Diffusion LLMs Can Do Faster-Than-AR Inference via Discrete Diffusion Forcing

Xu Wang[1]*, **Chenkai Xu**[1]*, **Yijie Jin**[1,3], **Jiachun Jin**[1], **Hao Zhang**[2], **Zhijie Deng**[1]†
[1]Shanghai Jiao Tong University [2]University of California San Diego [3]Shanghai University
{wangxu60,132435xck,jiachun.jin,zhijied}@sjtu.edu.cn, jyj2431567@shu.edu.cn

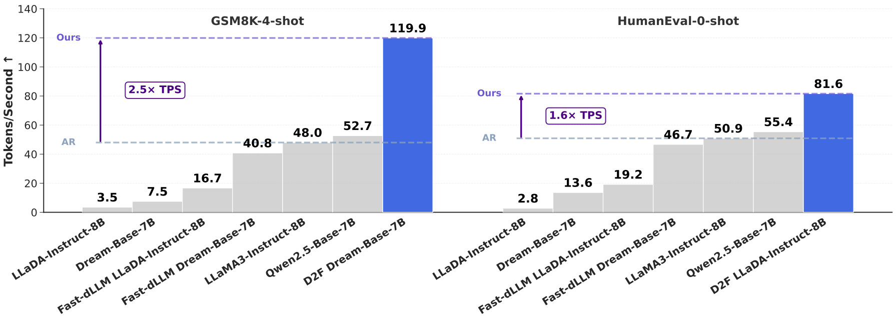

Figure 1: **D2F dLLMs surpass AR LLMs in inference speed for up to** $2.5\times$**.** Comparison of inference throughput among D2F dLLMs, vanilla dLLMs like Dream-Base-7B (Ye et al., 2025) and LLaDA-Instruct-8B (Nie et al., 2025), previous SOTA acceleration method Fast-dLLM (Wu et al., 2025), and similarly-sized AR baselines (Yang et al., 2024a; Grattafiori et al., 2024). The max generation length is set to 512.

Abstract

Diffusion Large Language Models (dLLMs) have emerged as a promising alternative to autoregressive (AR) LLMs for text generation, with the potential to decode multiple tokens in a single iteration. However, none of the existing open-source dLLMs have achieved superior inference speed over AR LLMs of similar size. This paper breaks this barrier based on a simple and effective strategy named discrete diffusion forcing (D2F). D2F equips dLLMs with two key capabilities: (1) block-wise autoregressive generation to enable KV cache utilization; (2) prediction of following tokens without requiring completion of prior blocks for inter-block parallel decoding. In this way, the vanilla dLLMs are refurbished into an AR-diffusion hybrid paradigm for efficient inference. D2F can be implemented with an asymmetric distillation process based on pre-trained dLLMs. We further propose a pipelined parallel decoding algorithm, which enables a trade-off between efficiency and efficacy. Empirically, D2F dLLMs achieve more than $\mathbf{2.5}\times$ inference speed than LLaMA3 and Qwen2.5 on GSM8K. Compared to vanilla dLLMs like LLaDA and Dream, the acceleration can be more than $\mathbf{50}\times$ while maintaining comparable output quality. The code is available at https://github.com/zhijie-group/Discrete-Diffusion-Forcing.

1 Introduction

Large Language Models (LLMs) have maintained a dominant position in text generation for a long time (Achiam et al., 2023; Touvron et al., 2023a; Yang et al., 2025; Touvron et al., 2023b; Grattafiori et al., 2024). Recently, Diffusion Large Language Models (dLLMs) have emerged as a promising

[*] Equal contribution.
[†] Corresponding author.

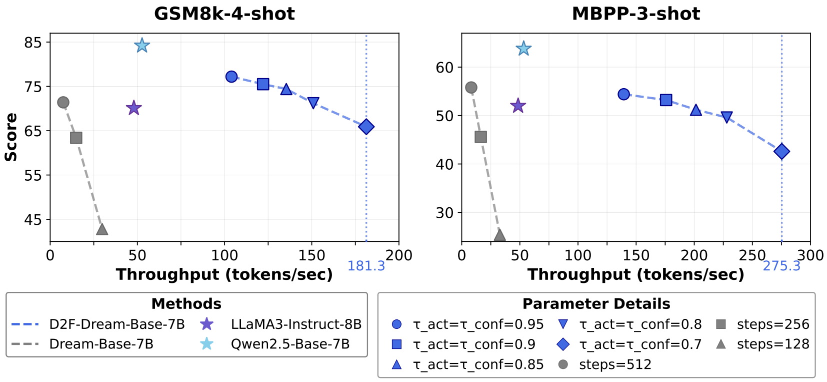

Figure 2: **Throughput vs. performance trade-off.** As shown, D2F achieves a more favorable trade-off compared to vanilla dLLMs. Refer to Section 4.3 for the details of the hyperparameters τ_{add} and τ_{conf}.

alternative to LLMs (Ye et al., 2025; Nie et al., 2025; Zhu et al., 2025), acknowledged by their potential to generate multiple text tokens in parallel. For example, closed-source dLLMs such as Gemini Diffusion (Google DeepMind, 2025) and Mercury (Inception et al., 2025) can yield thousands of tokens per second, 5-10 times faster than traditional autoregressive (AR) LLMs of similar size.

However, the speed merits of dLLMs have not been demonstrated within the open-source community. Approaches to bridge the gap include designing KV cache strategies (Arriola et al., 2025; Liu et al., 2025; Ma et al., 2025) and improving parallel sampling algorithms (Wu et al., 2025; Wei et al., 2025; Hu et al., 2025). For instance, Block Diffusion (Arriola et al., 2025) turns dLLMs into a block-wise sequential generation paradigm to enjoy KV cache. Yet, it precludes the inter-block parallelism, a crucial factor for efficient inference. Fast-dLLM (Wu et al., 2025) also adopts a block-wise generation order to facilitate the reuse of states of generated tokens and incorporates confidence-based remasking for parallel decoding. Nonetheless, the cached states can be biased after subsequent tokens are decoded due to the bidirectional nature of the involved attention.

