BEYOND THE CANVAS: EFFICIENT DLLMS VIA SELF-GUIDED COT COMPRESSION AND SUFFIX SPARSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) have emerged as a promising alternative to autoregressive generation. Existing acceleration methods mainly improve per-step decoding parallelism, but largely overlook redundancy at the trajectory level. dLLMs iteratively unmask tokens on a fixed-length masked sequence (canvas), which is typically set relatively large to accommodate unpredictable Chain-of-Thought (CoT) generation. We observe that such a design can lead to two key problems that affect the efficiency of dLLM reasoning: *(i)* the redundancy in reasoning CoT induced by an oversized canvas; and *(ii)* the redundancy in suffix tokens as context. To alleviate these problems, we propose the Beyond-the-Canvas dLLM (BC-dLLM), a framework with two complementary strategies: **S**elf-**G**uided CoT **C**ompression (SGC) and **P**rogress-**A**ware **S**uffix **S**parsification (PASS). Specifically, the SGC strategy aims to decouple the reasoning length from the canvas size, enabling the model to generate compact CoTs on large canvases. SGC uses the dLLM itself to construct variable-length CoT supervision data and applies conditional supervised fine-tuning to internalize this compact reasoning behavior on large canvases. The design of the PASS strategy is based on our insight that information gain is typically concentrated in suffix tokens adjacent to the current decoding block and diminishes as reasoning progresses. Hence, PASS shrinks the suffix window as reasoning progresses to reduce per-step computation on large canvases. Experimental evaluations against multiple competitive methods on four reasoning benchmarks show that our method significantly reduces decoding steps while maintaining comparable performance to the baselines. When applied to LLaDA-8B-Instruct, our method achieves a $10.1\times$ latency speedup on the GSM8K dataset while maintaining comparable performance.

1 INTRODUCTION

Diffusion large language models (dLLMs) (Nie et al., 2025; Zhu et al., 2025; Ye et al., 2025) have recently emerged as a promising alternative to autoregressive (AR) generation (Yang et al., 2025; Achiam et al., 2023). dLLMs operate on a fixed-length masked sequence, referred to as the *canvas*, and iteratively unmask multiple tokens per step with bidirectional attention. This formulation enables parallel decoding, random-order generation, and global context modeling, which promise substantial inference speedups. However, realizing this potential in practice has proven to be difficult. While closed-source dLLMs such as Mercury (Labs et al., 2025) and Seed Diffusion (Song et al., 2025) report impressive throughput and competitive quality, open-source dLLMs still lag behind and, in many cases, run even slower than their AR counterparts of comparable scale. For example, LLaDA (Nie et al., 2025) still requires decoding steps roughly proportional to the sequence length to maintain quality, largely offsetting its theoretical parallelism advantage. Closing this efficiency gap is therefore crucial for making open-source dLLMs practical alternatives to AR LLMs.

Existing efforts to enhance dLLM inference efficiency primarily center on step-level optimizations, which broadly fall into two complementary directions. One line of work reduces per-step computation. For example, Fast-dLLM (Wu et al., 2025) introduces a block-wise approximate KV-cache, while Sparse-dLLM (Song et al., 2026) and ES-dLLM (Zhu et al., 2026) skip unimportant tokens or layers in the forward pass. Another line of work increases per-step parallelism through improved

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.

schedulers. Representative strategies include confidence-aware unmasking (Wu et al., 2025; Yu et al., 2025), history-aware decoding (Ma et al., 2025b; Kim et al., 2025), and semi-autoregressive decoding (Nie et al., 2025; Wu et al., 2025; Bie et al., 2025; Chen et al., 2025b), which generate multiple tokens in parallel at each step while preserving the left-to-right order across blocks. Despite these improvements, existing efforts largely overlook the inefficiency problem in the overall reasoning trajectory. Common applications of dLLMs require a predefined, oversized canvas to accommodate the variable-length reasoning trajectories that dLLMs can produce. From our initial experiments, we observe two key problems in this design that cause inefficiency: *(i)* **redundancy in reasoning CoT induced by the oversized canvas** (Sec. 3.2), where dLLMs produce unnecessarily verbose reasoning even on simple problems; and *(ii)* **redundancy in suffix tokens as context** (Sec. 3.3), where every decoding step processes the entire remaining canvas although most suffix tokens contribute progressively less information during decoding.

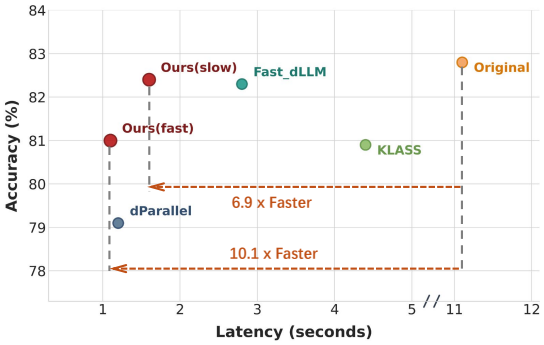


Fig. 1: Performance and speed comparison between our method and other methods (Fast_dLLM (Wu et al., 2025), KLASS (Kim et al., 2025) and dParallel (Chen et al., 2025b)) on GSM8K. With the corresponding conditional prompts, we obtain results in slow- and fast-thinking modes.

To address the above two problems, we propose Beyond-the-Canvas dLLM (BC-dLLM), a framework with two complementary strategies: **S**elf-**G**uided CoT **C**ompression (SGC) and **P**rogress-**A**ware **S**uffix **S**parsification (PASS). SGC tackles the first problem by decoupling the CoT length from the canvas size, enabling dLLMs to generate more compact reasoning paths on oversized canvases. The core idea is to leverage the dLLM itself as an internal compressor to construct variable-length CoT supervision data. During data construction, we first elicit comprehensive reasoning trajectories (*slow-thinking* mode) on a large canvas, and regenerate the later segments of these trajectories on a narrowed canvas, yielding more compact CoTs (*fast-thinking* mode). We then apply supervised fine-tuning (SFT) with conditional prompts on these CoT data to internalize compact reasoning capability on a large canvas. PASS is motivated by the observation that suffix information gain is typically concentrated near the current decoding block and diminishes as decoding progresses. It uses the predicted <EOS> probabilities at masked positions in the current block to estimate decoding progress and dynamically shrinks the suffix window size as decoding progresses, thereby cutting per-step computation on large canvases. By integrating SGC and PASS, BC-dLLM mitigates canvas-induced inefficiency and accelerates reasoning without major performance compromise as shown in Fig. 1. In fast-thinking mode, it achieves a $10.1\times$ latency speedup on GSM8K with LLaDA-8B-Instruct while maintaining competitive performance.

In summary, the contributions of this work are as follows:

- We identify two key bottlenecks in current dLLM acceleration frameworks induced by oversized canvases: (1) redundancy in reasoning CoT due to the unnecessary verbosity for filling a large canvas, and (2) redundancy in suffix tokens as context.
- We propose Beyond-the-Canvas dLLM (BC-dLLM), which integrates two complementary strategies: self-guided CoT compression (SGC) to decouple CoT length from canvas size, and progress-aware suffix sparsification (PASS) to progressively shrink the suffix window during decoding.
- We conduct extensive experiments to validate the effectiveness of BC-dLLM on different reasoning benchmarks and base dLLMs. Our method achieves substantial inference speedups while maintaining reasoning performance comparable to that of standard dLLMs.

