

# Wide-In, Narrow-Out: Revokable Decoding for Efficient and Effective DLLMs

Feng Hong<sup>1,\*</sup> Geng Yu<sup>1,\*</sup> Yushi Ye<sup>1</sup> Haicheng Huang<sup>1</sup>  
 Huangjie Zheng<sup>2</sup> Ya Zhang<sup>3</sup> Yanfeng Wang<sup>3</sup> Jiangchao Yao<sup>1</sup>  
<sup>1</sup>Cooperative Medianet Innovation Center, Shanghai Jiao Tong University  
<sup>2</sup>Apple <sup>3</sup>School of Artificial Intelligence, Shanghai Jiao Tong University  
 {feng.hong, Sunarker}@sjtu.edu.cn

## Abstract

Diffusion Large Language Models (DLLMs) have emerged as a compelling alternative to Autoregressive models, designed for fast parallel generation. However, existing DLLMs are plagued by a severe quality-speed trade-off, where faster parallel decoding leads to significant performance degradation. We attribute this to the irreversibility of standard decoding in DLLMs, which is easily polarized into the wrong decoding direction along with early error context accumulation. To resolve this, we introduce Wide-In, Narrow-Out (WINO), a training-free decoding algorithm that enables *revokable decoding* in DLLMs. WINO employs a parallel draft-and-verify mechanism, aggressively drafting multiple tokens while simultaneously using the model’s bidirectional context to verify and re-mask suspicious ones for refinement. Verified in open-source DLLMs like LLaDA and MMaDA, WINO is shown to decisively improve the quality-speed trade-off. For instance, on the GSM8K math benchmark, it accelerates inference by  $6\times$  while improving accuracy by 2.58%; on Flickr30K captioning, it achieves a  $10\times$  speedup with higher performance. More comprehensive experiments are conducted to demonstrate the superiority and provide an in-depth understanding of WINO. <sup>1</sup>

## 1 Introduction

Autoregressive (AR) large language models (Radford et al., 2018, 2019), such as the GPT series (OpenAI, 2022), have shown impressive performance in a ranging of language tasks. However, their foundational token-by-token generation mechanism introduces inherent limitations, including severe inference latency, susceptibility to error propagation (Stechly et al., 2023; Valmeekam et al., 2023), and challenges in maintaining global coherence (Mei et al., 2025). In response, Diffusion Large Language Models (DLLMs) have emerged as a compelling non-autoregressive alternative, architected to overcome these bottlenecks. By generating tokens simultaneously (Li et al., 2022), DLLMs theoretically enable massive inference acceleration, while their native bidirectional attention offers improved consistency. The immense potential of DLLMs has been showcased by proprietary, closed-source systems (e.g., Mercury Coder (Inception Labs, 2025) and Gemini Diffusion (Google DeepMind, 2025)), which have demonstrated astonishing speeds exceeding 1,000 tokens per second, serving as a powerful proof-of-concept.

Despite this promise, the performance of open-source DLLMs has been still disappointing. One critical bottleneck is that they are caught in a severe quality-speed trade-off dilemma. Specifically, to achieve high-quality output, these models are often forced to decode slowly, generating just one

\* Equal contribution

<sup>1</sup> Code: <https://github.com/Feng-Hong/WINO-DLLM>

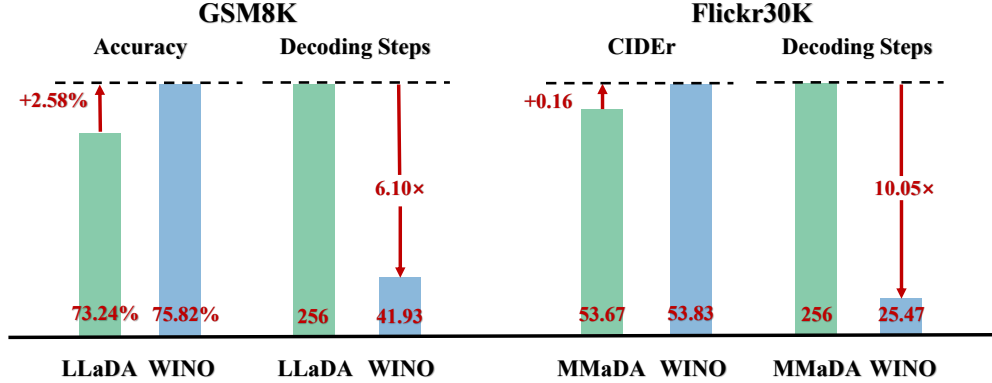


Figure 1: Demonstration of speedup and performance improvement.

token at a time, which negates their primary architectural advantage. Attempting to accelerate inference by generating multiple tokens in parallel invariably leads to a significant degradation in output quality (Nie et al., 2025). This stark trade-off has largely prevented the open-source DLLMs from becoming a viable, high-performance alternative to their AR counterparts.

We attribute this trade-off to a fundamental flaw in DLLMs (Ou et al., 2025; Sahoo et al., 2024): its irreversibility of the standard decoding process. Specifically, the standard generation in diffusion steps typically begins with a sequence of `[MASK]` tokens, which are then filled in greedily. Once a token is decoded, the decision is final and cannot be revised, even as more informative context becomes available in later steps. However, this is challenging for parallel decoding, where initial tokens are generated with very limited information, easily causing early errors to become permanently embedded, accumulated and propagated throughout the output. Therefore, such a rigid process essentially prevents DLLMs from using their greatest strength of bidirectional attention (Seo et al., 2017) to refine the early errors when the context progressively becomes rich.

To resolve this problem, we introduce Wide-In, Narrow-Out (WINO), a novel decoding algorithm that enables revokable decoding for DLLMs. WINO employs a novel draft-and-verify procedure that operates in parallel. At each step, a draft module aggressively proposes multiple new tokens based on a lenient threshold (the “Wide-In”). Concurrently, a verify module leverages the newly enriched global context to re-evaluate all previously generated tokens. Any token that fails a stricter verification check is re-masked for refinement in a future step (the “Narrow-Out”). This mechanism brings two merits: 1) it breaks the irreversibility of the conventional decoding in DLLMs, allowing the early error to be corrected for better performance; 2) it permits more aggressive token generation in each diffusion step for faster speedup with quality guarantee. Besides, our WINO is training-free and play-and-plug, which enables the general DLLMs to be both highly efficient and effective.

Our extensive experiments show that when applied to existing open-source models like LLaDA (Nie et al., 2025) and MMaDA (Yang et al., 2025), WINO achieves massive speedups, and also consistently improves model accuracy on both language and visual-language tasks. For instance, as shown in Fig. 1, on the GSM8K (Cobbe et al., 2021) math reasoning benchmark, WINO accelerates inference by  $6\times$  while simultaneously increasing accuracy by 2.58%, and on Flickr30K (Young et al., 2014) image captioning benchmark, it speedup decoding by  $10\times$  with even higher performance. By making decoding revokable, WINO unlocks the latent power of DLLMs in this area.