This paper achieves the first breakthrough in accelerating dLLMs to a faster-than-AR regime. Conceptually, we aim to embrace block-wise sequential generation to facilitate KV cache utilization, yet reject the dilemma that the decoding of subsequent blocks must wait for preceding blocks to be fully denoised. This implies a novel AR-diffusion hybrid paradigm, which, however, cannot be realized through naive teacher-forcing training. This is because teacher-forcing requires complete preceding information to predict subsequent content. Interestingly, this insight shares the spirit with the diffusion forcing (DF) (Chen et al., 2024a) technique developed specifically for continuous-space diffusion models. In this sense, this paper forms an extension of DF to discrete data, giving rise to the discrete diffusion forcing (D2F) method for dLLM acceleration.

Concretely, D2F dLLMs learn to denoise a sequence of token blocks with monotonically increasing mask ratios in parallel. Naturally, preceding blocks can finish before subsequent ones, allowing their KV states to be cached for subsequent computations. Note that we constrain the attention to be block-wise causal to ensure the KV cache remains accurate. For training efficiency, we distill D2F dLLMs from existing bidirectional attention dLLMs using an asymmetric distillation loss. In inference, we design a pipelined parallel decoding algorithm which enables inter-block parallelism and offers a decent trade-off between inference efficiency and performance (see Figure 2).

Distilled on the Bespoke-Stratos-17k (Labs, 2025) dataset for 12 hours with 8 NVIDIA A100-PCIe-40GB GPUs, D2F can accelerate LLaDA-Instruct-8B (Nie et al., 2025) and Dream-Base-7B (Ye et al., 2025) for more than 50× without compromising average performance on mathematical and programming benchmarks, including GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021b). More importantly, D2F-Dream-Base-7B is up to 2.5× faster than LLaMA3-Instruct-8B (Grattafiori et al., 2024) on GSM8K and 1.6× faster on HumanEval, establishing the first open-source dLLMs that outruns AR ones.

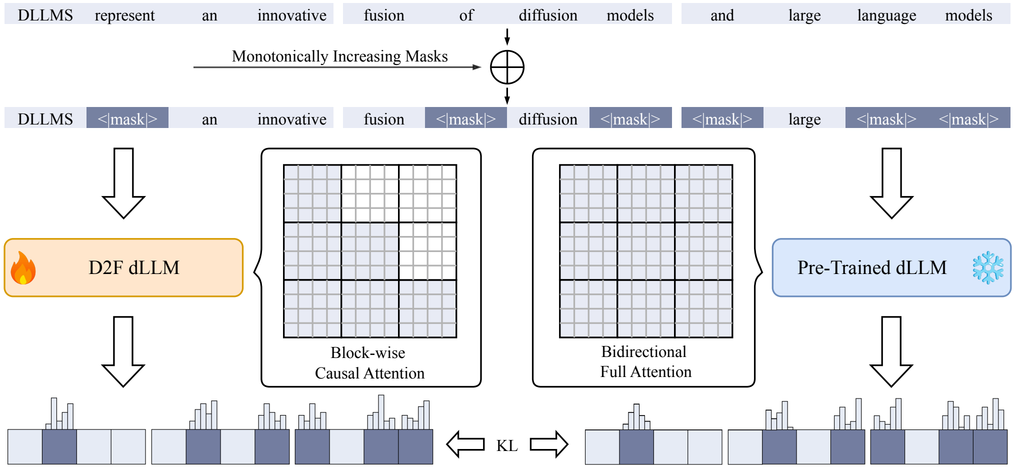

Figure 3: **An overview of discrete diffusion forcing (D2F).** During training, the answer is divided into blocks with progressively increasing masking ratios. D2F dLLM is specified with a block-wise causal attention mask, and trained to mimic the prediction of a pre-trained bidirectional dLLM conditioned on partially denoised preceding tokens. This enables inter-block parallel decoding with KV cache compatibility during inference.

In summary, this work makes the following key contributions:

- **D2F, a novel AR-diffusion hybrid model for text generation**, which combines the advantages of AR-style KV cache with the highly parallelized decoding of diffusion models.
- **The first faster-than-AR dLLMs**, which achieve up to 2.5× faster inference than prevalent AR LLMs like LLaMA3 (Grattafiori et al., 2024) and Qwen2.5 (Yang et al., 2024a).

2 Related Work

Diffusion Large Language Models (dLLMs). The landscape of language generation has long been defined by autoregressive models (Achiam et al., 2023; Guo et al., 2025; Liu et al., 2024). These models, renowned for their high-quality output, are inherently limited by a sequential, token-by-token decoding process. To overcome this latency bottleneck, dLLMs have emerged (Ye et al., 2025; Nie et al., 2025; Google DeepMind, 2025; Inception et al., 2025; Zhu et al., 2025). Instead of generating text sequentially, dLLMs operate by iteratively denoising a fully masked sequence, a process that enables the parallel prediction of all tokens at once. This approach, which draws from non-autoregressive principles, utilizes a bidirectional attention mechanism to achieve a more holistic understanding of context. Recent large-scale dLLMs, such as LLaDA (Nie et al., 2025), trained from scratch, and Dream (Ye et al., 2025), initialized from pre-trained AR weights, have demonstrated performance competitive with leading ARMs, establishing dLLMs as a powerful alternative for high-quality, parallelizable text generation.

Acceleration of dLLMs. dLLMs suffer from slower inference than autoregressive models due to incompatibility with standard KV cache and limited parallelization. Existing acceleration methods fall into two categories. First, caching-based approaches (Liu et al., 2025; Wu et al., 2025; Ma et al., 2025) develop approximate schemes to reuse computations for static sequence parts, as standard KV cache is incompatible with bidirectional attention. Second, sampling optimization methods reduce decoding steps through confidence-aware strategies (Wu et al., 2025), auxiliary model guidance (Israel et al., 2025), or adaptive decoding speeds (Wei et al., 2025). However, these methods achieve limited speedups due to inherent approximations and auxiliary model overhead, often failing to match the efficiency of comparable AR (Achiam et al., 2023; Touvron et al., 2023a; Yang et al., 2025; Touvron et al., 2023b; Grattafiori et al., 2024) models. Our approach fundamentally restructures generation into a block-autoregressive framework that is compatible with standard KV cache, while enabling prediction of subsequent tokens without completing previous blocks.

