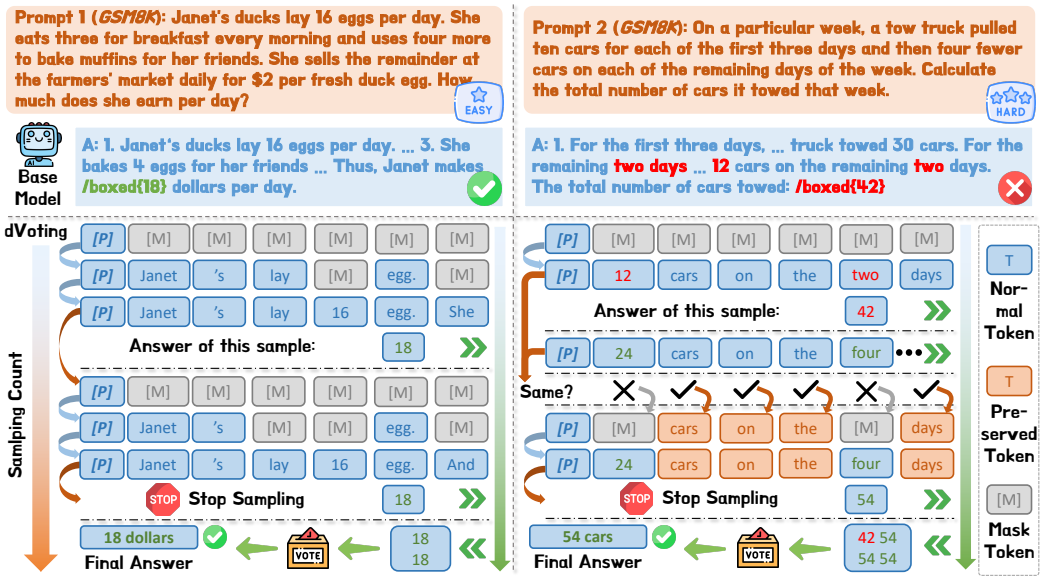


Figure 5: Case visualization. We present two cases where the original model yields a correct and an incorrect answer, respectively. Prompt-2 fails by incorrectly assuming a five-day week, while our method iteratively refines this part to produce the correct answer.

Generalization of DVOTING on RL-enhanced Models. We next evaluate the generalization of DVOTING on RL-enhanced models, such as LLaDA-1.5 fine-tuned with Variance-Reduced Preference Optimization (VRPO) (Zhu et al., 2025). As shown in Table 6, we conduct experiments across four benchmarks. The results demonstrate that DVOTING consistently brings significant performance improvements on RL-enhanced models while incurring only minimal additional inference cost, further confirming the generalization and effectiveness of our method.

5.3 VISUALIZATION

Figure 5 illustrates two GSM8K cases where LLaDA produces a correct and an incorrect prediction, respectively. For Prompt 1, the base model already yields the correct answer, and our DVOTING quickly identifies that no further remasking is needed, terminating sampling early. For Prompt 2, the base model fails, while our method performs five sampling iterations and aggregates the results via voting to recover the correct answer. Overall, our proposed DVOTING can adapt its computation according to problem difficulty automatically, efficiently solving simple cases while allocating more inference effort to harder ones to improve both efficiency and performance. We provide more cases (ARC-C and MATH500) in Appendix B.

5.4 ABLATION STUDY

We provide comprehensive ablation studies on remasking strategy, sampling upper bound n, and block size on LLaDA, evaluated on GSM8K, MATH500, and ARC-C. Furthermore, we conduct ablation on the entropy threshold α (provided in the Appendix C due to space limitations).

Table 7: Ablation on remasking strategies. We compare our DVOTING with some baselines. We report Pass@1 accuracy and the corresponding denoising step count.