## 2 Related Work

**Diffusion-based Language Models.** Diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2021), originally popularized in image generation (Nichol et al., 2022; Rombach et al., 2022; Saharia et al., 2022), have recently gained attention as an alternative to autoregressive language models (ARLMs) for text generation. This expansion from continuous domain to discrete domain is first studied by Sohl-Dickstein et al. (2015). Subsequently, D3PM (Austin et al., 2021a) provides a general framework which models the diffusion forward process as a discrete state Markov chain defined by the multiplication of specific transition matrices over discrete time steps. Campbell et al. (2022) later expands D3PM to a continuous time setting, utilizing the theory of continuous time

Markov chain(CTMC). More recently, research on masked diffusion models(MDMs) derived from the absorbing state diffusion in D3PM has shown promising results both in small-scale models (*e.g.*, MDLM (Sahoo et al., 2024) and RADD (Ou et al., 2025)) and large-scale implementations (*e.g.*, LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025)). Extending this line of work, MMaDA (Yang et al., 2025) introduces a novel class of multimodal large diffusion models featuring a shared probabilistic formulation and a modality-agnostic architecture.

**DLLM Acceleration Techniques.** The existing acceleration study for DLLMs falls into two directions: KV cache and sampling compression. The former targets to build the KV cache for DLLMs due to its bidirectional full attention mechanism, unlike the causal attention of ARLMs. Typical works like Block Diffusion (Arriola et al., 2025), Fast-dLLM-cache (Wu et al., 2025) and dLLM-cache (Liu et al., 2025) respectively explore different caching mechanisms, which shows promising performance for speedup. Note that this direction is out of the scope of our work here. The latter direction focuses on optimizing the sampling process itself. For the classic low-confidence remasking strategy, several works have introduced novel sampling strategies to dynamically adjust the number of tokens predicted in parallel, thereby improving inference efficiency. Fast-dLLM-parallel (Wu et al., 2025) adopts a straightforward approach by selecting tokens with confidence scores exceeding a predefined threshold. Meanwhile, Ben-Hamu et al. (2025) propose an entropy-bounded (EB) sampler, a drop-in replacement for conventional samplers that leverages an entropy-based unmasking procedure to dynamically decode multiple tokens per step while maintaining a predefined error tolerance. Although our WINO brings the acceleration promise due to sampling compression, different from these works, we explore to address the inherent limitation of standard decoding in DLLMs.

### 3 Preliminary: Decoding Process for DLLMs

Given a prompt  $X$ , a DLLM is designed to generate a response  $Y = [y_1, y_2, \dots, y_L]$  with a predefined response length  $L$ . The response sequence is initialized as all special mask tokens,  $Y^{(0)} = [\text{[MASK]}, \text{[MASK]}, \dots, \text{[MASK]}]$ . The decoding process iteratively refines the response sequence  $Y^{(k)}$  over a total of  $K$  denoising steps. In the following, we detail the case of  $K = L$  (*i.e.*, decoding one token per step), as existing models typically achieve optimal performance under this setting (Nie et al., 2025).

At step  $k$ , the goal of decoding is to refine the sequence  $Y^{(k-1)}$  into  $Y^{(k)}$ . Given the token vocabulary  $V$  and the model parameterized with  $\theta$ , the model estimates the probability distribution over the response sequence as  $p_\theta(\hat{Y}|X, Y^{(k-1)})$ . As a common example, in high-confidence greedy decoding,  $Y^{(k)}$  is obtained by unmasking the most confident  $\text{[MASK]}$  token based on  $Y^{(k-1)}$ , *i.e.*,

$$\begin{aligned} l^{(k)} &= \arg \max_{l \in \{l | y_l^{(k-1)} = \text{[MASK]}\}} \left( \max_{v \in V} p_\theta(\hat{y}_l = v | X, Y^{(k-1)}) \right), \\ y_l^{(k)} &= \begin{cases} \arg \max_{v \in V} p_\theta(\hat{y}_l = v | X, Y^{(k-1)}), & \text{if } l = l^{(k)}, \\ y_l^{(k-1)}, & \text{otherwise,} \end{cases} \quad \forall l \in \{1, 2, \dots, L\}. \end{aligned} \quad (1)$$

After completing all  $K$  decoding steps, the final generated response is  $Y = Y^{(K)}$ . Existing DLLMs, such as LLaDA (Nie et al., 2025) and MMaDA (Yang et al., 2025), can also accelerate the decoding process via naive parallel sampling by generating multiple tokens (*e.g.*, 2 or 4) per step. However, empirical results reveal that such strategies often result in substantial performance degradation, limiting their practical effectiveness despite the computational speedup.

**Semi-Autoregressive Diffusion Decoding.** This strategy is widely adopted by DLLMs like LLaDA (Nie et al., 2025) and MMaDA (Yang et al., 2025), which involves splitting the response sequence into multiple blocks and decoding them sequentially from left to right. Within each block, the typical diffusion decoding strategy described above is applied.

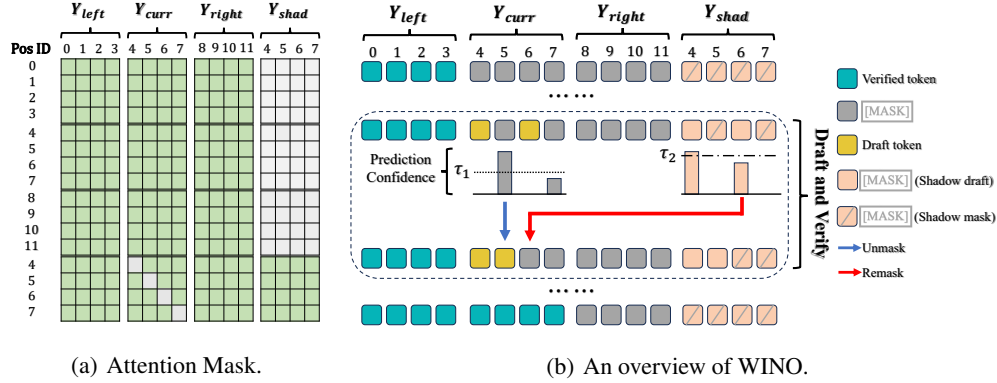


Figure 2: (a) Illustration of our designed attention mask. The green squares denote 1, the grey squares denote 0, and “Pos ID” is short for position ID. (b) An overview of WINO. Verified tokens refer to tokens in the prompt  $X$  or previously decoded blocks. Draft tokens denote tokens in the current block that are unmasked up to the current decoding step. [MASK] (shadow draft) refer to tokens in the shadow block whose position IDs correspond to the draft tokens while [MASK] (shadow mask) refer to the remaining tokens in the shadow block.