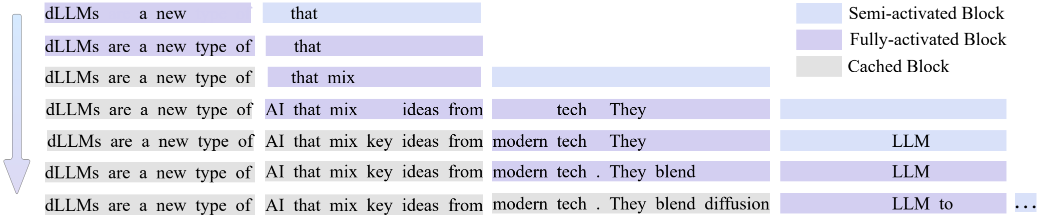
dLLMs a new | that
dLLMs are a new type of | that
dLLMs are a new type of | that mix
dLLMs are a new type of | AI that mix ideas from | tech They
dLLMs are a new type of | AI that mix key ideas from | modern tech They | LLM
dLLMs are a new type of | AI that mix key ideas from | modern tech . They blend | LLM
dLLMs are a new type of | AI that mix key ideas from | modern tech . They blend diffusion | LLM to | ...
Semi-activated Block
Fully-activated Block
Cached Block

Figure 4: **Visualization of the pipelined parallel decoding of D2F dLLMs.** As shown, a pipeline of blocks are decoded in parallel. A new block is dynamically added when the completion ratio of the last block exceeds a threshold τ_{add} ($= \frac{1}{3}$ here). The new block is initially semi-activated and will transition to a fully-activated state when its predecessor reaches the completion threshold τ_{act} ($= \frac{5}{6}$ here). The fully-activated blocks differ from semi-activated ones in that they would decode multiple tokens in each step more aggressively.

AR-diffusion hybrid models. Recent studies have explored incorporating the speed advantages of autoregressive models into diffusion-based frameworks, particularly in tasks such as video generation (Yin et al., 2025; Po et al., 2025; Sun et al., 2025; Huang et al., 2025). In video generation, it is common to model the temporal dependencies between frames using an autoregressive approach, while applying denoising within each frame independently. This hybrid architecture leverages the acceleration benefits of KV cache enabled by autoregressive modeling, while preserving the high generation quality of diffusion-based methods.

3 Preliminary: Diffusion Large Language Models (DLLMs)

Diffusion models, originally developed for continuous data, have achieved state-of-the-art results in image synthesis (Esser et al., 2024; Chen et al., 2024b; Podell et al., 2023; Labs et al., 2025) and video generation (Peng et al., 2025; Zheng et al., 2024; Yang et al., 2024b; Bao et al., 2024). In recent years, advances in the theory of discrete diffusion (Austin et al., 2021a; Lou et al., 2023; Campbell et al., 2022) have facilitated the emergence of large-scale dLLMs (Nie et al., 2025; Ye et al., 2025) for text generation tasks, which have demonstrated performance competitive with their AR counterparts, while offering the potential for parallel generation.

The majority of successful dLLMs operate under a masked diffusion paradigm (Nie et al., 2025). This process begins with a forward process, where an original text sequence of L tokens, $Y^0 = \{y_1^0, \ldots, y_L^0\}$, is progressively corrupted into a noisy sequence Y^t over a continuous time schedule $t \in [0, 1]$. This corruption is routinely achieved by replacing original tokens independently with a special `[MASK]` token. Typically, we can define the conditional distribution as

$$q(Y^t|Y^0) = \prod_{i=1}^{L} q(y_i^t|y_i^0), \quad \text{where} \quad q(y_i^t|y_i^0) = \begin{cases} 1-t, & \text{if } y_i^t = y_i^0 \\ t, & \text{if } y_i^t = \texttt{[MASK]} \end{cases}. \tag{1}$$

Consequently, Y^1 represents a fully masked sequence. dLLMs are designed to learn a parameterized model $p_\theta(Y^0|Y^t)$ to reverse the forward process, hence enabling denoising from the fully masked sequence Y^1 to language samples Y^0. This formulation allows the model p_θ to predict all mask tokens in the sequence simultaneously at each inference step, forming the basis of dLLMs' theoretical potential for high-speed, parallel generation.

However, practical implementations of dLLMs face severe inference efficiency bottlenecks. First, the use of bidirectional attention mechanisms fundamentally conflicts with KV cache, leading to significant redundant computation across denoising steps. Second, the reliance on a conditional independence assumption for parallel decoding makes it hard to generate interdependent tokens (Song & Zhou, 2025), so more iterative steps are required for high-quality outputs. While prior works attempt to mitigate these issues, for instance, by introducing block-wise sequential generation to enable KV cache (Arriola et al., 2025) or implementing approximate KV cache (Wu et al., 2025; Liu et al., 2025), they fail to simultaneously achieve both precise KV cache and efficient parallel decoding. Consequently, no open-source dLLMs has yet matched the inference speed of AR models.

Algorithm 1 Asymmetric Distillation for D2F

```
Require: Pre-trained dLLM $p_{\phi^-}$; D2F dLLM $p_\theta$; block size $k$; training dataset $\mathcal{D}$.
while training do
    Sample a sequence $Y$ from $\mathcal{D}$.
    Divide $Y$ into $N$ blocks $\{Y_{B_1}, \ldots, Y_{B_N}\}$, each of size $k$.
    Sample a monotonic noise schedule $\{t_1, \ldots, t_N\}$ where $t_1 < \cdots < t_N$.
    For each $i \in \{1, \cdots, N\}$, corrupt $Y_{B_i}$ to $Y_{B_i}^{t_i}$ using Eq. 1.
    Predict distributions for each block $Y_{B_i}^0$ with:
        Teacher (global view): $p_{\phi^-}(Y_{B_i}^0 | Y_{B_1}^{t_1}, \ldots, Y_{B_N}^{t_N})$
        Student (causal view): $p_\theta(Y_{B_i}^0 | Y_{B_1}^{t_1}, \ldots, Y_{B_i}^{t_i})$
    Update $p_\theta$ based on the loss $\mathcal{L}_{\text{D2F}}$ defined in Eq. 3.
```

4 Method

The paper derives discrete diffusion forcing (D2F) to accelerate the inference of dLLMs to a faster-than-AR regime. Refer to Figures 3 and 4 for an overview of the training and inference of D2F.