Method	GSM8K		MATH500		ARC-C	
	Acc.	Denoising Steps	Acc.	Denoising Steps	Acc.	Denoising Steps
LLaDA-8B-Instruct	76.72%	256.0	35.00%	256.0	74.66%	256.0
Remasking Strategies						
Early Stopping	83.02%	367.0	38.60%	641.6	82.93%	588.3
Random Remasking	77.56%	265.1	37.60%	373.5	75.23%	467.8
Confidence-based Remasking	82.87%	270.6	39.80%	416.8	80.97%	492.9
dVoting (ours)	83.78%	237.1	40.20%	473.7	83.36%	415.6

Table 8: Ablation on sampling upper bound n. We evaluate sampling upper bounds of 1, 5, 9, 13, and 17 while keeping all other hyperparameters fixed, and report Pass@1 accuracy and corresponding step count. We provide the majority voting baseline as a reference.

Dataset / Gen. Len.	Baselines		DVOTING				
	LLaDA-8B-Instruct	Majority Voting	n=1	n=5	n=9	n=13	n=17
GSM8K / 128	70.58% / 128.0	76.72% / 320.0	70.13% / 58.2	78.24% / 170.4	79.68% / 348.6	80.89% / 515.5	80.89% / 683.2
GSM8K / 256	76.72% / 256.0	82.33% / 640.0	77.55% / 85.3	83.78% / 237.1	85.97% / 477.3	86.13% / 708.6	86.20% / 937.7
MATH500 / 128	31.40% / 128.0	33.00% / 320.0	27.60% / 72.0	34.80% / 292.1	36.40% / 573.3	37.60% / 837.9	39.20% / 1105.0
MATH500 / 256	35.00% / 256.0	36.80% / 640.0	36.20% / 120.5	40.20% / 473.7	42.20% / 928.2	43.20% / 1350.7	43.20% / 1779.3

Ablation on Remasking Strategy. We perform additional experiments to better illustrate the value of our strategy in Table 7. Specifically, we implemented three baselines: early stopping without token reuse, random remasking without the consistency-aware policy (remasking 50% of the tokens), and confidence-based remasking (remasking the tokens with confidence lower than 0.95). We observe that our method achieves the highest accuracy with fewer denoising steps. The results show the effectiveness of our remasking strategy.

Ablation on Sampling Upper Bound. We first ablate the sampling upper bound n under two generation lengths. As shown in Table 8, performance consistently improves with larger n across both benchmarks and both generation lengths, until reaching a saturation point. This demonstrates that DVOTING can effectively scale test-time computation to boost performance, consistent with the test-time scaling law stated in prior work (Snell et al., 2024). In addition, we compare DVOTING under higher cost budgets with HEX in Appendix C.2 to further show the advantage of our method.

Table 9: Ablation on block size. We evaluate block sizes of 4, 8, 16, 32, and 64 while keeping all other hyperparameters fixed, and report Pass@1 accuracy along with the corresponding step count. We provide the majority voting baseline as a reference.

Dataset / Gen. Len.	Baselines		DVOTING				
	LLaDA-8B-Instruct	Majority Voting	block size=4	block size=8	block size=16	block size=32	block size=64
GSM8K / 128	70.58% / 128.0	76.72% / 320.0	79.23% / 168.6	78.24% / 170.4	77.71% / 176.1	78.01% / 177.8	75.82% / 184.3
GSM8K / 256	76.72% / 256.0	82.33% / 640.0	83.70% / 227.2	84.23% / 231.5	83.78% / 237.1	84.23% / 238.8	84.76% / 238.6
MATH500 / 128	31.40% / 128.0	33.00% / 320.0	35.00% / 281.0	34.80% / 292.1	34.00% / 296.4	34.00% / 297.9	34.80% / 308.8
MATH500 / 256	35.00% / 256.0	36.80% / 640.0	40.00% / 448.1	40.00% / 469.8	40.20% / 473.7	40.40% / 478.5	40.40% / 463.7

Ablation on Block Size. We next ablate the block size, with results summarized in Table 9. Across block sizes ranging from 4 to 64, DVOTING consistently outperforms both the original results and the majority voting baseline, indicating that its performance is robust to this hyperparameter and further demonstrating the effectiveness of our approach.