## 4 Method

### 4.1 Key Limitation of decoding process for DLLMs

While architecturally suited for parallelism, DLLMs face a critical bottleneck that hinders effective multi-token decoding. During the early generation stages, the sparse context often causes the model to produce low-quality or contradictory tokens when attempting to decode in parallel. This issue is severely magnified by the irreversibility nature of the standard decoding process. As these flawed initial predictions are permanently locked in, they inevitably propagate and degrade the final generation quality in the progressive diffusion steps, forcing an undesirable trade-off between the speed of parallel decoding and the quality of serial decoding.

To improve the trade-off, one critical point is to abandon the assumption of irreversibility in the standard decoding of DLLMs, and build a process of *revokable decoding* for progressive refinement. This principle allows the model to continuously revise its own preliminary, parallel outputs. As more context emerges during generation, the model can correct its initial predictions. Such a mechanism finally resolves the core conflict. It marries the efficiency of parallel generation with the precision of context-aware correction, unlocking the true potential of DLLMs.

### 4.2 Iterative Refinement via Parallel Draft-and-Verify

Motivated by the above analysis and the design intuition, we propose a parallel Draft-and-Verify framework to enable revokable decoding for more efficient and higher-quality generation in DLLMs.

Specifically, our framework performs two modules in parallel at each decoding step: 1) Draft: aggressively unmask multiple [MASK] tokens into candidate meaningful tokens; 2) Verify: evaluates all currently unmasked tokens and re-masks those deemed low-quality for further refinement. We adopt the most common and general semi-autoregressive decoding paradigm to present our method. When the block length equals the generation length, it becomes equivalent to full diffusion decoding.

#### 4.2.1 Drafting

We denote the entire sequence as  $Y = [Y_{left}, Y_{cur}, Y_{right}]$ , where  $Y_{left}$  contains the prompt  $X$  and the previously decoded blocks,  $Y_{cur} = [y_{cur,1}, \dots, y_{cur,L_b}]$  represents the current block being decoded, and  $Y_{right}$  denotes the remaining blocks to be decoded. Here,  $L_b$  is the block length. At the  $k$ -th decoding step, instead of decoding a fixed number of tokens, we perform aggressive multi-token

parallel decoding based on a confidence threshold  $\tau_1$ :

$$y_{\text{cur},l}^{(k)} = \arg \max_{v \in V} p_\theta(\hat{y}_{\text{cur},l} = v | Y), \text{ if } \max_{v \in V} p_\theta(\hat{y}_{\text{cur},l} = v | Y) > \tau_1 \text{ and } y_{\text{cur},l}^{(k-1)} = \boxed{\text{[MASK]}}. \quad (2)$$

Here, a relatively low confidence threshold  $\tau_1$  is adopted to allow more possible tokens to be decoded at each step, which will achieve the acceleration if only a few tokens among them are revoked during the verification module detailed in the next section. This will be demonstrated in the experiments.

#### 4.2.2 Verification

The design principle of the verification module is to utilize the increasingly enriched semantic context at each decoding step—relative to earlier steps, to evaluate the quality of previously unmasked tokens. By re-masking low-quality tokens, the decoding process becomes revokable and amenable for the proper early error correction.

To realize effective quality verification about the decoded tokens, we design an auxiliary shadow block consisting entirely of  $\boxed{\text{[MASK]}}$ ,  $Y_{\text{shad}} = [\boxed{\text{[MASK]}}] \times L_b$ . This block is appended to the sequence  $Y$ , resulting in an extended sequence  $\tilde{Y} = [Y_{\text{left}}, Y_{\text{cur}}, Y_{\text{right}}, Y_{\text{shad}}]$ . We carefully design the position IDs and attention mask associated with  $Y_{\text{shad}}$  to ensure that its output can effectively verify the quality of the tokens decoded at the corresponding positions in  $Y_{\text{cur}}$ .

**Position IDs.** Although  $Y_{\text{shad}}$  is appended to the right end of the sequence, we assign it the same position IDs as  $Y_{\text{cur}}$ . Thus, the output of  $Y_{\text{shad}}$  corresponds to the same positions as  $Y_{\text{cur}}$ , enabling position-wise verification.

**Attention Mask.** As illustrated in Fig. 2(a), we carefully design the attention mask after incorporating  $Y_{\text{shad}}$  into the sequence  $\tilde{Y}$ . Specifically, tokens in  $Y_{\text{left}}$ ,  $Y_{\text{cur}}$ , and  $Y_{\text{right}}$  can freely attend to each other, but they are not allowed to attend to  $Y_{\text{shad}}$ . In contrast, each token in  $Y_{\text{shad}}$  is allowed to attend to all tokens except its corresponding position in  $Y_{\text{cur}}$ .

With the above design of position IDs and attention masks, we achieve the following properties:

- For any token in the current block  $Y_{\text{cur}}$ , appending  $Y_{\text{shad}}$  does not affect the model’s output. Formally,

$$p_\theta(\hat{y}_{\text{cur},l} | Y) = p_\theta(\hat{y}_{\text{cur},l} | \tilde{Y}).$$

- For any token in  $Y_{\text{shad}}$ , the following properties hold. For example, consider the token  $y_{\text{shad},3}$  in Fig. 2(b), which is assigned position ID 6.
  - It shares the same position ID as  $y_{\text{cur},3}$ , and is allowed to attend to  $Y_{\text{left}}$  and  $Y_{\text{right}}$ ;
  - It is explicitly prevented from attending to  $y_{\text{cur},3}$ , effectively avoiding information leakage during verification;
  - For all other positions in  $Y_{\text{cur}}$ , each position is attended by exactly one decoded token (from  $Y_{\text{cur}}$ ) and one  $\boxed{\text{[MASK]}}$  in  $Y_{\text{shad}}$ . The former provides progressively richer contextual semantics during decoding, while the latter serves to regularize the confidence of decoded tokens in  $Y_{\text{cur}}$ , reflecting the uncertainty and the need for potential refinement.

With the specially designed position IDs and the attention mask described above, the verification module can be formally expressed as:

$$y_{\text{cur},l}^{(k)} = \boxed{\text{[MASK]}}, \text{ if } p_\theta(\hat{y}_{\text{shad},l} = y_{\text{cur},l}^{(k-1)} | \tilde{Y}) < \tau_2 \text{ and } y_{\text{cur},l}^{(k-1)} \neq \boxed{\text{[MASK]}}, \quad (3)$$

where  $\tau_2$  is the confidence threshold for verification.