4.1 Discrete Diffusion Forcing

D2F endows dLLMs with two critical capabilities: (1) *block-level AR generation*, which enables efficient standard KV cache and significantly reduces computational redundancy; and (2) *inter-block parallel decoding*, where the model is trained to predict future blocks from incomplete, partially reconstructed predecessors to maximize the number of decoded tokens per inference step. This naturally gives rise to an AR-diffusion hybrid modeling paradigm, but naive teacher-forcing training, as employed in block diffusion (Arriola et al., 2025), cannot lead to the second capability. To address this, D2F trains the model to perform conditional denoising of the current block based on a partially denoised prefix, enabling more coherent and efficient sequential generation.

Concretely, D2F partitions a clean sequence Y^0 into N blocks of size k. Let $B_i := \{(i-1)*k, \ldots, i*k-1\}$ denote the token indices in the i-th block and Y_{B_i} denote the corresponding subsequence. In the forward process, D2F applies a monotonically increasing noise schedule ($t_1 < t_2 < \cdots < t_N$) to the N blocks, i.e., $Y^t = \{Y_{B_1}^{t_1}, \ldots, Y_{B_N}^{t_N}\}$. Namely, the earlier blocks in Y^t are progressively less masked (i.e., more complete), while later blocks remain increasingly masked (i.e., more uncertain). For the reverse process, D2F trains a θ-parameterized model to characterize:

$$p_\theta(Y^0|Y^t) = \prod_{i=1}^{N} p_\theta(Y_{B_i}^0 | Y_{B_1}^{t_1}, \ldots, Y_{B_i}^{t_i}). \tag{2}$$

Intuitively, the learned model can first finalize the decoding of preceding blocks while simultaneously advancing the denoising of subsequent ones, which effectively enables inter-block parallel decoding. By preserving a causal attention structure across blocks—wherein intra-block attention remains bidirectional—we can cache the KV states of already decoded blocks for exact reuse, thereby reducing redundant computations and improving inference efficiency.

Connection to diffusion forcing (Chen et al., 2024a). From a high-level perspective, our approach bears a strong conceptual resemblance to diffusion forcing (DF) (Chen et al., 2024a), originally developed for continuous-space diffusion models and notably applied in video generation (Yin et al., 2025). Both methods involve predicting the tokens of the next block conditioned on a noisy, incomplete premise, and our framework can be viewed as an extension of DF to discrete sequences. Such a principled extension motivates our naming of *discrete diffusion forcing*.

4.2 Asymmetric Distillation

Noting that training a dLLM with billions of parameters from scratch can be costly (Nie et al., 2025; Ye et al., 2025), we propose to distill a D2F dLLM from a pre-trained vanilla dLLM available in the open-source community. Let p_{ϕ^-} denote the standard bidirectional teacher dLLM and p_θ the

Algorithm 2 Pipelined Parallel Decoding for D2F Inference

```
Require: D2F model $p_\theta$; thresholds $\tau_{\text{add}}, \tau_{\text{act}}, \tau_{\text{conf}}$.
Initialize $Y = \{Y_{B_1}\}$ with a block of mask tokens.
while generation is not complete do
    if the ratio of decoded tokens in $Y_{B_{-1}}$ exceeds $\tau_{\text{add}}$ and <|EOS|> not in $Y$ then
        Append a new fully masked block with semi-activated state to $Y$.
    Forward pass of $Y$ with D2F dLLM $p_\theta$ using cached KV
    for the active block $Y_{B_i}$ in $Y$ do
        Let $J_i$ record the set of token positions in $B_i$ with $> \tau_{\text{conf}}$ prediction confidence
        if $Y_{B_i}$ is fully-activated and $J_i$ is $\emptyset$ then
            Add the token position with the highest confidence to $J_i$
        Sample tokens with positions in $J_i$ and remask other positions
        if the ratio of decoded tokens in $Y_{B_{i-1}}$ exceeds $\tau_{\text{act}}$ then
            Set $Y_{B_i}$ to be fully-activated
    Update KV cache for completed blocks
```

student D2F dLLM (θ is initialized as ϕ^-). The distillation minimizes the following loss

$$\mathcal{L}_{\text{D2F}} = \mathbb{E}_{t_1 < \cdots < t_N} \left[\sum_{i=1}^{N} D_{\text{KL}} \left(p_\theta(Y_{B_i}^0 | Y_{B_1}^{t_1}, \ldots, Y_{B_i}^{t_i}) \| p_{\phi^-}(Y_{B_i}^0 | Y_{B_1}^{t_1}, \ldots, Y_{B_N}^{t_N}) \right) \right], \tag{3}$$

where D_{KL} represents the KL divergence aggregated over the mask tokens. As shown, the distillation is asymmetric—the teacher p_{ϕ^-} predicts for each block $Y_{B_i}^0$ with a global view of all noisy blocks while the student p_θ learns to approximate using only a causally restricted view. In this way, the mask prediction capabilities of existing dLLMs can be embedded into a new D2F dLLM. This objective also connects to CausVid (Yin et al., 2025), which distills existing holistic diffusion video generators into streaming ones. As for model architecture, the student differs from the teacher solely in attention masks—it uses the block-wise causal attention instead of a bidirectional one.

We summarize the whole algorithmic process in Algorithm 1.

4.3 Pipelined Parallel Decoding

As illustrated in Figure 4, we introduce a pipelined parallel decoding algorithm for the inference of D2F dLLMs. Specifically, we maintain a sliding window of active blocks and dynamically append a new fully masked block of tokens once the decoding progress of the last block exceeds a threshold τ_{add}. The dynamic strategy significantly reduces the per-step computational cost compared to maintaining a full sequence of massive blocks throughout inference.

Observing that aggressive decoding of a newly added block can degrade performance, we incorporate a dual-state decoding mechanism. Concretely, the newly added block is initialized in a semi-activated state to enable conservative parallel decoding, and will become fully activated when its predecessor has finished τ_{act} of the decoding—i.e., sufficient contextual information has been accumulated to support aggressive decoding of the latter block. Following Fast-dLLM (Wu et al., 2025), semi-activated blocks admit tokens with confidence above a threshold τ_{conf}, while fully activated blocks additionally enforce the selection of the most confident token when no such token exists.