2 Related Work

Diffusion Large Language Models. dLLMs (Gao et al., 2024; Nie et al., 2025; Zhu et al., 2025; Ye et al., 2025; Li et al., 2022; He et al., 2023; Dieleman et al., 2022) offer an alternative to autoregressive LLMs (Yang et al., 2025; Achiam et al., 2023; Li et al.; Team et al., 2024; Liu et al.,

2023), using continuous (Dieleman et al., 2022; Li et al., 2022) or discrete methods (He et al., 2023; Nie et al., 2025; Ye et al., 2025). The two families differ in the space where denoising takes place and, consequently, in how their predictions are converted to tokens. Continuous models such as Diffusion-LM (Li et al., 2022) and CDCD (Dieleman et al., 2022) denoise embeddings before projecting them to tokens; the mismatch between embeddings and tokens can destabilize predictions. Discrete models operate directly on categorical tokens: the LLaDA series (Nie et al., 2025; Zhu et al., 2025; You et al., 2025) learns forward masking and reverse unmasking from scratch, while the Dream family (Ye et al., 2025; Xie et al., 2025) adapts pretrained autoregressive LLMs. Hybrid models such as SDAR (Cheng et al., 2025) and Block Diffusion (Arriola et al., 2025) combine autoregressive inter-block and parallel intra-block generation. Our work focuses on discrete dLLMs (Nie et al., 2025; Ye et al., 2025), where masked-token generation makes the canvas size central to inference efficiency.

Chain-of-Thought (CoT) Compression. Alongside the choice of model architecture, the length of a reasoning trajectory is a major source of inference cost. CoT improves complex reasoning in AR LLMs (Wei et al., 2022; Grattafiori et al., 2024; Guo et al., 2025; Huang et al., 2025), but long traces increase latency. Compression methods (Kang et al., 2025; Xia et al., 2025; Li et al., 2025; Luo et al., 2025) use RL with length regularization (Luo et al., 2025; Aggarwal & Welleck, 2025; Li et al., 2025) or SFT on curated shorter trajectories (Xia et al., 2025; Kang et al., 2025; Cui et al., 2025). These approaches pursue the same goal through different forms of supervision: RL generally costs more to train, while SFT depends on high-quality short trajectories. TokenSkip (Xia et al., 2025), for example, prunes tokens with lower semantic contribution. In dLLMs, this problem is further tied to the fixed generation canvas, whose size can encourage longer CoTs. We use the dLLM to generate compact reasoning trajectories, then fine-tune it to produce them even on a large canvas.

Inference Acceleration for dLLMs. Shorter reasoning trajectories do not, by themselves, remove the cost of each diffusion step. dLLM inference is slow because of iterative unmasking (Israel et al., 2025; Zhong et al., 2026; Fu et al., 2025) and limited compatibility with autoregressive KV caching (Shazeer, 2019; Pope et al., 2023). Existing acceleration methods therefore target either the number of decoding steps or the computation performed at each step. Sampling schedulers increase token-level parallelism via confidence-aware unmasking (Fast-dLLM (Wu et al., 2025), Dimple (Yu et al., 2025)) or history-aware decoding (DInfer (Ma et al., 2025b), KLASS (Kim et al., 2025)). Other methods (Wu et al., 2025; Song et al., 2026; Zhu et al., 2026; Ma et al., 2025a; Chen et al., 2025a) reduce per-step computation: Fast-dLLM (Wu et al., 2025) approximates a block-wise KV-cache, while dPad (Chen et al., 2025a) uses a fixed suffix-token sliding window. A fixed suffix window, however, cannot respond to how the contribution of suffix tokens changes across decoding blocks. We use decoding progress to shrink the suffix window as that contribution declines, reducing computation while preserving reasoning quality.

3 METHOD

In this section, we first introduce the preliminaries for the training and the inference process of dLLMs. We then present our proposed Self-Guided CoT Compression (SGC) and Progress-Aware Suffix Sparsification (PASS) strategies in detail.

3.1 PRELIMINARIES

Masked Diffusion Large Language Models. Discrete dLLMs (Austin et al., 2021a) have emerged as a promising non-autoregressive paradigm for natural language generation. As a representative variant, masked dLLMs model the generation as a probabilistic process comprising forward masking corruption and reverse denoising. The forward process gradually corrupts a clean token sequence $\mathbf{x}_0$ of length L into a noisy sequence $\mathbf{x}_t$ at continuous time level $t \in [0, 1]$:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \prod_{i=1}^{L} \Big[(1-t)\, \delta(x_t^i = x_0^i) + t\, \delta(x_t^i = [\texttt{MASK}]) \Big], \tag{1}$$

where x_t^i denotes the token at the i-th position, and $[\texttt{MASK}]$ is a special masking token. The reverse process is parameterized by a neural network p_θ, which attempts to recover the clean sequence $\mathbf{x}_0$

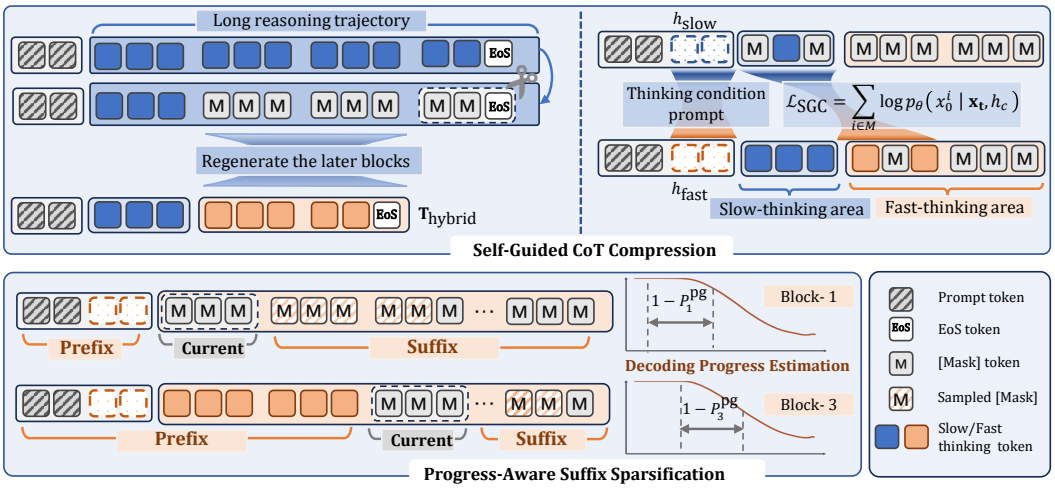


Fig. 2: The overall pipeline of our proposed method. **(Top) Self-Guided CoT Compression (SGC):** We generate a long reasoning trajectory on a large canvas, truncate the later blocks, and regenerate them in a restricted canvas as the fast-thinking blocks. These hybrid-mode trajectories, paired with mode-specific prompts, are then used to fine-tune the dLLM via conditional SFT. **(Bottom) Progress-Aware Suffix Sparsification (PASS):** We adjust the number of sampled suffix tokens over the reasoning process based on decoding progress estimation P_b^{pg}, reducing redundant computation while maintaining essential context.