#### 4.2.3 Overall Procedure

In summary, at decoding step  $k$ , our framework enables both the drafting and verification processes to be completed in a single forward pass:

Table 1: Performance and inference speedup comparison on diverse reasoning and generation benchmarks in the language domain.

| Benchmark                      | Method | Accuracy       | Steps            | Step Reduction | TPS             | TPS Speedup |
|--------------------------------|--------|----------------|------------------|----------------|-----------------|-------------|
| GSM8K<br>Math Reasoning        | LLaDA  | 73.24          | 256              | 1.00 ×         | 17.76           | 1.00 ×      |
|                                | WINO   | 75.82 (+2.58)  | 41.93 (-214.07)  | 6.10 ×         | 100.53 (+82.77) | 5.66 ×      |
| MATH<br>Math Reasoning         | LLaDA  | 32.00          | 256              | 1.00 ×         | 17.62           | 1.00 ×      |
|                                | WINO   | 34.20 (+2.20)  | 74.44 (-181.56)  | 3.44 ×         | 55.86 (+38.24)  | 3.17 ×      |
| HumanEval<br>Code Generation   | LLaDA  | 37.80          | 256              | 1.00 ×         | 14.52           | 1.00 ×      |
|                                | WINO   | 42.07 (+4.27)  | 93.32 (-162.68)  | 2.74 ×         | 37.19 (+22.67)  | 2.56 ×      |
| MBPP<br>Code Generation        | LLaDA  | 36.40          | 256              | 1.00 ×         | 18.52           | 1.00 ×      |
|                                | WINO   | 36.40 (+0.00)  | 96.57 (-159.43)  | 2.65 ×         | 45.39 (+26.87)  | 2.45 ×      |
| Countdown<br>Logical Reasoning | LLaDA  | 24.21          | 256              | 1.00 ×         | 17.22           | 1.00 ×      |
|                                | WINO   | 33.20 (+8.99)  | 105.88 (-150.12) | 2.41 ×         | 38.97 (+21.75)  | 2.26 ×      |
| Sudoku<br>Logical Reasoning    | LLaDA  | 14.23          | 256              | 1.00 ×         | 11.61           | 1.00 ×      |
|                                | WINO   | 15.20 (+0.97)  | 131.96 (-124.04) | 1.94 ×         | 21.11 (+9.50)   | 1.82 ×      |
| ARC-E<br>Commonsense Reasoning | LLaDA  | 59.13          | 256              | 1.00 ×         | 17.26           | 1.00 ×      |
|                                | WINO   | 81.19 (+22.06) | 40.19 (-215.81)  | 6.37 ×         | 101.61 (+84.35) | 5.89 ×      |
| ARC-C<br>Commonsense Reasoning | LLaDA  | 51.87          | 256              | 1.00 ×         | 17.10           | 1.00 ×      |
|                                | WINO   | 73.89 (+22.02) | 47.41 (-208.59)  | 5.40 ×         | 85.42 (+68.32)  | 5.00 ×      |

$$y_{\text{cur},l}^{(k)} = \begin{cases} \arg \max_{v \in V} p_{\theta}(\hat{y}_{\text{cur},l} = v | \tilde{Y}), & \text{if } \max_{v \in V} p_{\theta}(\hat{y}_{\text{cur},l} = v | \tilde{Y}) > \tau_1 \text{ and } y_{\text{cur},l}^{(k-1)} = [\text{MASK}], \\ [\text{MASK}], & \text{if } p_{\theta}(\hat{y}_{\text{shard},l} = y_{\text{cur},l}^{(k-1)} | \tilde{Y}) < \tau_2 \text{ and } y_{\text{cur},l}^{(k-1)} \neq [\text{MASK}], \\ y_{\text{cur},l}^{(k-1)}, & \text{otherwise.} \end{cases} \quad (4)$$

We iteratively refine the entire  $Y_{\text{cur}}$  using the procedure in Eq. (4), until all tokens in  $Y_{\text{cur}}$  are no longer  $[\text{MASK}]$ . We set the drafting threshold  $\tau_1$  and the verification threshold  $\tau_2$  such that  $\tau_1 < \tau_2$ . A lower  $\tau_1$  accelerates the decoding process by allowing more tokens to be generated in parallel, while a higher  $\tau_2$  ensures the quality of the final output by enforcing stricter acceptance criteria. We refer to this design philosophy as "Wide-In, Narrow-Out" and term our method as WINO in short.

## 5 Experiment

### 5.1 Experiment setup

**Datasets and Baselines.** We conduct experiment to evaluate WINO across different types of tasks and domains. Specifically, for language domain, we compare WINO with the standard decoding of LLaDA on eight tasks: GSM8K (Cobbe et al., 2021), MATH (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), covering four categories of textual generation tasks—math reasoning, code generation, logical reasoning, and commonsense reasoning. For vision-language domain, we evaluate WINO against the standard decoding of MMaDA (Yang et al., 2025) on six multimodal understanding tasks: Flickr30k (Young et al., 2014), AI2D (Kembhavi et al., 2016), Math-Vision (Wang et al., 2024), MathVista (Lu et al., 2024), MMMU (Yue et al., 2024) and ScienceQA (Lu et al., 2022), spanning four types of multimodal tasks—captioning, chart understanding, math reasoning and multi-discipline reasoning. For clarity, we test on the validation set of MMMU and the official testmini subset of MathVista.

**Evaluation Details.** All benchmarks are evaluated in a zero-shot manner. Regarding the performance metrics of the considered benchmarks, we use the CIDEr metric (Vedantam et al., 2015) for the Flickr30K benchmark and accuracy for all the remaining benchmarks. To assess the inference efficiency of our method, we measure the required decoding steps and Tokens Per Second (TPS) of the baselines and WINO on every task by averaging over all the samples in a benchmark. TPS represents the average number of tokens generated by the model per second. We define the step reduction



Table 2: Performance and inference speedup comparison across diverse generation and reasoning benchmarks in the vision-language domain. We use CIDEr for Flickr30K and accuracy for other benchmarks as the performance metrics.