The synergy between the dynamic block management and dual-state mechanism conjoins per-step efficiency and inter-block parallelism. It is interesting to note that our approach also shares conceptual similarities with prior work in video generation (Teng et al., 2025). More algorithmic details are provided in Algorithm 2 and a hyperparameter analysis is in Table 3.

5 Experiments

This section details the experimental setup and presents the results of D2F dLLMs.

Test Set	Method	TPS ↑	Latency (s) ↓	Gen. Length	Score ↑
GSM8K 4-shot	LLaDA-Instruct	7.2 (1.0x)	32.3 (1.0x)	231	77.4
	dLLM-Cache	20.1 (2.8x)	11.5 (2.8x)	231	77.5
	Fast-dLLM (Prefix-Cache)	33.3 (4.6x)	7.0 (4.6x)	232	77.8
	Fast-dLLM (Dual-Cache)	35.2 (4.9x)	6.6 (4.9x)	232	**78.9**
	D2F-LLaDA	**52.5** (7.3x)	**2.8** (11.5x)	144	77.3
MBPP 3-shot	LLaDA-Instruct	0.9 (1.0x)	71.4 (1.0x)	65	**39.0**
	dLLM-Cache	2.3 (2.6x)	28.3 (2.5x)	66	37.0
	Fast-dLLM (Prefix-Cache)	13.0 (14.4x)	4.9 (14.6x)	64	37.6
	Fast-dLLM (Dual-Cache)	15.3 (17.0x)	3.8 (18.8x)	58	36.4
	D2F-LLaDA	**47.6** (52.9x)	**1.4** (51.0x)	68	38.0
HumanEval 0-shot	LLaDA-Instruct	2.8 (1.0x)	38.8 (1.0x)	107	36.0
	dLLM-Cache	4.5 (1.6x)	23.3 (1.7x)	104	39.0
	Fast-dLLM (Prefix-Cache)	13.7 (4.9x)	7.4 (5.2x)	102	38.4
	Fast-dLLM (Dual-Cache)	19.2 (6.9x)	5.2 (7.5x)	100	35.4
	D2F-LLaDA	**81.6** (29.1x)	**1.6** (24.3x)	133	**40.2**
Math 4-shot	LLaDA-Instruct	21.1 (1.0x)	11.5 (1.0x)	243	23.7
	dLLM-Cache	26.9 (1.3x)	9.1 (1.3x)	246	23.2
	Fast-dLLM (Prefix-Cache)	47.7 (2.3x)	5.2 (2.2x)	246	22.4
	Fast-dLLM (Dual-Cache)	42.5 (2.0x)	5.8 (2.0x)	246	22.4
	D2F-LLaDA	**90.2** (4.3x)	**4.3** (2.7x)	384	**29.1**

Table 1: **Performance comparison of various acceleration methods on LLaDA-Instruct-8B**. Speedup ratios are shown in (green). All baseline methods use the default sampling configuration from the original LLaDA implementation. See Appendix A.3 for detailed hyperparameters.

5.1 Experiment Settings

We perform evaluation on two representative dLLMs: LLaDA-Instruct-8B (Nie et al., 2025) and Dream-Base-7B (Ye et al., 2025). For our distillation-based training, we utilize a dataset derived from the Bespoke-Stratos-17k benchmark (Labs, 2025). Specifically, we use two publicly available collections sourced from the HuggingFace Hub[1], where a third party had previously generated responses to the problems from Bespoke-Stratos-17k using the Qwen2.5-7B model (Yang et al., 2024a). These collections are pre-filtered to a maximum length of 600 tokens. The `q_proj`, `v_proj`, `k_proj`, and `o_proj` modules of D2F dLLMs are tuned with the LoRA (Hu et al., 2022) technique, using the rank of 32, the scaling factor of 32, and the dropout rate of 0.1. During training, We truncate or pad all sequences to a final length of 512 tokens, and employ a block size of 16. We applied block-wise monotonically increasing mask ratios, with a maximum threshold of 0.7 and a minimum threshold of 0.3, inspired by Block Diffusion (Arriola et al., 2025). During inference, unless otherwise specified, τ_{conf} is set to 0.9, τ_{add} is set to 0.1, and τ_{act} is set to 0.95. The models are trained for 12 hours using an AdamW optimizer with a constant learning rate of 10^{-5}. All training and inference are conducted on a setup comprising 8 NVIDIA A100-PCIe-40GB GPUs. Detailed hyperparameter configurations for inference on each benchmark are provided in the Appendix A.3.

5.2 Main Results

Benchmarks. Following established conventions, performance evaluation of D2F is conducted across mathematical reasoning and code generation benchmarks, including GSM8K (Cobbe et al., 2021), GSM8K-CoT (Chain-of-Thought reasoning variant of GSM8K), Math (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021b).

[1] The datasets, processed and provided by a third party, are available at: `https://huggingface.co/datasets/Lansechen/bs17k_collection_filtered_hard_maxlength600` and `https://huggingface.co/datasets/Lansechen/bs17k_collection_filtered_easy_maxlength600`.