from the masked sequence $\mathbf{x}_t$ by jointly predicting all masked tokens under the assumption of conditional independence. In recent works (Ou et al., 2024; Austin et al., 2021a), the simplified objective of the model is to minimize the negative log-likelihood restricted only to the masked positions. This objective serves as a surrogate upper bound for the model's true negative log-likelihood:

$$\mathcal{L}(\theta) = -\mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = [\texttt{MASK}]]\log p_\theta(x_0^i \mid \mathbf{x}_t)\right], \tag{2}$$

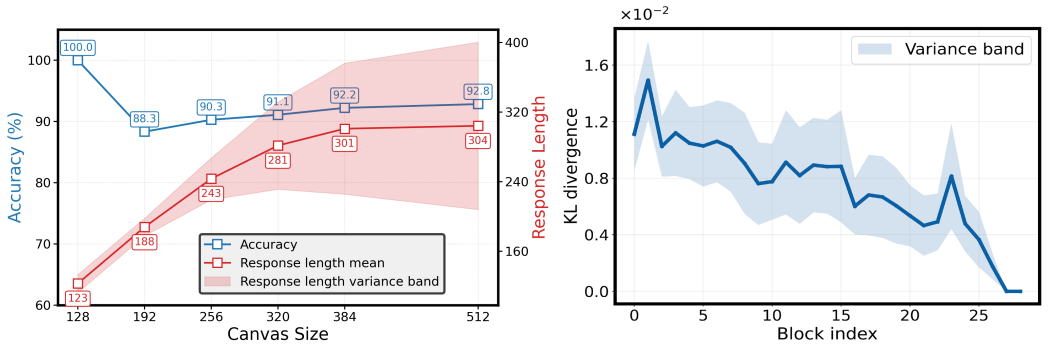
where t is sampled uniformly from $\mathcal{U}(0,1)$, and $\mathbf{1}[\cdot]$ is the indicator function.

Inference of dLLMs. The sampling process of dLLMs begins by initializing a canvas with a fixed-length masked sequence. During generation, unmasked tokens remain unmodified, ensuring that the model focuses exclusively on refining the currently masked positions. Then, at each step, the model predicts distributions for all masked tokens on the canvas in parallel. It samples provisional tokens and applies a dynamic remasking strategy to determine which positions should be permanently unmasked and which should remain masked for further refinement in subsequent steps.

3.2 Self-Guided CoT Compression

Redundancy in Reasoning CoT. As introduced in Sec. 1, the oversized masked sequence often induces redundant reasoning trajectories, especially for simple problems. Here, we conduct a pilot study to further demonstrate this problem, which motivates our proposed Self-Guided CoT Compression (SGC) method. We collect GSM8K (Cobbe et al., 2021) problems that LLaDA-8B already solves correctly at canvas size 128 and measure accuracy and average response length as we enlarge the canvas. As shown in Fig. 3(a), accuracy remains essentially unchanged, yet the average response length grows from 123 to 304 tokens, a nearly 2.5× increase, with a widening variance band. The result demonstrates that the CoT length is tightly entangled with the canvas size, meaning the model tends to take on more redundant reasoning steps as the canvas grows. To decouple the CoT length from the canvas capacity, we propose Self-Guided CoT Compression (SGC). This method utilizes the dLLM itself as an internal compressor to construct high-quality, variable-length CoT data.

Data Construction via Self-Compression. The data construction process consists of two stages associated with different *thinking modes*, as illustrated in the left upper part of Fig. 2. Given an input prompt (often including a question and exemplars), we first initialize a large canvas $\mathbf{C}_{\text{large}}$ to



(a) (b)

Fig. 3: Pilot studies on GSM8K. **(a)** Average response length rises as the canvas size increases, illustrating their entanglement. **(b)** KL divergence between current-block predictions with and without suffix context across decoding blocks.

induce the model into a *slow-thinking mode*. In this setting, we encourage the dLLM to follow a comprehensive reasoning trajectory, including the correct final answer and an <EOS> token to terminate the generation. We denote the long reasoning trajectories as $\mathbf{T}_{\text{slow}}$. To obtain a compact CoT trajectory, we leverage the model's inherent self-compression capability. We randomly truncate the later reasoning blocks in $\mathbf{T}_{\text{slow}}$, and regenerate the remaining steps starting from these intermediate blocks. Crucially, we enforce a *fast-thinking mode* by restricting the available unmasking space to a narrower canvas. With this limited generation horizon, the dLLM acts as its own compressor and attempts to derive the same correct answer in a more compact trajectory, denoted as $\mathbf{T}_{\text{hybrid}}$.

Conditional Supervised Fine-Tuning. To empower the model to generate compact CoTs even on a large canvas, we perform conditional supervised fine-tuning (SFT). Given a training dataset $\mathcal{D}$, we aggregate the trajectories $\mathbf{T}_{\text{hybrid}}$ by self-compression and filter out trajectories with incorrect answers to ensure data quality. We pair the input prompts with their corresponding trajectories $\mathbf{T}_{\text{hybrid}}$ to form the conditional SFT dataset $\mathcal{D}_{\text{SGC}}$. To construct the masked sequence for training, we adopt the semi-autoregressive forward masking strategy as done in previous works (Chen et al., 2025b; Wang et al., 2025). Specifically, a subset of tokens in the block currently being generated and all tokens in subsequent blocks are replaced with [MASK] tokens, while the previous blocks remain unmasked and serve as context. This masking strategy effectively simulates the generation process of dLLMs. We also include distinct conditional prompts in the original instructions to control the model's thinking mode while it decodes the current block. Let h_{slow} and h_{fast} denote the prompt tokens indicating the *slow-thinking* and *fast-thinking* modes, respectively. The training objective leverages the standard negative log-likelihood loss on the masked positions, conditioned on these prompts:

$$\mathcal{L}_{\text{SGC}}(\theta) = -\mathbb{E}_{(\mathbf{x}^c)\sim\mathcal{D}_{\text{SGC}},t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = [\text{MASK}]]\log p_\theta(x_0^i \mid \mathbf{x}_t, h_c)\right], \tag{3}$$

where $c \in \{\text{slow}, \text{fast}\}$, and $\mathbf{x}_0$ corresponds to the reasoning blocks from either the slow- or fast-thinking mode within the trajectory $\mathbf{T}_{\text{hybrid}}$. Through this conditional SFT, the model learns to associate the desired CoT reasoning modes with the corresponding conditional prompts.

During inference, we can explicitly specify h_{fast} to instruct the dLLM to yield concise CoTs even on a large canvas, thereby drastically saving parallel decoding steps.

3.3 Progress-Aware Suffix Sparsification (PASS)

An oversized canvas not only leads to redundant reasoning CoTs but also introduces redundant context computation. With bidirectional attention, the model uses both prefix and suffix context to predict the current block. Unlike unmasked prefix tokens, which provide diverse reasoning context, the suffix tokens are [MASK] tokens with different position encodings, which provide relatively homogeneous context information.