6 CONCLUSION

In this work, we address the challenge of the high inference cost incurred by test-time scaling in dLLMs. We first identify a key empirical insight: repeated tokens frequently emerge across multiple samples for the same prompt. Building on this observation, we leverage the remasking mechanism of dLLMs and propose DVOTING, a fast voting strategy tailored to dLLMs. Extensive evaluations demonstrate that our method consistently enhances reasoning performance with modest extra inference cost, achieving a leading performance–efficiency trade-off. Overall, our work provides a foundation for future test-time scaling and unlocks the inference-time potential of dLLMs.

AI USE STATEMENT

In preparing this work, we used generative AI tools (large language models) only for tasks in the recommended-disclosure category of the ICLR policy, namely improving the readability of the manuscript (correcting grammar, refining wording and sentence structure) and making small formatting changes to the LaTeX source. We did not use generative AI tools for any task that requires disclosure: the research ideas, the design of the proposed method and its algorithms, the implementation, the experimental design, the data processing, and the analysis and interpretation of results were all carried out by the authors, and no code, figures, tables, or citations were produced by AI tools. Only anonymized excerpts of the manuscript text were provided to these tools; no proprietary data or unpublished experimental results were shared. Every AI-suggested edit was checked and, where necessary, revised by the authors, and we take full responsibility for the content of this paper. Since AI assistance was confined to presentation, it has no bearing on the reproducibility of our results, all of which can be reproduced from our code and configurations.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 5.1 details the evaluation setup, including the baselines, inference settings, and evaluation protocol; Appendix A.1 presents the complete algorithm of DVOTING; and Appendix G describes the hardware used for all experiments. Together, these details allow our results to be replicated quickly.

REFERENCES

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

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *arXiv preprint arXiv:2305.11860*, 2023.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

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.

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.

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.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

Tianlang Chen, Minkai Xu, Jure Leskovec, and Stefano Ermon. Rfg: Test-time scaling for diffusion large language model reasoning with reward-free guidance. *arXiv preprint arXiv:2509.25604*, 2025a.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms. *arXiv preprint arXiv:2509.26488*, 2025c.

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.

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. *arXiv preprint arXiv: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. *arXiv preprint arXiv:2110.14168*, 2021.

Meihua Dang, Jiaqi Han, Minkai Xu, Kai Xu, Akash Srivastava, and Stefano Ermon. Inference-time scaling of diffusion language models with particle gibbs sampling. *arXiv preprint arXiv:2507.08390*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. In *ICML*, 2024.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025a.

Sicheng Feng, Kaiwen Tuo, Song Wang, Lingdong Kong, Jianke Zhu, and Huan Wang. Rewardmap: Tackling sparse rewards in fine-grained visual reasoning via multi-stage reinforcement learning. *arXiv preprint arXiv:2510.02240*, 2025b.

Sicheng Feng, Song Wang, Shuyi Ouyang, Lingdong Kong, Zikai Song, Jianke Zhu, Huan Wang, and Xinchao Wang. Can mllms guide me home? a benchmark study on fine-grained visual reasoning from transit maps. *arXiv preprint arXiv:2505.18675*, 2025c.

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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models. *arXiv preprint arXiv:2505.10446*, 2025.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Jihoon Lee, Hoyeon Moon, Kevin Zhai, Arun Kumar Chithanar, Anit Kumar Sahu, Soummya Kar, Chul Lee, Souradip Chakraborty, and Amrit Singh Bedi. Test-time scaling in diffusion llms via hidden semi-autoregressive experts. *arXiv preprint arXiv:2510.05040*, 2025.

Qi Li, Runpeng Yu, Haiquan Lu, and Xinchao Wang. Every step counts: Decoding trajectories as authorship fingerprints of dllms. *arXiv preprint arXiv:2510.05148*, 2025a.

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. *arXiv preprint arXiv:2505.16839*, 2025b.

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

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 *ICLR*, 2023.