Test Set	Method	TPS ↑	Latency (s) ↓	Gen. Length	Score ↑
GSM8K-CoT 8-shot	Dream-Base	9.5 (1.0x)	26.8 (1.0x)	255	75.0
	dLLM-Cache	26.0 (2.7x)	9.8 (2.7x)	255	72.0
	Fast-dLLM (Prefix-Cache)	50.3 (5.3x)	5.1 (5.3x)	255	76.6
	Fast-dLLM (Dual-Cache)	49.8 (5.2x)	5.1 (5.3x)	255	75.0
	D2F-Dream	**91.2** (9.6x)	**2.8** (9.6x)	256	**77.6**
MBPP 3-shot	Dream-Base	10.4 (1.0x)	24.6 (1.0x)	256	56.2
	dLLM-Cache	25.5 (2.5x)	10.0 (2.5x)	256	52.6
	Fast-dLLM (Prefix-Cache)	71.6 (6.9x)	3.6 (6.8x)	256	**56.4**
	Fast-dLLM (Dual-Cache)	73.2 (7.0x)	3.5 (7.0x)	256	51.0
	D2F-Dream	**105** (10.1x)	**2.3** (10.7x)	240	55.2
HumanEval 0-shot	Dream-Base	20.2 (1.0x)	12.6 (1.0x)	255	54.3
	dLLM-Cache	23.2 (1.1x)	11.0 (1.1x)	255	**55.5**
	Fast-dLLM (Prefix-Cache)	62.4 (3.1x)	4.1 (3.1x)	255	54.3
	Fast-dLLM (Dual-Cache)	60.0 (3.0x)	4.3 (2.9x)	255	53.0
	D2F-Dream	**73.2** (3.6x)	**3.1** (4.1x)	227	54.3
Math 4-shot	Dream-Base	9.9 (1.0x)	25.8 (1.0x)	256	35.8
	dLLM-Cache	12.7 (1.3x)	20.2 (1.3x)	256	34.5
	Fast-dLLM (Prefix-Cache)	65.6 (6.6x)	3.9 (6.6x)	256	**37.6**
	Fast-dLLM (Dual-Cache)	67.0 (6.8x)	3.8 (6.8x)	256	37.1
	D2F-Dream	**98.8** (10.0x)	**2.6** (9.9x)	256	35.4

Table 2: **Performance comparison of various acceleration methods on Dream-Base-7B.** Speedup ratios relative to the baseline are shown in (green). The max generation length is set to 256. See Appendix A.3 for detailed hyperparameters.

Baselines. Comprehensive comparisons are established between D2F and state-of-the-art acceleration strategies, including Fast-dLLM (Wu et al., 2025) and dLLM-Cache (Liu et al., 2025), implemented on LLaDA-Instruct-8B (Nie et al., 2025) and Dream-Base-7B (Ye et al., 2025) architectures. Additional benchmarking against leading auto-regressive (AR) LLMs of comparable scale, specifically LLaMA3-Instruct-8B (Grattafiori et al., 2024) and Qwen2.5-Base-7B (Yang et al., 2024a), demonstrates the efficacy of D2F in achieving faster-than-AR inference speeds.

Quantitative Results. As shown in Figure 1, D2F represents the first open-source dLLMs to surpass state-of-the-art AR LLMs in inference speed. The maximum generation length is set to 512 for all methods to ensure fairness. Concretely, D2F-Dream-Base-7B achieves a throughput of 119.9 tokens/s on GSM8K. This constitutes a 2.5× speedup over LLaMA3-Instruct-8B (48.0 tokens/s) and a 2.3× speedup over Qwen2.5-Base-7B (52.7 tokens/s).

D2F also significantly accelerates the baseline dLLMs while maintaining equivalent average performance. As detailed in Table 1, D2F-LLaDA-Instruct-8B achieves a 52.9× speedup (47.6 tokens/s vs. baseline 0.9 tokens/s) with minimal performance difference (38.0 vs. baseline 39.0). Additionally, D2F-LLaDA-Instruct-8B consistently outperforms prior dLLM acceleration techniques. On MATH, it attains 90.2 tokens/s, providing a 2.1× speedup over Fast-dLLM's Dual-Cache (42.5 tokens/s). To ensure a fair comparison, the hyperparameter configuration of baseline is aligned with the paper of LLaDA (Nie et al., 2025).

Similarly, as shown in Table 2, D2F-Dream-Base-7B reaches 91.2 tokens/s on GSM8K-CoT. This corresponds to a 9.6× speedup over Dream-Base-7B (9.5 tokens/s) with slight performance improvement, and a 1.8× speedup over Fast-dLLM (49.8 tokens/s). These results demonstrate that D2F not only surpasses existing acceleration methods but also enables dLLMs to exceed AR LLMs in throughput, substantially enhancing their practical applicability. Since the base model struggles to generate the stop token correctly, we set a unified maximum length of 256 for all methods for a fair comparison, a setting adopted by the paper of dLLM-cache (Liu et al., 2025). More detailed hyperparameter settings are available in Appendix A.3.

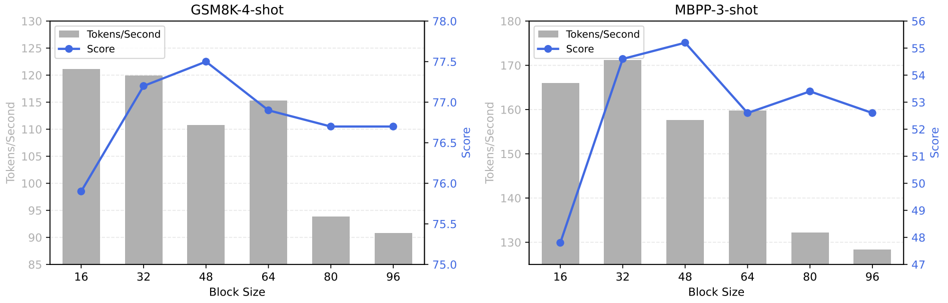

Figure 5: **Ablation study on the block size during inference.** The block size for inference is tested with integer multiples of the training block size (16). All experiments were conducted with a maximum length of 512, $\tau_{\text{conf}} = 0.9$, $\tau_{\text{add}} = 0.1$, and $\tau_{\text{act}} = 0.95$.

$\tau_{\text{act}} = \tau_{\text{conf}}$	τ_{add}	**TPS ↑**	**Score ↑**
0.95	0.95	105.2	76.9
	0.7	107.2	77.2
	0.5	106.3	77.3
	0.1	104.0	**77.7**
0.90	0.90	124.5	74.7
	0.7	126.2	75.7
	0.5	124.7	76.2
	0.1	122.1	**76.4**
0.85	0.85	136.8	72.6
	0.7	139.0	74.2
	0.5	138.5	74.0
	0.1	135.4	**75.0**

Table 3: **Ablation study of inference hyperparameter on the GSM8K-4-shot.** We measured the impact of τ_{conf}, τ_{act}, and τ_{add} on performance and speed. The results are tested with a block size of 32 and a maximum length of 512.

5.3 Ablations and Analysis

This section presents ablation studies to dissect the contributions of key components within D2F. All ablation experiments are conducted on the D2F-Dream-Base-7B model unless otherwise specified.