Redundancy in Suffix Tokens as Context. To investigate inefficiency in suffix context utilization, we analyze the attention distribution of the current block across the entire input sequence on the GSM8K dataset. The attention distribution is shown in Appendix C (Fig. 9). While the prefix provides reasoning context, **the suffix tokens closest to the current block receive the most attention within the suffix region**. Furthermore, we analyze the importance of suffix context across different decoding stages by a counterfactual intervention experiment. Specifically, for each decoding block, we quantify the Kullback-Leibler divergence between the model's predicted token distributions for the current block with and without suffix context (Fig. 3(b)). We find that **the context from the suffix tokens shows diminishing gains as decoding progresses.** We hypothesize that, during the initial phase of generation, the decoding process is highly fragile and suffix tokens are relatively important for preserving structural consistency. As unmasking proceeds, the model becomes increasingly confident, and suffix tokens bring only marginal information gain. Inspired by this observation, we believe that progressively adjusting the visible suffix context can reduce redundancy in the later decoding stages while preserving essential structure, thereby improving decoding efficiency.

Dynamic-length Suffix Window. In this work, we propose a progress-aware suffix sparsification strategy that dynamically shrinks the suffix window based on the estimated decoding progress. The key challenge lies in the reliable estimation of the current decoding progress. In practice, language models often terminate generation when predicting a special end-of-sequence token (<EOS>). This indicates that the model has implicitly learned to signal the completion of generation through the <EOS> token. Inspired by this, we use the model's predicted <EOS> probabilities at masked positions to estimate decoding progress. Specifically, let $\mathcal{M}_b$ denote the set of masked token positions in the current decoding block b. We estimate generation progress as one minus the joint predicted probability that all masked positions in the current block are non-<EOS> tokens:

$$P_b^{\text{pg}} = 1 - \prod_{j \in \mathcal{M}_b} \left(1 - p_\theta\left(x_0^j = \texttt{<EOS>} \mid \mathbf{x}_t\right)\right). \tag{4}$$

Based on the decoding progress estimation, we dynamically adjust the suffix window for the corresponding decoding block. The window size is formulated as:

$$D_b = D^{\text{init}} \cdot \sigma(1 - P_b^{\text{pg}}), \tag{5}$$

where D^{init} is the hyper-parameter that bounds the maximum suffix window size, and $\sigma(\cdot)$ is the sigmoid function that maps the progress estimation to a scaling factor. According to the above formulation, during the fragile early stages, the model attends to more suffix tokens and absorbs rich structural cues. As decoding matures, the suffix horizon gradually shrinks towards a narrower window. Our PASS effectively reduces the computation overhead in each decoding step while maintaining the essential reasoning context.

4 Experiments

4.1 Experimental Setup

Baselines and Benchmarks. We evaluate the original dLLM under its official default inference setting (unmasking one token per step). We also compare our approach with recent methods that improve parallel decoding, including Fast-dLLM (Wu et al., 2025), KLASS (Kim et al., 2025), and dParallel (Chen et al., 2025b). We evaluate all methods on four reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), SVMAP (Patel et al., 2021), and Asdiv (Miao et al., 2020), which span different difficulty levels of mathematical reasoning. We report accuracy (Acc), the average number of decoding steps (Steps), and latency (Lat., measured as average runtime) to provide a comprehensive evaluation. To verify that our domain-specific fine-tuning does not lead to capability forgetting on other general QA tasks, we also compare our fine-tuned model with the original model on several general benchmarks, including MBPP (Austin et al., 2021b), MMLU (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2022) (TQA), PIQA (Bisk et al., 2020), and ARC-C (Clark et al., 2018).

Implementation Details. We evaluate our method on three representative open-source dLLMs: LLaDA-8B-Instruct (LLaDA-8B-Ins) (Nie et al., 2025), LLaDA-1.5 (Zhu et al., 2025), and Dream-7B-Instruct (Ye et al., 2025) (Dream-7B-Ins). For semi-autoregressive masking, we set the block

Tab. 1: Efficiency comparison with other inference acceleration methods on four reasoning benchmarks. We report accuracy (Acc), the average number of decoding steps (Steps), and latency (Lat.).

Methods	GSM8K			MATH-500			SVMAP			Asdiv		
	Acc ↑	Steps ↓	Lat. ↓	Acc ↑	Steps ↓	Lat. ↓	Acc ↑	Steps ↓	Lat. ↓	Acc ↑	Steps ↓	Lat. ↓
LLaDA-8B-Ins												
Original (Nie et al., 2025) [NeurIPS2025]	82.8	512	11.1	39.0	512	12.4	88.7	512	7.3	84.5	512	7.3
Fast_dLLM (Wu et al., 2025) [ICLR2026]	82.3	91	2.8	39.0	148	4.8	88.7	65	1.9	83.6	65	1.9
KLASS (Kim et al., 2025) [NeurIPS2025]	80.9	129	4.4	37.2	201	7.0	85.7	95	3.3	83.8	96	3.2
dParallel (Chen et al., 2025b) [ICLR2026]	79.1	37	1.2	37.2	67	2.1	83.5	26	0.9	80.3	27	0.9
Ours	81.0	39	1.1	38.2	65	1.9	86.4	25	0.8	81.4	26	0.8
LLaDA-1.5												
Original (Ye et al., 2025) [arXiv2025]	82.5	512	6.0	40.5	512	8.1	87.3	512	6.9	84.8	512	6.8
Fast_dLLM (Wu et al., 2025) [ICLR2026]	82.8	80	2.0	40.6	121	4.0	88.6	64	1.7	84.3	64	1.6
KLASS (Kim et al., 2025) [NeurIPS2025]	80.5	129	4.4	39.2	202	7.0	86.0	93	3.2	83.7	96	3.2
Ours	80.2	34	0.9	39.2	81	2.6	87.0	35	0.9	83.1	31	0.8
Dream-7B-Ins												
Original (Ye et al., 2025) [arXiv2025]	82.0	512	4.1	42.0	512	4.7	87.3	512	3.3	86.3	512	3.2
Fast_dLLM (Wu et al., 2025) [ICLR2026]	82.3	103	2.4	41.1	117	2.8	87.7	85	2.0	85.6	84	2.0
KLASS (Kim et al., 2025) [NeurIPS2025]	80.2	217	8.6	42.0	174	6.6	86.0	179	6.8	83.9	175	6.6
dParallel (Chen et al., 2025b) [ICLR2026]	78.2	61	1.9	39.6	83	2.6	86.3	46	1.5	83.0	47	1.4
Ours	80.5	37	1.1	41.1	65	2.0	85.4	36	0.9	85.2	37	0.9

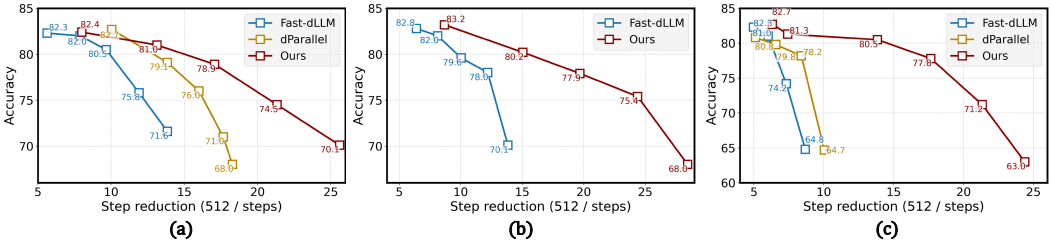


Fig. 4: Comparison of speed-accuracy trade-off curves for Fast-dLLM, dParallel, and our method on GSM8K with LLaDA-8B-Ins (a), LLaDA-1.5 (b), and Dream-7B-Ins (c). The x-axis shows the reduction in decoding steps relative to the original setting (512 steps).

length $L_b = 32$. We construct training data from the GSM8K and MATH-500 training sets and sample 4K instances for training. We train for 4 epochs with a batch size of 128 and a learning rate of 1×10^{-6} on H20 GPUs. For complex benchmarks such as MATH-500, we set the mask ratio to 50%; for other benchmarks, we set it to 90% to achieve stronger compression. During inference, following Fast-dLLM (Wu et al., 2025), we adopt a confidence-threshold semi-autoregressive remasking strategy and set the maximum generation length to 512 tokens. For suffix sparsification, we set the initial suffix window size D^{init} to 256 for a better balance between computation and performance.