Nianyi Lin, Jiajie Zhang, Lei Hou, and Juanzi Li. Boundary-guided policy optimization for memory-efficient rl of diffusion large language models. *arXiv preprint arXiv:2510.11683*, 2025.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. Non-myopic generation of language models for reasoning and planning. *arXiv preprint arXiv:2410.17195*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *EMNLP*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

OpenAI. OpenAI o1. https://openai.com/o1/, 2024.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Fred Zhangzhi Peng, Shuibai Zhang, and Alex Tong. contributors. open-dllm: Open diffusion large language models.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *COLM*, 2024.

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

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

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

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

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.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. In *NeurIPS*, 2024.

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.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *arXiv preprint arXiv:2408.13457*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *arXiv preprint arXiv:2503.01422*, 2025a.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025b.

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. In *NeurIPS*, 2022.

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.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Jun Liu, Qika Lin, and Zhiyong Wu. ϕ-decoding: Adaptive foresight sampling for balanced inference-time exploration and exploitation. *arXiv preprint arXiv:2503.13288*, 2025.

Jingyi Yang, Guanxu Chen, Xuhao Hu, and Jing Shao. Taming masked diffusion language models via consistency trajectory reinforcement learning with fewer decoding step. *arXiv preprint arXiv:2509.23924*, 2025a.

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*, 2025b.

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.

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

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.

Runpeng Yu, Qi Li, and Xinchao Wang. Discrete diffusion in large language and multimodal models: A survey. *arXiv preprint arXiv:2506.13759*, 2025a.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*, 2025b.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *NeurIPS*, 2025.

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.

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*, 2025a.

Siyan Zhao, Mengchen Liu, Jing Huang, Miao Liu, Chenyu Wang, Bo Liu, Yuandong Tian, Guan Pang, Sean Bell, Aditya Grover, et al. Inpainting-guided policy optimization for diffusion large language models. *arXiv preprint arXiv:2509.10396*, 2025b.

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.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

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.

Jiace Zhu, Yingtao Shen, Jie Zhao, and An Zou. Path-consistency: Prefix enhancement for efficient inference in llm. *arXiv preprint arXiv:2409.01281*, 2024.

APPENDIX

In Appendix A.1, we present the detailed process of our DVOTING. Appendix B includes additional case analysis to further demonstrate the effectiveness of our DVOTING. We present more experiments and results in Appendix C. We further discuss future works in Appendix D. Finally, we provide the impact statement in Appendix E, license statement in Appendix F, and computing resources in Appendix G.

A The Details of DVOTING **15**

A.1 The Detailed Algorithm of the Consistency-Guided Designs 15

B Case Analysis **15**

C Supplementary Experimental Results **15**

C.1 Ablation on Entropy Threshold 15

C.2 DVOTING with higher cost budget 16

D Limitations & Future Work **16**

E Impact Statement **17**

F License **17**

G Computing Resources **18**

A THE DETAILS OF DVOTING

A.1 THE DETAILED ALGORITHM OF THE CONSISTENCY-GUIDED DESIGNS

We provide the detailed algorithm for the pipeline of our DVOTING in Algorithm 1. The entire pipeline is simple and easy to implement. We implement this pipeline with the PyTorch library (Paszke et al., 2019).

B CASE ANALYSIS

As shown in Figure A1, we present additional examples from ARC-C and MATH500, further demonstrating that our method quickly produces correct answers on simple problems and refines initially incorrect answers on more challenging ones. Prompt 1 is an easy ARC-C question, where DVOTING early-stops after two samplings upon obtaining consistent answers. In contrast, Prompt 2 is a harder MATH500 problem. While the base model outputs an incorrect answer in the first sampling, DVOTING refines key tokens through multiple samplings and stops after four sampling runs when a consistent answer is observed, yielding the correct result.

C SUPPLEMENTARY EXPERIMENTAL RESULTS

C.1 ABLATION ON ENTROPY THRESHOLD

We further conduct an ablation study on the entropy threshold α. As shown in Table A1, DVOTING maintains strong performance and a favorable efficiency trade-off across a broad range of threshold values (*e.g.*, 0.1-0.7). A noticeable performance drop is observed only when the threshold is set excessively high, such as α=0.9, where DVOTING still outperforms the original results. Moreover, the results follow a test-time scaling trend, with increased computation leading to improved performance.