Throughput-performance trade-off. Figure 2 illustrates the throughput-performance trade-off for different methods on GSM8K and MBPP. D2F-Dream-Base-7B employs a fixed $\tau_{\text{add}} = 0.1$ and a block size of 32, with unified thresholds $\tau_{\text{act}} = \tau_{\text{conf}}$ being varied. Results demonstrate that D2F-Dream-Base-7B achieves significantly higher efficiency than baselines. For instance, on GSM8K, D2F-Dream-Base-7B attains 150.9 tokens/sec with a score of 71.2, achieving 3.1× the throughput of LLaMA3-Instruct-8B (48.0 tokens/sec) while exceeding its score (70.1). In contrast, Dream-Base-7B exhibits substantial performance degradation at higher throughput: reducing sampling steps from 512 to 128 causes its GSM8K score to drop from 71.4 to 42.8. This demonstrates the superior capability of D2F-Dream-Base-7B in maintaining performance during accelerated inference.

Effect of block size for inference. Figure 5 demonstrates the influence of block size on inference performance and speed. Overall, increasing block size consistently reduces throughput while yielding an initial performance improvement followed by deterioration. Optimal block size selection moderately reduces throughput but maximizes performance—for example, a block size of 48 achieves the peak GSM8K score of 77.5, surpassing smaller block sizes such as 16 with score of 75.9 despite throughput reduction.

Ablation on hyperparameters of inference pipeline. Table 3 details an ablation study examining inference hyperparameters τ_{conf}, τ_{act}, and τ_{add}. We analyze the impact of the relationship between

Benchmark	τ_{act}	Model	TPS ↑	Score ↑
MBPP 3-shot	0.95	random D2F	147.2 **171.2**	49.6 **54.6**
	0.90	random D2F	148.5 **175.5**	48.6 **53.2**
	0.85	random D2F	150.6 **177.5**	47.6 **52.6**
GSM8k 4-shot	0.95	random D2F	114.5 **119.9**	77.1 **77.2**
	0.90	random D2F	116.7 **123.5**	**76.5** 76.4
	0.85	random D2F	118.8 **124.8**	**75.9** 75.4

Table 4: **Ablation study on the noise scheduling strategy on the MBPP-3-shot and GSM8k-4-shot.** We compare our D2F method against a random baseline, which is trained with an independent random noise schedule for each block. All experiments use a maximum length of 512, a block size of 32, and fixed hyperparameters $\tau_{conf} = 0.9$ and $\tau_{add} = 0.1$.

τ_{add} and τ_{act}. The setting $\tau_{\text{add}} = \tau_{\text{act}}$ represents a single-state pipeline where new blocks are always fully-activated upon addition. In contrast, our dual-state approach ($\tau_{\text{add}} < \tau_{\text{act}}$) allows for a more conservative initial state. The results demonstrate that this dual-state strategy outperforms the single-state baseline. For instance, with $\tau_{\text{act}} = 0.85$, $\tau_{\text{add}} = 0.7$ increases the score from 72.6 to 74.2 while improving throughput from 136.8 to 139.0 TPS. Further reducing τ_{add} yields additional score improvements at the cost of marginal throughput reduction.

Advantages of D2F training over fully random masks. As shown in Table 4, the D2F training strategy significantly outperforms a baseline utilizing random noise schedules per block. On MBPP, D2F demonstrates comprehensive superiority, achieving a 5.0-point score improvement (54.6 versus 49.6) and 24.0 TPS gain at $\tau_{\text{act}} = 0.95$. These results substantiate that the structured progressive noising of D2F delivers a more effective training objective than fully random masking.

6 Conclusion

In this work, we introduce Discrete Diffusion Forcing (D2F), a novel training paradigm for dLLMs. D2F employs a generation scheme that conditions on partially predicted tokens from previous blocks to predict the next block, thereby supporting KV cache and enabling parallel generation across multiple blocks, resulting in significantly faster inference. Empirically, extensive experiments demonstrate that D2F achieves the milestone of being the first dLLM to support faster-than-AR inference.

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.

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.

Fan Bao, Chendong Xiang, Gang Yue, Guande He, Hongzhou Zhu, Kaiwen Zheng, Min Zhao, Shilong Liu, Yaole Wang, and Jun Zhu. Vidu: a highly consistent, dynamic and skilled text-to-video generator with diffusion models. *arXiv preprint arXiv:2405.04233*, 2024.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024a.

Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *European Conference on Computer Vision*, pp. 74–91. Springer, 2024b.

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

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Google DeepMind. Gemini Diffusion. `https://deepmind.google/models/gemini-diffusion`, 2025. Accessed: 2025-05-24.

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

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

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

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

Labs Inception, 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.

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

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL `https://arxiv.org/abs/2506.15742`.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

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

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

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

Xiangyu Peng, Zangwei Zheng, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, Wenjun Li, et al. Open-sora 2.0: Training a commercial-level video generation model in 200 k. *arXiv preprint arXiv:2503.09642*, 2025.

Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. *arXiv preprint arXiv:2505.20171*, 2025.

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

Jiaming Song and Linqi Zhou. Ideas in inference-time scaling can benefit generative pre-training algorithms. *arXiv preprint arXiv:2503.07154*, 2025.

Mingzhen Sun, Weining Wang, Gen Li, Jiawei Liu, Jiahui Sun, Wanquan Feng, Shanshan Lao, SiYu Zhou, Qian He, and Jing Liu. Ar-diffusion: Asynchronous video generation with auto-regressive diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 7364–7373, 2025.

Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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

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.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

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.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024b.

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`.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 22963–22974, 2025.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

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.

A Appendix

A.1 Ablation Studies on D2F Components

To isolate the performance gains from different components of our D2F framework, we conduct ablation studies on both the **D2F-LLaDA** (Table 5) and **D2F-Dream** (Table 7) models. We compare two configurations:

- **Cache-only**: This configuration utilizes the block-wise causal attention mechanism to enable standard KV cache but does not employ the parallel, asynchronous block generation pipeline. Generation proceeds serially, one block at a time.
- **Cache + Para**: This is the full D2F method, combining KV cache with our pipelined parallel decoding algorithm.