4.2 Main Results

Results on Different Models. As shown in Tab. 1, our method consistently outperforms other methods in terms of the balance between speed and performance across three open-source dLLMs and different reasoning benchmarks. Specifically, on the GSM8K dataset, our method achieves approximately a 13× reduction in decoding steps with only a slight drop in accuracy relative to the original LLaDA model. Inference latency falls from 11.1 s to 1.1 s, yielding a 10× speedup on GSM8K. Compared with Fast-dLLM and KLASS, which adopt more advanced parallel decoding strategies, our method achieves a more significant reduction in decoding steps and latency, while maintaining competitive accuracy. Compared with dParallel, which uses 100K training examples for consistency distillation to support more aggressive parallel decoding, our method achieves a better balance between accuracy and efficiency while requiring only 4K training examples. For complex benchmarks such as MATH-500, which require more decoding steps for reasoning, the model's limited capability makes effective self-compression data harder to construct. Our method still significantly reduces decoding steps and latency while maintaining competitive accuracy.

Speed–Accuracy Trade-off. To further analyze the efficiency of different methods, we visualize the speed–accuracy trade-off curves of Fast-dLLM, dParallel and our method on the GSM8K dataset in Fig. 4. Specifically, we vary the confidence threshold to obtain different points on the trade-off curve. The horizontal axis represents the reduction factor of decoding steps compared to the original method (512 steps), and the vertical axis represents the accuracy. Our method consistently

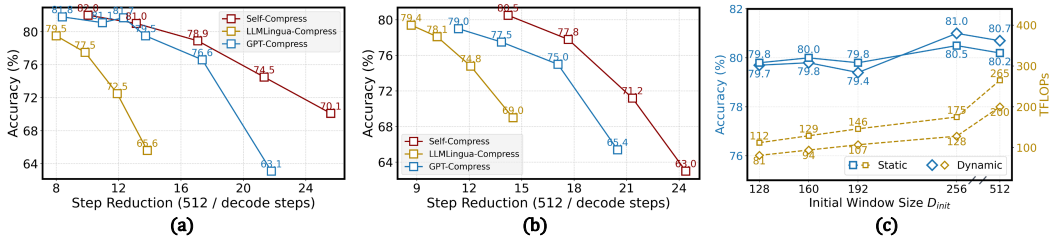


Fig. 5: (a) and (b) compare different CoT compression strategies on GSM8K for the LLaDA-8B-Ins and Dream-7B-Ins models, respectively. (c) shows accuracy and compute (measured in TFLOPs) for different suffix window sizes under the static and our dynamic strategies on GSM8K with the LLaDA-8B-Ins model.

achieves a better trade-off between speed and accuracy across different dLLMs. Specifically, on the Dream-7B-Ins model, with an $8\times$ reduction in decoding steps, our method only suffers a minor drop in accuracy, while Fast-dLLM and dParallel suffer drops of 17.0% and 3.8% respectively, which demonstrates the effectiveness of our method in accelerating dLLM inference.

4.3 Ablation Studies

Different CoT Compression Strategies. Instead of using external models, we propose utilizing the dLLM itself as the CoT compressor to construct the variable-length CoT trajectories for training. To verify the effectiveness of self-compression, we compare it with two other CoT compression strategies on GSM8K using the LLaDA-8B-Ins and Dream-7B-Ins models: `GPT-Compress` (which uses `GPT-5.2` for CoT compression) and `LLMLingua-Compress` (Xia et al., 2025) (which uses LLMLingua to prune less important tokens). As shown in Fig. 5, we find that `LLMLingua-Compress` yields the worst speed-accuracy trade-off among the three methods. We attribute this to LLMLingua's limited ability to identify redundant tokens and to the semantic discontinuity introduced by token-level pruning. Though `GPT-Compress` generally provides stronger compression quality, the resulting CoT trajectories exhibit a larger distribution gap from native dLLM trajectories, which reduces parallel decoding efficiency after fine-tuning. Our *Self-Compress* strategy strikes a better balance: it achieves effective CoT compression while retaining parallel decoding capability through self-distillation.

Tab. 2: Comparison of suffix sparsification strategies on GSM8K with the LLaDA-8B-Ins model. We report accuracy and average compute (measured in TFLOPs) for different numbers of suffix tokens, N.

N	128	160	192	256	512
Uniform	78.9/112	79.8/129	79.8/146	**80.3/175**	80.5/255
dPad	79.3/108	79.4/124	79.7/139	**80.9/172**	80.5/255
PASS	79.7/81	79.8/94	79.4/107	**81.0/128**	79.8/200

Different Suffix Sparsification Strategies. In Tab. 2, we compare three suffix sparsification strategies on GSM8K: *(i)* uniform sampling, *(ii)* dPad with Gaussian sampling, and *(iii)* our PASS strategy. Compared with dPad, the PASS strategy consistently reduces compute while maintaining or improving accuracy; for example, at N=256, compute drops from 172 to 128 TFLOPs ($\sim$25.6%) while achieving slightly better accuracy. PASS also outperforms Uniform sampling: at N=256, PASS reaches 81.0% accuracy with 128 TFLOPs, whereas Uniform sampling obtains 80.3% accuracy with 175 TFLOPs. These results indicate that the PASS strategy can achieve the best computational efficiency.

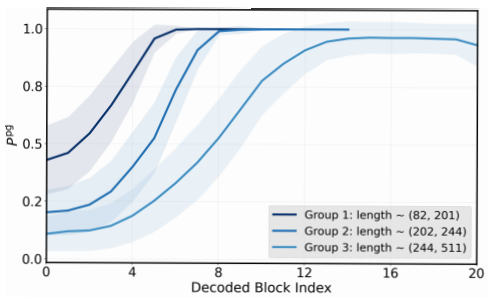


Fig. 6: Estimated progress P_b^{pg} across decoding blocks on GSM8K, grouped by final generation length (tokens). Progress generally increases during decoding.

Effectiveness of Decoding Progress Estimation. In Fig. 6, we visualize the decoding progress estimation P_{pg} across decoding blocks. We divide the samples into three groups based on their final generation lengths, and for each group, we compute the average P_{pg} at each decoding block. We can see that as the generation proceeds, P_{pg} exhibits a steady increase. Moreover, we find that samples with longer trajectories tend to have lower P_{pg} at the initial decoding stages, while shorter samples exhibit higher P_{pg}. These results suggest that P_{pg} can effectively reflect the progress of decoding, providing a reasonably accurate estimation of decoding progress.