Algorithm 1 Detailed Process of dVoting

Require: Prompt p, dLLM f_θ, maximum samples n, generation length L

$X_{\text{all}} \leftarrow \emptyset$ ▷ store all sampled generations
for $i = 1$ to n **do**
  Initialize sequence x with prompt tokens and masked generation positions
  Initialize a Boolean array $m \in \{0, 1\}^L$ representing whether remask or not
  if $X_{\text{all}} \neq \emptyset$ **then**
    for each generation position t **do**
      if The token at position t remain consistent across at least two samples **then**
        Fix the token at position t
        Set $m[t] \leftarrow$ False
      else
        Mask the token at position t
        Set $m[t] \leftarrow$ True
      end if
    end for
    if The number of consistent candidate answers exceeds half of the sampling upper bound **then**
      Fix all tokens
      Set $m[t] \leftarrow$ False for all $t \in \{1, \ldots, L\}$
    end if
  end if
  if all entries of m are False **then**
    break ▷ *Stop sampling here!*
  end if
  if there exist masked tokens in x **then**
    Perform entropy-threshold-based parallel decoding with a semi-autoregressive strategy ▷ *Details omitted as this is a typical strategy!*
  end if
  Append the completed sequence x to X_{all}
end for
return X_{all}
Perform voting on candidate answers in X_{all} to get final prediction ▷ *Voting part is here!*

Overall, this ablation study further demonstrates the robustness and effectiveness of our proposed method.

Table A1: Ablation on entropy threshold α. We evaluate entropy thresholds of 0.1, 0.3, 0.5, 0.7, and 0.9 while keeping all other hyperparameters fixed, and report Pass@1 accuracy and corresponding step count. We provide the majority voting baseline as a reference.

Dataset / Gen. Len.	Baselines		dVoting				
	LLaDA-8B-Instruct	Majority Voting	α=0.1	α=0.3	α=0.5	α=0.7	α=0.9
GSM8K / 128	70.58% / 128.0	76.72% / 320.0	78.24% / 213.7	78.24% / 170.4	77.48% / 152.6	77.18% / 140.4	74.91% / 136.4
GSM8K / 256	76.72% / 256.0	82.33% / 640.0	84.76% / 299.9	83.78% / 237.1	83.70% / 206.1	82.48% / 188.6	80.14% / 189.4
MATH500 / 128	31.40% / 128.0	33.00% / 320.0	34.80% / 345.7	34.80% / 292.1	33.80% / 260.9	33.60% / 242.5	31.40% / 221.4
MATH500 / 256	35.00% / 256.0	36.80% / 640.0	42.60% / 565.4	40.20% / 473.7	40.80% / 411.0	39.20% / 378.0	35.20% / 361.5

C.2 dVoting with higher cost budget

The results in Table A2 further show that our dVoting, with increasing cost, achieves better performance than HEX at a lower cost.

D Limitations & Future Work

Our method and experiments primarily focus on the language modality. As an emerging architecture, diffusion language models have demonstrated performance comparable to autoregressive models in