| Benchmark                               | Method | Metric         | Steps           | Step Reduction | TPS            | TPS Speedup |
|---|--------|----------------|-----------------|----------------|----------------|-------------|
| Flickr30K<br>Captioning                 | MMaDA  | 53.67          | 256             | 1.00 ×         | 6.41           | 1.00 ×      |
|   | WINO   | 53.83 (+0.16)  | 25.47 (-230.53) | 10.05 ×        | 55.11 (+48.70) | 8.60 ×      |
| AI2D<br>Chart Understanding             | MMaDA  | 54.86          | 256             | 1.00 ×         | 6.31           | 1.00 ×      |
|   | WINO   | 57.19 (+2.33)  | 30.90 (-225.10) | 8.30 ×         | 46.04 (+39.73) | 7.30 ×      |
| Math-Vision<br>Math Reasoning           | MMaDA  | 8.55           | 256             | 1.00 ×         | 6.22           | 1.00 ×      |
|   | WINO   | 9.57 (+1.02)   | 44.69 (-211.31) | 5.73 ×         | 31.17 (+24.95) | 5.01 ×      |
| MathVista-mini<br>Math Reasoning        | MMaDA  | 31.10          | 256             | 1.00 ×         | 6.21           | 1.00 ×      |
|   | WINO   | 31.40 (+0.30)  | 33.45 (-222.55) | 7.65 ×         | 41.96 (+35.75) | 6.76 ×      |
| MMMU-val<br>Multi-discipline Reasoning  | MMaDA  | 18.56          | 256             | 1.00 ×         | 6.02           | 1.00 ×      |
|   | WINO   | 24.00 (+5.44)  | 38.47 (-217.53) | 6.65 ×         | 36.13 (+30.11) | 6.00 ×      |
| ScienceQA<br>Multi-discipline Reasoning | MMaDA  | 30.89          | 256             | 1.00 ×         | 6.07           | 1.00 ×      |
|   | WINO   | 42.24 (+11.35) | 28.12 (-227.88) | 9.10 ×         | 49.45 (+43.38) | 8.15 ×      |

Table 3: Experiment results on different generation length with block length  $L_b$  equal to 128. The baselines unmask 1 tokens per decoding step.

| Benchmark | Generation Length | Method | Accuracy | Steps | Step Reduction | TPS    | TPS Speedup |
|-----------|-------------------|--------|----------|-------|----------------|--------|-------------|
| GSM8K     | 256               | LLaDA  | 73.24    | 256   | 1.00 ×         | 17.76  | 1.00 ×      |
|           |                   | WINO   | 75.82    | 41.93 | 6.10 ×         | 100.53 | 5.66 ×      |
|           | 512               | LLaDA  | 74.60    | 512   | 1.00 ×         | 11.84  | 1.00 ×      |
|           |                   | WINO   | 79.91    | 68.53 | 7.47 ×         | 82.64  | 6.98 ×      |
| MMMU-val  | 256               | MMaDA  | 18.56    | 256   | 1.00 ×         | 6.02   | 1.00 ×      |
|           |                   | WINO   | 24.00    | 38.47 | 6.65 ×         | 36.13  | 6.00 ×      |
|           | 512               | MMaDA  | 18.44    | 512   | 1.00 ×         | 5.01   | 1.00 ×      |
|           |                   | WINO   | 23.44    | 64.82 | 7.90 ×         | 35.01  | 6.99 ×      |

as the ratio of the required decoding steps of the baseline to that of WINO and TPS speedup as the ratio of the TPS of the baseline to that of WINO.

**Implementation details.** For the model configuration, we adopt the open-sourced LLaDA-8B-Instruct<sup>2</sup> for language domain tasks and MMaDA-8B-MixCoT<sup>3</sup> for vision-language domain tasks. We employ the semi-autoregressive sampling strategy introduced in LLaDA (Nie et al., 2025), where the output sequence is partitioned into multiple blocks and generated from left to right. In our evaluation, we set the generation length to 256, the decoding steps of baseline models to 256, and the block length to 128, unless specified otherwise. For output sequences that contain reasoning traces, we extract the final answer for evaluation. For the hyperparameters of WINO, we set the verification threshold  $\tau_2$  to 0.9 and tune the drafting threshold  $\tau_1$  from a range of {0.4, 0.5, 0.6, 0.7}.

## 5.2 Main Results

**Performance and speedup on text generation.** We report the performance, decoding step number and throughput (TPS) of LLaDA-8B-Instruct, with and without WINO, on language domain tasks in Tab. 1. WINO achieves significantly better accuracy on all textual reasoning and generation benchmarks with far fewer decoding steps than the baseline LLaDA, except for the MBPP task, where WINO achieves the same performance as LLaDA. For instance, WINO improves accuracy on GSM8K by 2.58% with 6.10× step reduction and 5.66× TPS speedup. These experimental results demonstrate the effectiveness of WINO in enhancing generation quality and inference efficiency.

**Performance and speedup on multimodal understanding and reasoning.** We assess the performance and efficiency gain of WINO incorporated into MMaDA and summarize the results in Tab. 2. Compared to the vanilla MMaDA, WINO demonstrates consistent and substantial improvements in inference efficiency across all benchmarks. Notably, the speedup effect is even more pronounced

<sup>2</sup><https://huggingface.co/GSAI-ML/LLaDA-8B-Instruct>

<sup>3</sup><https://huggingface.co/Gen-Verse/MMaDA-8B-MixCoT>

Table 4: Experiment results on full diffusion setting with block length  $L_b$  equal to generation length. The baselines unmask 1 tokens per decoding step.

| Benchmark | Generation Length | Method | Accuracy       | Steps | Step Reduction | TPS    | TPS Speedup |
|-----------|-------------------|--------|----------------|-------|----------------|--------|-------------|
| GSM8K     | 256               | LLaDA  | 34.34          | 256   | 1.00 ×         | 17.73  | 1.00 ×      |
|           |                   | WINO   | 58.22 (+23.88) | 38.77 | 6.60 ×         | 93.61  | 5.28 ×      |
|           | 128               | LLaDA  | 58.60          | 128   | 1.00 ×         | 23.23  | 1.00 ×      |
|           |                   | WINO   | 62.32 (+3.72)  | 23.95 | 5.34 ×         | 114.29 | 4.92 ×      |
| MMM-Val   | 256               | MMaDA  | 17.22          | 256   | 1.00 ×         | 6.11   | 1.00 ×      |
|           |                   | WINO   | 22.44 (+5.22)  | 24.94 | 10.26 ×        | 50.03  | 8.19 ×      |
|           | 128               | MMaDA  | 15.33          | 128   | 1.00 ×         | 6.70   | 1.00 ×      |
|           |                   | WINO   | 23.11 (+7.78)  | 19.14 | 6.69 ×         | 39.94  | 5.96 ×      |

than that on textual domain tasks, when compared with results in Tab. 1. Furthermore, WINO greatly improves the task performance of MMaDA on AI2D, MMMU and ScienceQA while maintaining comparable results on Flickr30K, Math-Vision and MathVista. These results indicate that WINO consistently delivers both performance gains and accelerated inference in the multimodal domain.