Mask Ratio Hyper-parameter Analysis. We analyze the impact of different mask ratios on model performance using the LLaDA-8B-Ins model on the GSM8K dataset. We compute the average response length (Avg length), the number of decoding steps, and the accuracy under different mask ratios in Tab. 3. Compared with the original model, our method substantially shortens response lengths and reduces the number of decoding steps. More aggressive compression (*e.g.*, a mask ratio of 0.5) yields shorter responses and fewer steps, but causes a larger accuracy drop. As the mask ratio increases, accuracy generally improves, while response length and the number of decoding steps increase overall. In this work, we set the mask ratio to 0.9 for a favorable balance.

Tab. 3: Hyperparameter analysis of training mask ratios for the LLaDA-8B-Ins model on GSM8K.

Mask Ratio	0.5	0.7	0.8	0.9	1.0	original
Avg length	93	127	135	148	138	310
Steps	27	36	37	39	38	512
Acc	69.4	76.8	78.5	81.0	79.9	82.8

Suffix Window Size Analysis. In Fig. 5(c), we visualize the accuracy and compute (measured in TFLOPs) of different window sizes for both *Static* (fixed suffix window size) and *Dynamic* (PASS) strategies on the GSM8K dataset with the LLaDA-8B-Ins model. We notice that for both *Static* and *Dynamic* strategies, accuracy does not necessarily improve as the suffix window size increases, possibly because additional redundant context can interfere with the model's reasoning. For our *Dynamic* strategy, accuracy is relatively low with a small initial window size (*e.g.*, 128). As the window narrows with decoding progress, the suffix context available to the model becomes limited, which can hinder reasoning and reduce accuracy. Increasing the initial window size improves accuracy and gives the *Dynamic* strategy a better trade-off between accuracy and computation than the *Static* strategy by providing more useful context. In this work, we set D^{init} to 256 for a better balance between accuracy and computation on the GSM8K dataset.

Tab. 4: Comparison on general benchmarks with LLaDA-8B-Ins model.

Methods	**MBPP**	**MMLU**	**TQA**	**PIQA**	**ARC-C**
Original	38.1	59.3	31.2	76.0	52.8
Ours	37.8	59.0	30.1	76.2	53.6

Capabilities Analysis on General Benchmarks. In this work, we primarily use mathematical datasets for fine-tuning and evaluation. To verify that our method does not overfit to the mathematical domain and forget its capabilities on general question-answering (QA) tasks, we compare our fine-tuned model with the original model on several general benchmarks. As shown in Tab. 4, our method achieves comparable performance to the original model across all benchmarks, demonstrating that our domain-specific fine-tuning does not lead to significant capability forgetting on general QA tasks.

5 Conclusion

In this paper, we investigate dLLM acceleration from a trajectory-level perspective and show that oversized canvases induce two major inefficiencies: redundancy in CoT and redundancy in suffix context computation. To address these issues, we propose Beyond-the-Canvas dLLM (BC-dLLM), which integrates two complementary strategies: Self-Guided CoT Compression (SGC) to decouple reasoning length from canvas size, and Progress-Aware Suffix Sparsification (PASS) to dynamically reduce suffix computation as decoding progresses. By combining these strategies, the BC-dLLM framework substantially improves inference efficiency without sacrificing model capability. In future work, we will explore methods such as RLHF to further enhance the efficiency of dLLMs. We will also evaluate BC-dLLM on longer and more diverse multimodal inputs, which is important for real-world applications.

AI USE STATEMENT

We used OpenAI's ChatGPT and Codex tools during manuscript preparation to edit the language and to draft and revise LaTeX source. Their assistance was limited to improving grammar, wording, and the flow of the written presentation, as well as editing the LaTeX used to typeset text, equations, figures, tables, and cross-references. The tools were not used to develop or refine the research idea, formulate the method, design the experiments, analyze data, or generate results. The authors reviewed the AI-assisted edits and remain responsible for all scientific content and claims in the final manuscript.

ETHICS STATEMENT

BC-dLLM aims to make diffusion language model inference more efficient by reducing unnecessary reasoning steps and suffix-context computation. This may lower the computational cost of using these models, but it does not address biases, factual errors, or unsafe outputs that may arise from the underlying models and their training data. Greater efficiency could also make it easier to produce misleading or harmful content at scale. Because our evaluation focuses on reasoning and general question-answering benchmarks, the reported results should not be taken as evidence of safety or reliability in other settings. Applications of BC-dLLM should therefore include safety evaluations and safeguards appropriate to their intended use, together with review of generated outputs where needed.

REPRODUCIBILITY STATEMENT

We provide the main methodological and experimental details needed to assess and reproduce our results. Sec. 3 explains how SGC constructs compact reasoning trajectories and uses them for fine-tuning, then defines the decoding-progress estimate and the PASS suffix-window rule. The models, benchmarks, evaluation metrics, training settings, and decoding configuration are reported in Sec. 4.1. Appendix A gives further details on the baselines and prompt templates, while Algorithm 1 in Appendix B specifies the PASS inference procedure. Additional evidence, including the suffix-attention pilot study, suffix-token visualization, compute ablation, and a case study, appears in Appendices C–F.

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 and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

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

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021a.

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

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.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

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

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

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, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, et al. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18581–18597, 2025.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Hengyu Fu, Baihe Huang, Virginia Adams, Charles Wang, Venkat Srinivasan, and Jiantao Jiao. From bits to rounds: Parallel decoding with exploration for diffusion language models. *arXiv preprint arXiv:2511.21103*, 2025.

Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Empowering diffusion models on the embedding space for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4664–4683, 2024.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 4521–4534, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

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

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24312–24320, 2025.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models. *arXiv preprint arXiv:2511.05664*, 2025.

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

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *Transactions on Machine Learning Research*.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Compressing chain-of-thought in llms via step entropy. *arXiv preprint arXiv:2508.03346*, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025a.

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

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

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.

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. *arXiv preprint arXiv:2406.03736*, 2024.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

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

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.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

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*, 2025.

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

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

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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.

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

Yangyang Zhong, Yanmei Gu, Zhengqing Zang, Xiaomeng Li, Yuqi Ding, Xibei Jia, Yuting Shen, Zhenzhong Lan, Liwang Zhu, Weiping Liu, et al. Parallelism and generation order in masked diffusion language models: Limits today, potential tomorrow. *arXiv preprint arXiv:2601.15593*, 2026.

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.

Zijian Zhu, Fei Ren, Zhanhong Tan, and Kaisheng Ma. Es-dllm: Efficient inference for diffusion large language models by early-skipping. *arXiv preprint arXiv:2603.10088*, 2026.