The results clearly demonstrate that while enabling KV cache (Cache-only) provides a substantial speedup, the addition of our parallel decoding pipeline (Cache + Para) further accelerates inference by a significant margin (e.g., from 2.4x to 7.3x on GSM8K for LLaDA), highlighting the efficacy of the asynchronous generation strategy.

A.2 Control Experiment for Data Contribution

To prove that our performance gains stem from the D2F framework and not the training data, we conducted a control experiment. We created a control model, Dream-Base*, by directly fine-tuning the original Dream-Base on the same distillation dataset. For a fair comparison, this fine-tuning used LoRA with the exact same parameters as our D2F training.

The results are in Table 6. While Dream-Base* shows minor score improvements on some tasks, its inference speed drops significantly. This speed degradation is due to the computational overhead of the added LoRA layers. This control experiment confirms that our D2F methodology, rather than the data, is the key driver of the exceptional inference acceleration.

A.3 Hyperparameter Details

Table 8 details the specific hyperparameter configurations used for both the baseline models and our D2F models across all evaluated benchmarks. Baseline settings are adopted from their respective original works to ensure fair comparison. For our D2F models, we specify the maximum generation length, the inference block size, and the key thresholds for our pipelined parallel decoding algorithm: the token addition threshold (τ_{add}), the block activation threshold (τ_{act}), and the token confirmation confidence (τ_{conf}).

Test Set	Configuration	TPS ↑	Latency (s) ↓	Gen. Length	Score ↑
GSM8K	Cache-only	17.5 (2.4x)	8.3 (3.9x)	145	**78.1**
4-shot	Cache + Para	**52.5** (7.3x)	**2.8** (11.5x)	144	77.3
MBPP	Cache-only	18.1 (20.1x)	3.8 (18.8x)	69	37.6
3-shot	Cache + Para	**47.6** (52.9x)	**1.4** (51.0x)	68	**38.0**
HumanEval	Cache-only	28.0 (10.0x)	4.8 (8.1x)	135	**40.2**
0-shot	Cache + Para	**81.6** (29.1x)	**1.6** (24.3x)	133	**40.2**
Math	Cache-only	30.6 (1.5x)	12.6 (0.9x)	385	**29.6**
4-shot	Cache + Para	**90.2** (4.3x)	**4.3** (2.7x)	384	29.1

Table 5: Ablation study of our proposed method (**D2F-LLaDA**) on the **LLaDA-Instruct** model. Performance ratios relative to the LLaDA-Instruct baseline (from Table 1) are shown in (bright green).

Test Set	Method	TPS ↑	Latency (s) ↓	Gen. Length	Score ↑
GSM8K-CoT 8-shot	Dream-Base	**9.5** (1.0x)	**26.8** (1.0x)	255	75.0
	Dream-Base*	6.7 (0.7x)	29.9 (0.9x)	199	**77.8**
MBPP 3-shot	Dream-Base	**10.4** (1.0x)	**24.6** (1.0x)	256	56.2
	Dream-Base*	4.2 (0.4x)	27.4 (0.9x)	114	**57.4**
HumanEval 0-shot	Dream-Base	**20.2** (1.0x)	**12.6** (1.0x)	255	**54.3**
	Dream-Base*	8.8 (0.4x)	14.2 (0.9x)	125	51.8
Math 4-shot	Dream-Base	**9.9** (1.0x)	**25.8** (1.0x)	256	**35.8**
	Dream-Base*	5.4 (0.5x)	28.6 (0.9x)	154	33.4

Table 6: Comparison between the baseline Dream-Base model and its fine-tuned version, Dream-Base*. The fine-tuned version, highlighted with a gray background, is obtained by directly fine-tuning on the distillation data. Performance ratios relative to the baseline are shown in (bright green), and the highest score in each test set is marked in **bold**.

Test Set	Configuration	TPS ↑	Latency (s) ↓	Gen. Length	Score ↑
GSM8K-CoT 8-shot	Cache-only	40.7 (4.3x)	6.3 (4.3x)	256	**77.8**
	Cache + Para	**91.2** (9.6x)	**2.8** (9.6x)	256	77.6
MBPP 3-shot	Cache-only	40.5 (3.9x)	5.9 (4.2x)	240	54.2
	Cache + Para	**105** (10.1x)	**2.3** (10.7x)	240	**55.2**
HumanEval 0-shot	Cache-only	43.6 (2.2x)	5.1 (2.5x)	222	53.7
	Cache + Para	**73.2** (3.6x)	**3.1** (4.1x)	227	**54.3**
Math 4-shot	Cache-only	39.6 (4.0x)	6.5 (4.0x)	256	**35.9**
	Cache + Para	**98.8** (10.0x)	**2.6** (9.9x)	256	35.4

Table 7: Ablation study of our proposed method (**D2F-Dream**) on the **Dream-Base** model. Performance ratios relative to the Dream-Base baseline (from Table 2) are shown in (bright green).

Benchmark	Configuration	length	block_size	τ_{add}	τ_{act}	τ_{conf}
Method: D2F-LLaDA						
GSM8K	Baseline	256	8	–	–	–
	D2F	512	64	0.7	0.95	0.9
MBPP	Baseline	512	32	–	–	–
	D2F	512	32	0.9	0.95	0.9
HumanEval	Baseline	512	32	–	–	–
	D2F	512	32	0.1	0.95	0.9
MATH	Baseline	256	256	–	–	–
	D2F	512	32	0.1	0.95	0.9
Method: D2F-Dream						
GSM8K-COT	Baseline	256	–	–	–	–
	D2F	256	32	0.1	0.95	0.9
MBPP	Baseline	256	–	–	–	–
	D2F	256	48	0.1	0.95	0.9
HumanEval	Baseline	256	–	–	–	–
	D2F	256	32	0.9	0.95	0.95
MATH	Baseline	256	–	–	–	–
	D2F	256	64	0.1	0.95	0.9

Table 8: Sampling hyperparameters for the experiments in Table 1 and Table 2. Baseline settings are adopted from prior work. For D2F, the `length` parameter only sets a maximum generation limit without affecting the sampling distribution due to its block-wise nature. Notably, for the Dream-based model, we match D2F's maximum length to the baseline's, as the base model often fails to generate proper termination tokens.