**Relation between speedup and task complexity.** As shown in Tab. 1 and Tab. 2, we observe a consistent positive correlation between the degree of speedup and task performance across all benchmarks. Notably, both step reduction and TPS speedup values are significantly higher for benchmarks when the baseline model achieves higher task scores (than those of more complex tasks). For instance, WINO achieves a  $10.05\times$  step reduction on the relatively simple captioning task Flickr30K, compared to only  $5.73\times$  step reduction on the more challenging math reasoning benchmark Math-Vision. This phenomenon implicitly indicates that if the base model performs better on the target tasks, the gain of WINO could become more significant. Constrained by the fixed decoding steps and fixed generated tokens per step, the baseline model shows inference inefficiency, particularly for simpler reasoning and generation tasks. To further investigate this phenomenon, we evaluate the decoding steps of WINO across subsets of the MATH benchmark categorized by difficulty levels. As illustrated in Fig. 3, WINO achieves progressively greater acceleration as the difficulty level decreases, highlighting its capability to adaptively optimize inference speed based on task complexity.

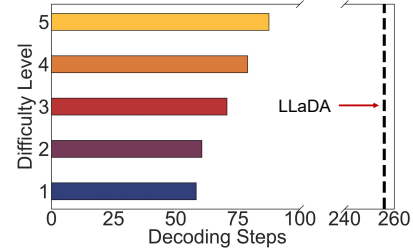


Figure 3: Decoding steps of WINO on subsets of the MATH benchmark with varied difficulty levels.

### 5.3 Ablation Study and Further Analysis.

**On different generation length.** In Tab. 3, we evaluate the performance of WINO with experiments on different generation lengths, where the block length  $L_b$  is fixed to 128 and the baselines unmask 1 tokens every decoding step (to achieve its best generation performance). When the generation length is set to 512, WINO still achieves comparable or better task performance with significantly fewer decoding steps, demonstrating the effectiveness of WINO across different generation lengths.

**On full diffusion decoding (instead of semi-autoregressive decoding).** In Tab. 4, we compare the baselines and WINO applying full diffusion decoding, which means the block length  $L_b$  is set equal to the generation length. Compared to results on the semi-autoregressive decoding in Tab. 1 and Tab. 2, WINO demonstrates substantially strong accuracy gains under the full diffusion setting. Notably, while LLaDA suffers a substantial accuracy drop on GSM8K with full diffusion decoding, WINO maintains reasonable performance with far fewer decoding steps. These results indicate that WINO unlocks significantly greater potential for boosting model performance and computational efficiency when applied in full diffusion decoding scenarios.

**Ablation on verification module.** We conduct an ablation study on a variant of WINO that excludes the verification module, implemented by setting the verification threshold  $\tau_2$  to zero. As presented in Tab. 5, this variant exhibits significant performance degradation across both benchmarks compared to WINO. Specifically, when the drafting threshold  $\tau_1$  is small (corresponding to 0.6 in the table), more candidate tokens are unmasked per decoding step, which naturally introduces a higher proportion of unreliable tokens and ultimately compromises output quality. Conversely, when  $\tau_1$  is large



Table 5: Experiment results on the variant of WINO without the verification module.

| Benchmark | Method                        | Accuracy | Steps | Step Reduction | TPS    | TPS Speedup |
|-----------|-------------------------------|----------|-------|----------------|--------|-------------|
| GSM8K     | LLaDA                         | 73.24    | 256   | 1.00 ×         | 17.76  | 1.00 ×      |
|           | Only Draft ( $\tau_1 = 0.6$ ) | 70.28    | 34.79 | 7.36 ×         | 130.89 | 7.37 ×      |
|           | Only Draft ( $\tau_1 = 0.9$ ) | 72.33    | 81.39 | 3.15 ×         | 56.12  | 3.16 ×      |
|           | WINO                          | 75.82    | 41.93 | 6.10 ×         | 100.53 | 5.66 ×      |
| MMM-Val   | MMaDA                         | 18.56    | 256   | 1.00 ×         | 6.02   | 1.00 ×      |
|           | Only Draft ( $\tau_1 = 0.6$ ) | 19.89    | 35.63 | 7.18 ×         | 43.22  | 7.18 ×      |
|           | Only Draft ( $\tau_1 = 0.9$ ) | 18.56    | 79.74 | 3.21 ×         | 19.38  | 3.22 ×      |
|           | WINO                          | 24.00    | 38.47 | 6.65 ×         | 36.13  | 6.00 ×      |

(corresponding to 0.9 in the table), fewer candidate tokens are unmasked per decoding step, which can mitigate error propagation but at the expense of computational efficiency. Crucially, without the verification module, the generation process lacks a mechanism to correct erroneous predictions. As a result, even with a large  $\tau_1$ , the model may fail to achieve comparable performance, underscoring the necessity of the verification module in maintaining both generation quality.

**Comparison with naive parallel sampling.** The decoding process of existing DLLMs can be sped up by generating multiple tokens per step, *i.e.*, naive parallel sampling. However, as presented in Tab. 6, directly increasing the fixed number of generated tokens per step for DLLMs leads to significant performance degradation. For instance, accuracy on GSM8K drops from 73.24% (1 token/step) to 64.67% (4 tokens/step). In contrast, the draft-and-verify procedure of WINO allows flexible decoding during the generation process, which substantially improves task performance while achieving greater inference speedup.

Table 6: Experiment results on baselines with naive parallel sampling.

| Benchmark      | Method | Steps | Accuracy | Step Reduction |
|----------------|--------|-------|----------|----------------|
| GSM8K          | LLaDA  | 256   | 73.24    | 1.00 ×         |
|                |        | 128   | 71.11    | 2.00 ×         |
|                |        | 64    | 64.67    | 4.00 ×         |
|                | WINO   | 41.93 | 75.82    | 6.10 ×         |
| MathVista-mini | MMaDA  | 256   | 31.10    | 1.00 ×         |
|                |        | 128   | 30.02    | 2.00 ×         |
|                |        | 64    | 29.80    | 4.00 ×         |
|                | WINO   | 33.45 | 31.40    | 7.65 ×         |

**GPU memory usage.** To facilitate efficient and effective quality verification of the unmasked tokens, WINO introduces an auxiliary shadow block whose size equals the specified block length  $L_b$  of the whole semi-autoregressive decoding process. Therefore, WINO induces additional GPU memory cost to store the auxiliary shadow block. We evaluate the GPU memory cost of WINO and present the results in Fig. 4. The additional memory cost of WINO remains marginal compared to the baselines across all the benchmarks. For instance, on GSM8K, WINO increases GPU memory usage by only 2.4% (from 16.18 GB to 16.57 GB) compared to standard LLaMA decoding. Across other datasets, the additional memory overhead introduced by WINO is similarly negligible.