Appendix Contents

A Implementation Details 16

B Pseudo Code of Progress-Aware Suffix Sparsification 16

C Suffix Context Attention Pilot Study 17

D Visualization of the Number of Sampled Suffix Tokens 18

E Ablation of the Required Compute 18

F Case Studies 19

G Limitation and Future Work 20

A Implementation Details

We evaluate the original dLLM under its official default inference setting (unmasking one token per step). We also compare our approach with recent methods that improve parallel decoding, including (1) Fast-dLLM (Wu et al., 2025), which uses a confidence-threshold strategy to select tokens for unmasking and enables KV caching for both prefix and suffix tokens;
(2) KLASS (Kim et al., 2025), which uses token-level KL divergence and confidence to guide parallel decoding;
(3) dParallel (Chen et al., 2025b), which introduces a certainty-forcing distillation mechanism to improve parallel decoding and uses an entropy-threshold strategy to select tokens for unmasking during inference.
Following previous works (Xia et al., 2025; Kang et al., 2025) in the autoregressive (AR) domain, we adopt a conditional training approach that trains language models with CoT data from different thinking modes simultaneously by inserting distinct prompt tokens in instructions. In this work, we use conditional prompt tokens: (i) *think detailedly and carefully* for slow mode, and (ii) *think compactly and concisely* for fast mode. Figures 7 and 8 show the two prompt templates for LLaDA-8B-Ins. The placeholder {{problem}} is replaced with the input problem; \n denotes a newline in the prompt.

Slow-thinking prompt template

LLaDA-8B-Ins

```
<|startoftext|><|start_header_id|>user<|end_header_id|>
You need to put your final answer in \boxed{}.  Please think
detailedly and carefully.
This is the problem:\n
{{problem}}<|eot_id|><|startoftext|>
<|start_header_id|>assistant<|end_header_id|>\n
```

Fig. 7: Slow-thinking prompt template for LLaDA-8B-Ins.

Fast-thinking prompt template

LLaDA-8B-Ins

```
<|startoftext|><|start_header_id|>user<|end_header_id|>
You need to put your final answer in \boxed{}.  Please think
compactly and concisely.
This is the problem:\n
{{problem}}<|eot_id|><|startoftext|>
<|start_header_id|>assistant<|end_header_id|>\n
```

Fig. 8: Fast-thinking prompt template for LLaDA-8B-Ins.

B Pseudo Code of Progress-Aware Suffix Sparsification

We provide the pseudo code for the inference process based on the proposed progress-aware suffix sparsification strategy in Algorithm 1. The algorithm takes as input the model M_θ, sequence length L, confidence threshold τ, block length L_b, and block number N_b. It outputs the generated sequence $\mathbf{x}$ after performing parallel decoding with dynamic suffix sparsification based on the estimated progress of decoding. During the decoding process, we also introduce an early stopping mechanism based on the presence of the EOS token. If the generated content contains an <EOS> token, we will terminate the decoding process early.

Algorithm 1: Inference based on Progress-Aware Suffix Sparsification (PASS)

Input: model M_θ, sequence length L, confidence threshold τ, block length L_b, block number $N_\mathrm{b} = L/L_\mathrm{b}$
Output: Generated sequence $\mathbf{x}$
Initialize $\mathbf{x} \leftarrow [\mathrm{Prompt}; \mathrm{MASK}_{1:L}]$;
for $b \leftarrow 1$ **to** N_b **do**
  // Estimate the progress of decoding
  $\ell \leftarrow M_\theta(\mathbf{x}).\mathrm{logits}$;
  $p_\theta \leftarrow \mathrm{softmax}(\ell)$;
  $P_b^{\mathrm{pg}} = 1 - \prod_{j \in \mathcal{M}_b} \left(1 - p_\theta(\mathbf{x}^j = \texttt{<EOS>} \mid \mathbf{x}_t)\right)$;
  // Sparsify the suffix tokens in $\mathbf{x}$
  $\mathbf{x}_{\mathrm{forward}} \leftarrow f_{\mathrm{PASS}}(\mathbf{x}, P_b^{\mathrm{pg}})$;
  for $t \leftarrow 1$ **to** L_b **do**
    $\ell \leftarrow M_\theta(\mathbf{x}_{\mathrm{forward}}).\mathrm{logits}$;
    $p_\theta \leftarrow \mathrm{softmax}(\ell)$;
    $c \leftarrow \max(p_\theta)$; // Get per-token confidence scores
    $\mathrm{high_conf} \leftarrow (c > \tau)$;
    // Update sequence
    $\mathrm{unmask_at_indices}(\mathbf{x}, p_\theta, \mathrm{high_conf})$;
  end
  // check EOS or Any mask token
  $\mathrm{Is_masked} \leftarrow \mathrm{AnyMask}(\mathbf{x})$;
  $\mathrm{Is_EOS} \leftarrow \mathrm{AnyEOS}(\mathbf{x})$;
  if Is_EOS or $\neg$ Is_masked **then**
    break;
  end
end
Return $\mathbf{x}$

C Suffix Context Attention Pilot Study

Figure 9 shows the attention distribution across prefix, current-block, and suffix token positions in the GSM8K pilot study. Within the suffix region, attention is highest next to the current block and decreases with distance.

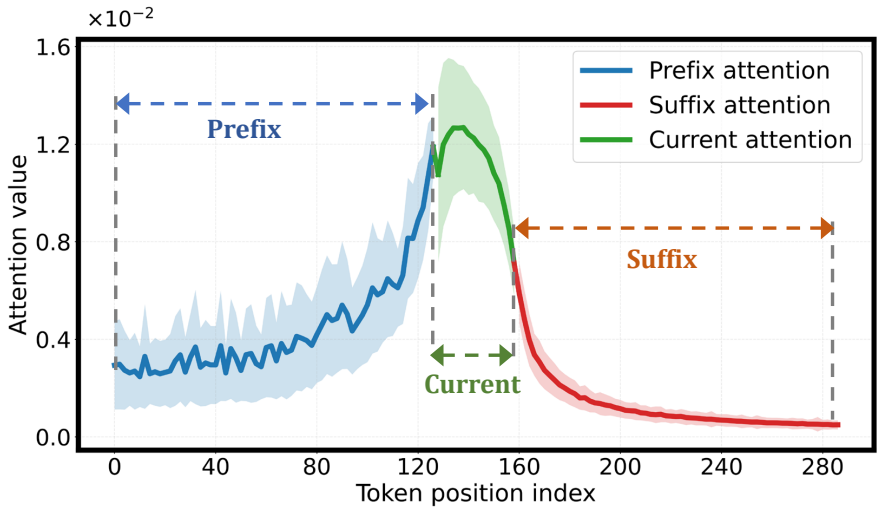


Fig. 9: Attention distribution across prefix, current-block, and suffix positions on GSM8K. The shaded regions denote variance bands.

D Visualization of the Number of Sampled Suffix Tokens

In Fig. 10, we show how the number of suffix tokens sampled by LLaDA-8B-Ins changes across blocks on GSM8K and MATH-500. On GSM8K, the model samples more suffix tokens in the first few blocks, but the number rapidly decreases as the block index increases, eventually approaching half of its initial value. On MATH-500, the number decreases more gradually and reaches a lower level only in the last few blocks. We hypothesize that for more complex problems, the model typically requires longer chains of thought and is less likely to generate termination signals in the earlier blocks, thus sampling more suffix tokens. For simpler problems, the model may generate termination signals more quickly, resulting in fewer sampled suffix tokens in the earlier blocks. This reflects the different decoding dynamics of the model's reasoning process on datasets with varying difficulty levels.