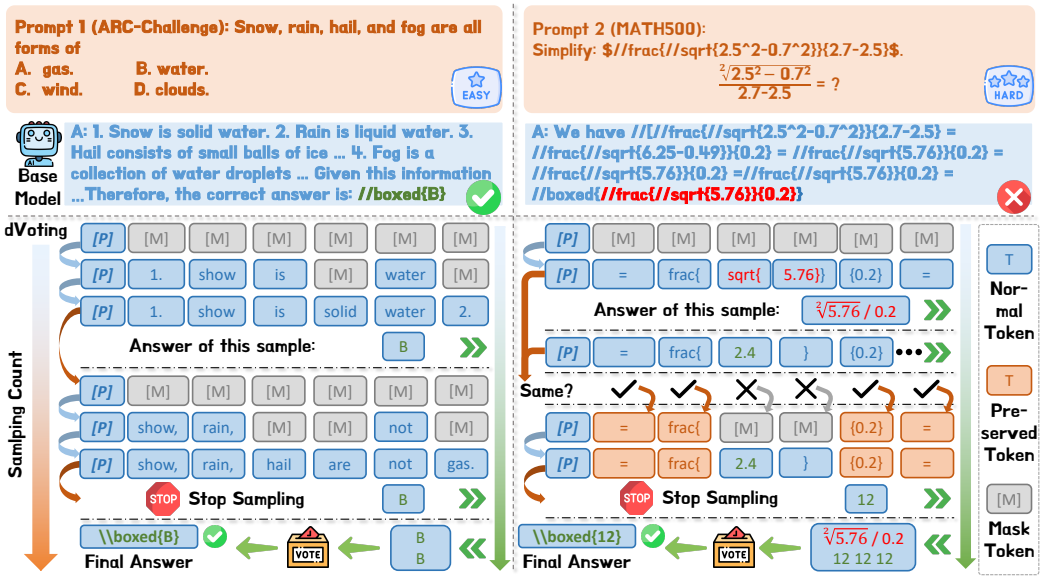


Figure A1: Additional case visualization.

Table A2: DVOTING with a higher cost budget. We report Pass@1 accuracy and the corresponding step count.

Dataset / Gen. Len.	Baselines		DVOTING
	LLaDA-8B-I	HEX	dVoting
GSM8K / 128	70.58% / 128.0	80.14% / 1600.0	80.89% / 683.2
GSM8K / 256	76.72% / 256.0	85.75% / 3200.0	86.20% / 937.1
MATH500 / 128	31.40% / 128.0	39.60% / 1600.0	39.60% / 1105.0
MATH500 / 256	35.00% / 256.0	43.60% / 3200.0	43.60% / 1779.3

language tasks, while also showing promise for extension to other modalities and tasks (Li et al., 2025a), such as multimodal question answering (Yu et al., 2025b; You et al., 2025; Yang et al., 2025b; Feng et al., 2025c; Yu et al., 2025a). Exploring parallel test-time scaling in these broader settings is a promising direction for future work. Moreover, parallel test-time scaling is simple to implement (easy to code) and computationally lightweight (easy to run), making it easy to adopt and follow. In practice, DVOTING can be applied to resource-constrained and cost-sensitive deployment settings, such as on-device reasoning and large-scale inference services.

E IMPACT STATEMENT

This work investigates efficient test-time scaling for dLLMs by proposing a simple voting strategy that leverages the remasking mechanism to focus computation on uncertain tokens. The approach improves reasoning performance while significantly reducing inference cost, making test-time scaling more practical under limited computational budgets. By improving inference efficiency, the proposed approach reduces computational overhead and resource consumption, thereby making large-scale model deployment more feasible in practice. The method does not introduce additional ethical or societal concerns beyond those commonly associated with large language models.

F LICENSE

The model and dataset included in this paper are publicly available. We followed their corresponding licenses for our experiments. The specific licenses for the model and dataset used are as follows:

- LLaDA[4] ..... MIT License

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

- LLaDA1.5[5] ..... MIT License
- Dream[6] ..... Apache-2.0 License
- GSM8K[7] ..... MIT License
- MATH500[8] ..... MIT License
- ARC-C[9] ..... cc-by-sa-4.0 License
- GPQA[10] ..... cc-by-4.0 License
- MMLU[11] ..... MIT License

To ensure fair and reproducible evaluation, we implement all inference procedures by adhering closely to the official documentation and recommended practices of each model.

G Computing Resources

The experiments were conducted on a server with an AMD EPYC 9354 32-Core Processor, 755 GB system memory, and 8 NVIDIA RTX A6000 GPUs with 48 GB VRAM. In our experiments, we used data parallelism to accelerate the evaluation process; each experiment was completed within 24 hours.

[5] https://huggingface.co/GSAI-ML/LLaDA-1.5
[6] https://huggingface.co/Dream-org/Dream-v0-Instruct-7B
[7] https://huggingface.co/datasets/openai/gsm8k.
[8] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[9] https://huggingface.co/datasets/allenai/ai2_arc
[10] https://huggingface.co/datasets/Idavidrein/gpqa
[11] https://huggingface.co/datasets/cais/mmlu