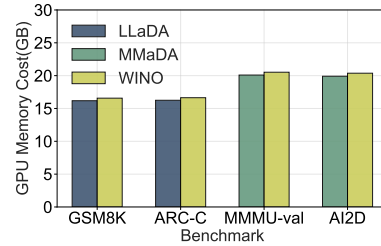


Figure 4: GPU memory usage.

**Effect of threshold tuning.** In Fig. 5, we present the evaluation results of WINO with varying drafting threshold  $\tau_1$ . Our experiments suggest that WINO consistently outperforms baselines across different benchmarks and the  $\tau_1$  values in terms of both task performance and inference efficiency. As the  $\tau_1$  value decreases, more candidate tokens are unmasked at each decoding step, thereby accelerating inference by reducing the required decoding steps. However, this comes at the cost of introducing more unreliable predictions, which may place a greater burden on the verification module to correct errors. Empirically, we find that setting the value of drafting threshold  $\tau_1$  within the range of 0.5 to 0.7 achieves an optimal balance, maintaining competitive task performance while preserving efficient generation.

**Case Study: Decoding Dynamics.** To conduct a fine-grained examination of the decoding processes of the baselines and WINO, we present two generation examples—one from GSM8K and one from MMM-Val below. As shown, the baseline may produce erroneous tokens at the early decoding stages. Since the generated tokens by the baseline remain unchanged in subsequent decoding steps, the false contextual information propagates throughout the whole generation process, eventu-

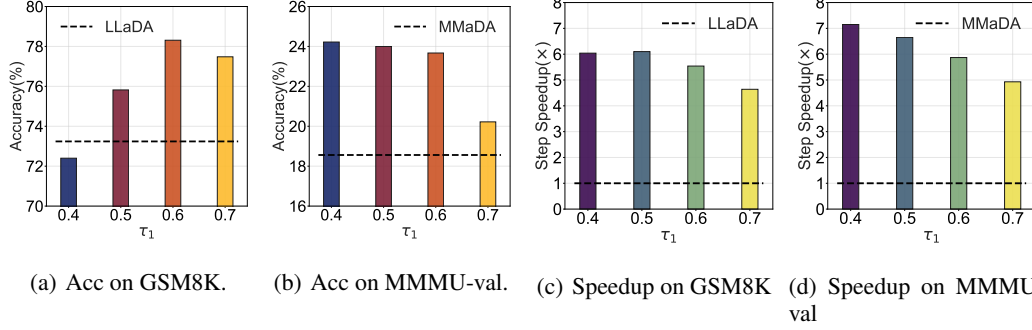


Figure 5: Experiments on the drafting threshold  $\tau_1$  with the verification threshold  $\tau_2$  fixed to 0.9.

ally leading to low-quality generation results. In contrast, WINO enables dynamical refinement of generated tokens via an iterative draft-and-verify mechanism, which mitigates error accumulation and facilitates high-quality decoded outputs.

Case: GSM8K, Generation Length 256, Block Length 128, 2 blocks in total

**Question:** In a family, there are 2 brothers and 3 sisters. All sisters are the same age, which is 16. One of the brothers is 12 years old, which is half the age of the older brother. What is the total age of all these siblings?

**GT:** 84.

**LLaDA:**

Step: 128 (1st block decoding completed). 1st block:

<reasoning>

1. We know that all three sisters are 16 years old.
2. One of the brothers is 12 years old, which is half the age of the older brother.
3. Let's denote the age of the older brother as  $x$ .
4. According to the problem, 12 is half the age of the older brother, so we can write the equation:  $12 = \frac{x}{2}$ .
5. Solving for  $x$ , we multiply both sides by 2:  $x = 24$ .
6. Therefore, the

Step: 162 (early overconfident error). 2nd block:

older brother is 24 years old.  
 7.<|mdm\_mask|>.....<|mdm\_mask|>  
 </reasoning>  
 <answer>  
 96  
 </answer><|eot\_id|><|endoftext|>

Final (error accumulation):

<reasoning>

1. We know that all three sisters are 16 years old.
2. One of the brothers is 12 years old, which is half the age of the older brother.
3. Let's denote the age of the older brother as  $x$ .
4. According to the problem, 12 is half the age of the older brother, so we can write the equation:  $12 = \frac{x}{2}$ .
5. Solving for  $x$ , we multiply both sides by 2:  $x = 24$ .
6. Therefore, the older brother is 24 years old.
7. The younger brother, being the only other brother, is also 24 years old.
8. The total age of the two brothers is  $24 + 24 = 48$ .
9. The total age of the three sisters is  $3 \times 16 = 48$ .
10. The total age of all the siblings is  $48 + 48 = 96$ .

</reasoning>

<answer>

96  
 </answer><|eot\_id|><|endoftext|>

**WINO:**

Step: 35 (1st block decoding completed). 1st block:

<reasoning>

1. We know that all three sisters are 16 years old.
2. One of the brothers is 12 years old, which is half the age of the older brother.

3. Let's denote the age of the older brother as  $x$ .
4. According to the problem, 12 is half the age of the older brother, so we can write the equation:  $12 = \frac{x}{2}$ .
5. Solving for  $x$ , we multiply both sides by 2:  $x = 24$ .
6. Therefore, the

Step: 37 (draft error). 2nd block:

older brother is 24 years old.

7. <|mdm\_mask|>.....<|mdm\_mask|> = 96<|mdm\_mask|><|mdm\_mask|><|mdm\_mask|>

</reasoning<|mdm\_mask|>

<answer>

9<|mdm\_mask|>

</answer><|eot\_id|><|endoftext|>

Step: 42 (refinement: still wrong). 2nd block:

older brother is 24 years old.

7. The total age of<|mdm\_mask|> sisters<|mdm\_mask|> \(\ 3<|mdm\_mask|>\text{times } 16

$= 4<|mdm_mask|>.....<|mdm_mask|>$  total age of all the<|mdm\_mask|> is \(\ 48 +

<|mdm\_mask|><|mdm\_mask|> = 86 \) years.

</reasoning>

<answer>

86

</answer><|eot\_id|><|endoftext|>

Final (all correct through iterative refinement):

<reasoning>

1. We know that all three sisters are 16 years old.
2. One of the brothers is 12 years old, which is half the age of the older brother.
3. Let's denote the age of the older brother as  $x$ .
4. According to the problem, 12 is half the age of the older brother, so we can write the equation:  $12 = \frac{x}{2}$ .
5. Solving for  $x$ , we multiply both sides by 2:  $x = 24$ .
6. Therefore, the older brother is 24 years old.
7. The total age of the sisters is  $3 \times 16 = 48$  years.
8. The total age of the brothers is the sum of the ages of the 12-year-old and the 24-year-old brother:  $12 + 24 = 36$  years.
9. The total age of all the siblings is  $48 + 36 = 84$  years.