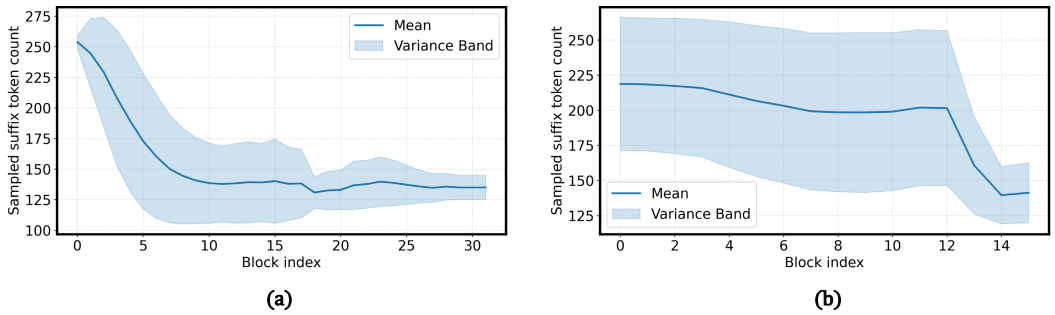


Fig. 10: (a) and (b) show the number of suffix tokens sampled by LLaDA-8B-Ins across decoding blocks on GSM8K and MATH-500, respectively.

E Ablation of the Required Compute

We further analyze the impact of our proposed self-guided CoT compression (SGC) and progress-aware suffix sparsification (PASS) strategies on the average required compute (TFLOPs) for the LLaDA-8B-Ins model on GSM8K and MATH-500 datasets. As shown in Fig. 11, compared to the

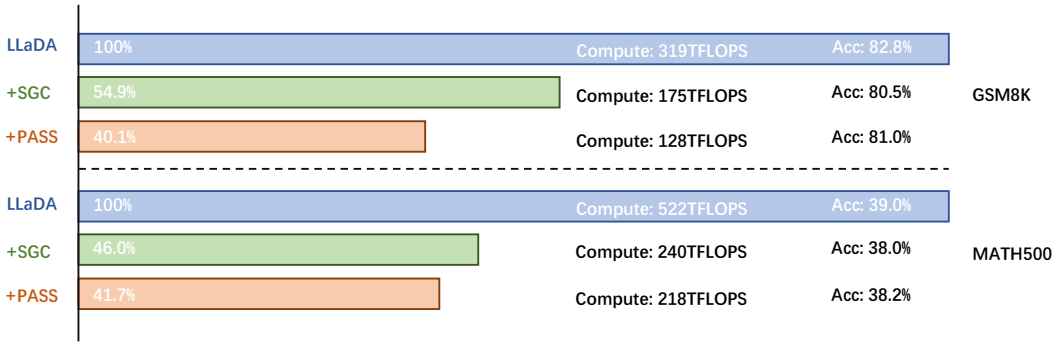


Fig. 11: Comparison of the average required compute (TFLOPs) on GSM8K and MATH-500 datasets with the LLaDA-8B-Ins model.

original dLLM, our method significantly reduces the required compute by approximately 60% on both datasets. Specifically, on the GSM8K dataset, the average required compute for the original dLLM is approximately 319 TFLOPs, which is reduced to about 175 TFLOPs after introducing our SGC strategy. With the further introduction of the PASS strategy, the average required compute is further reduced to around 128 TFLOPs. On the MATH-500 dataset, a similar trend is observed, with the average required compute decreasing from approximately 522 TFLOPs for the original dLLM to about 240 TFLOPs with SGC, and further down to around 218 TFLOPs with the addition of PASS.

Additionally, we note that on the MATH-500 dataset, the reduction in required compute brought by the SGC strategy is more significant, while the reduction brought by the PASS strategy is relatively

smaller. This may be because for more complex problems, the model needs to sample more suffix tokens in more blocks, which is consistent with the trend we observed in Fig. 10.

F Case Studies

We present a GSM8K case study with LLaDA-8B-Ins to illustrate how our method influences generation behavior. With CoT compression, the model gives the correct answer in 150 tokens. The original 260-token response instead answers a different question about signatures per week. Both responses are reproduced below, with formatting adjusted for typesetting.

(a) LLaDA-8b-Instruct Compact Output Example

Question. Carol and Jennifer are sisters from Los Angeles who love collecting signatures from celebrities. During their summer break from school, the sisters spend every afternoon collecting signatures. After five weeks, Carol and Jennifer compare their autograph books, counting up the number of signatures each sister has collected. Carol has 20 signatures in her book, and Jennifer has 44. The sisters have three more weeks of summer vacation, and they decide they want to reach 100 signatures between them by the end of the summer. How many signatures do the sisters need to collect to reach their goal?

Response.
To determine how many signatures Carol and Jennifer need to collect to reach their goal, we start by calculating the total number of signatures they currently have together.
Carol has 20 signatures and Jennifer has 44 signatures. Therefore, the total number of signatures they currently have is:

$$20 + 44 = 64$$

Their goal is to reach 100 signatures between them. To find out how many more signatures they need, we subtract the number of signatures they currently have from their goal:

$$100 - 64 = 36$$

Thus, they need to collect $\boxed{36}$ signatures to reach their goal.
Response length: 150

(b) LLaDA-8b-Instruct Original Output Example

Question. Carol and Jennifer are sisters from Los Angeles who love collecting signatures from celebrities. During their summer break from school, the sisters spend every afternoon collecting signatures. After five weeks, Carol and Jennifer compare their autograph books, counting up the number of signatures each sister has collected. Carol has 20 signatures in her book, and Jennifer has 44. The sisters have three more weeks of summer vacation, and they decide they want to reach 100 signatures between them by the end of the summer. How many signatures do the sisters need to collect to reach their goal?

Response.
To determine how many signatures Carol and Jennifer need to collect to reach their goal of 100 signatures between them by the end of the summer, we can follow these steps:
1. Calculate the total number of signatures they currently have.
2. Determine how many more signatures they need to reach their goal.
3. Calculate the number of signatures they need to collect in the remaining three weeks.

First, let's calculate the total number of signatures they currently have:
- Carol has 20 signatures.
- Jennifer has 44 signatures.
- Total current signatures = $20 + 44 = 64$.

Next, we determine how many more signatures they need to reach their goal:
- Their goal is 100 signatures.
- Signatures needed = $100 - 64 = 36$.

Finally, we calculate the number of signatures they need to collect in the remaining three weeks:
- They have 3 weeks left.
- Signatures needed per week = $\frac{36}{3} = 12$.

Therefore, the sisters need to collect $\boxed{12}$ signatures each week to reach their goal.
Response length: 260

G Limitation and Future Work

Our method improves dLLM decoding efficiency from a trajectory-level perspective, but it still has several limitations. First, self-guided CoT compression relies on high-quality compact reasoning trajectories. When the model's capabilities are limited or the data quality is poor, the generated compressed chains of thought may fail to adequately cover the key information in the original chains of thought, leading to a decline in reasoning performance. In addition, we mainly focus on reasoning tasks with long chains of thought, and our CoT compression method may not bring significant efficiency improvements for tasks that do not require long chains of thought (*e.g.*, cloze tests, code completion). In the future, we plan to explore more advanced RLHF strategies to further enhance the quality of the generated compact reasoning trajectories and to investigate the effectiveness of our method on diverse multimodal reasoning tasks that require long chains of thought, such as visual question answering and video understanding.