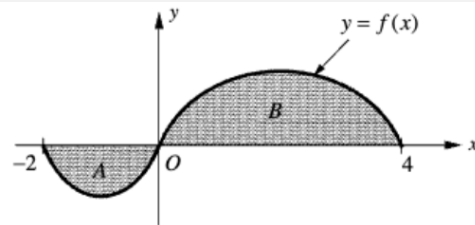
</reasoning>

<answer>

84

</answer><|eot\_id|><|endoftext|>

Case: MMMU-val, Geneation Length 256, Block Length 128, 2 blocks in total



**Question:** The graph of  $y = f(x)$  is shown in the figure above. The shaded region  $A$  has area  $a$  and the shaded region  $B$  has area  $b$ . If  $g(x) = f(x) + 3$ , what is the average value of  $g$  on the interval  $[-2, 4]$ ?

- A.  $(a + b + 3)/6$
- B.  $(-a + b + 3)/6$
- C.  $(-a + b)/6 + 3$
- D.  $(a + b)/6 + 3$

GT: C.

MMaDA:

Step: 70 (early overconfident error). 1st block:

<think>

```

1. To find the average value of  $g(x)$  on the interval  $[-2, 4]$ , we need to calculate the area of the
graph of  $g(x)$  over this interval.
2. The area of the graph of  $g(x)$  is given by the sum of the areas of the two shaded regions.
<|mdm_mask|>.....<|mdm_mask|>

```

Step: 128 (**error accumulation**). 1st block:

```

<think>
1. To find the average value of  $g(x)$  on the interval  $[-2, 4]$ , we need to calculate the area of the
graph of  $g(x)$  over this interval.
2. The area of the graph of  $g(x)$  is given by the sum of the areas of the two shaded regions.
3. The area of region  $A$  is  $a$  and the area of region  $B$  is  $b$ .
4. Therefore, the total area of  $g(x)$  is  $a + b$ .
5. The average value of  $g(x)$  the interval

```

Final (**error accumulation**):

```

<think>
1. To find the average value of  $g(x)$  on the interval  $[-2, 4]$ , we need to calculate the area of the
graph of  $g(x)$  over this interval.
2. The area of the graph of  $g(x)$  is given by the sum of the areas of the two shaded regions.
3. The area of region  $A$  is  $a$  and the area of region  $B$  is  $b$ .
4. Therefore, the total area of  $g(x)$  is  $a + b$ .
5. The average value of  $g(x)$  on the interval  $[-2, 4]$  is  $\frac{a+b}{6}$ .
</think>
A<|endoftext|>

```

**WINO:**

Step: 13 (**draft error**). 1st block:

```

<think>
1. To find the average value of  $g(x)$  on the interval  $[-2, 4]$ , we need to use the<|mdm_mask|> for
the<|mdm_mask|> of  $f(x)$  over<|mdm_mask|> interval.
2. The average value of <|mdm_mask|>(x) is given by  $\frac{1}{2} \times$ 
<|mdm_mask|><|mdm_mask|>^4<|mdm_mask|>.....<|mdm_mask|>

```

Step: 33 (**refinement: correct**). 1st block:

```

<think>
1. To find the average value of  $g(x)$  on the interval  $[-2, 4]$ , we need to use the formula for the
average of  $f(x)$  over an interval.
2. The average value of  $f(x)$  is given by  $\frac{1}{6} \cdot \int_{-2}^4 f(x) dx$ .
3. Given that  $g(x) = f(x) + 3$ , we need to find  $f(x)$ . The average value of  $f(x)$  is  $\frac{-a+b}{6}$ .
4. Therefore, <|mdm_mask|>.....<|mdm_mask|>

```

Final (**all correct through iterative refinement**):

```

<think>
1. To find the average value of  $g(x)$  on the interval  $[-2, 4]$ , we need to use the formula for the
average of  $f(x)$  over an interval.
2. The average value of  $f(x)$  is given by  $\frac{1}{6} \cdot \int_{-2}^4 f(x) dx$ .
3. Given that  $g(x) = f(x) + 3$ , we need to find  $f(x)$ . The average value of  $f(x)$  is  $\frac{-a+b}{6}$ .
4. Therefore, the average value of  $g(x)$  on the interval  $[-2, 4]$  is  $\frac{-a+b}{6} + 3$ .
5. The correct answer is option C.
</think>
C<|endoftext|>

```

## 6 Conclusion

In this work, we introduce Wide-In, Narrow-Out (WINO), a training-free decoding algorithm that resolves the critical quality-speed trade-off in Diffusion Large Language Models (DLLMs) by making their generation process revokable. WINO overcomes the limitations of irrevokable standard decoding by employing a parallel draft-and-verify mechanism, allowing the model to aggressively generate tokens and iteratively correct errors using its full bidirectional context. Our experiments on existing open-source models like LLaDA and MMaDA demonstrate that WINO simultaneously accelerates inference by up to  $10\times$  while significantly improving accuracy across a diverse set of language and vision-language tasks. While acknowledging areas for future architectural improvements, WINO fundamentally enhances the practicality of DLLMs by rethinking the decoding process itself, establishing them as a truly efficient and high-quality alternative to autoregressive systems.

## References

- Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- 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, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 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.
- 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, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 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.
- 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.
- Google DeepMind. Gemini diffusion. Google DeepMind Models, 2025. URL <https://deepmind.google/models/gemini-diffusion>.
- 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, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- Inception Labs. Introducing mercury: The first commercial diffusion-based language model. Inception Labs Blog, 2025. URL <https://www.inceptionlabs.ai/introducing-mercury>.
- 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.

- Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. d1lm-cache: Accelerating diffusion large language models with adaptive caching. *CoRR*, abs/2506.06295, 2025.
- 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.
- Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo, and Shenghua Liu. A survey of context engineering for large language models, 2025. URL <https://arxiv.org/abs/2507.13334>.
- 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, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 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. Chatgpt: Optimizing language models for dialogue. OpenAI Blog, November 2022. URL <https://openai.com/blog/chatgpt/>.
- 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 *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022.
- Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 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, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.
- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.
- Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015.
- Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. GPT-4 doesn’t know it’s wrong: An analysis of iterative prompting for reasoning problems. *CoRR*, abs/2310.12397, 2023.
- Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. Can large language models really improve by self-critiquing their own plans? *CoRR*, abs/2310.08118, 2023.
- Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575, 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.
- 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.
- Ling Yang, Ye Tian, Bowen Li, Xincheng 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, 2025. URL <https://hkunlp.github.io/blog/2025/dream>.
- 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, Ruqi 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 Wenhui Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, pages 9556–9567. IEEE, 2024.
- 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.