TEMPORAL-AWARE TRAJECTORY SELF-DISTILLATION FOR FAST AND ACCURATE DIFFUSION LLM

Anonymous authors
Paper under double-blind review

ABSTRACT

Trajectory distillation is a promising training-based approach for accelerating diffusion large language models (dLLMs) by adapting them to generate more tokens per forward pass. However, existing methods construct trajectories from the base model alone and supervise masked tokens without distinguishing their decoding horizons, which limits the knowledge transferred through distillation and yields a suboptimal accuracy–parallelism trade-off. Enriching the knowledge carried by these trajectories and tailoring supervision to temporal decoding dynamics remain open challenges. Inspired by learning with privileged information, and motivated by our observation that the teacher–student distributional gap varies systematically with a token's time to reveal, we propose **TAD**, a **T**emporal-**A**ware trajectory self-**D**istillation framework. TAD conditions a frozen teacher on target responses available only during training to construct informative decoding trajectories. It then partitions masked tokens into near and distant subsets according to their temporal distance to decoding, applying hard cross-entropy supervision to near tokens for confident parallel decoding and soft distribution matching to distant tokens to avoid premature decisions while preserving future-planning knowledge. Across mathematical reasoning and code generation benchmarks, TAD improves the average accuracy of LLaDA-8B-Instruct from 46.1% to 51.5% and Dream from 58.9% to 61.2%. Its speed-oriented configuration further achieves the highest average Accuracy Under Parallelism scores of 256.6 and 193.9 on the two backbones respectively, which validate that TAD establishes a superior accuracy-parallelism frontier, simultaneously enhancing task accuracy and decoding throughput. Our code is available at https://anonymous.4open.science/r/TAD-B9D0/.

1 INTRODUCTION

Diffusion large language models (dLLMs) (Austin et al., 2021a; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025) have emerged as a promising alternative to autoregressive (AR) language models, with the ability to generate multiple tokens in parallel. However, realizing this parallelism without sacrificing generation quality remains challenging (Feng et al., 2026; Kang et al., 2025; Zhong et al., 2026; Zhang et al., 2026a). Since masked tokens are predicted independently within each forward pass, aggressive parallel decoding can overlook strong inter-token dependencies and substantially degrade generation quality (Wu et al., 2025b). Consequently, existing dLLMs (Nie et al., 2025; Ye et al., 2025) often rely on conservative decoding schedules with hundreds of denoising steps, leaving a large gap between their potential parallelism and realized efficiency.

To close this gap, recent studies have explored both training-free (Hong et al., 2025; Wu et al., 2025b; Xu et al., 2025a; Mohamed et al., 2025; Peng et al., 2025) and training-based methods (Chen et al., 2025; Zhang et al., 2026b; Qian et al., 2026; Wang et al., 2025a; Chen et al., 2026) for few-step parallel decoding. Training-free approaches provide plug-and-play acceleration by modifying the decoding process. However, the model itself remains unchanged, their attainable parallelism is bounded by the base model's ability (Lin et al., 2026). Training-based approaches instead adapt the model to few-step generation, allowing it to internalize parallel decoding behavior and thus offering greater potential to improve the accuracy–parallelism trade-off. Representative methods such as dParallel (Chen et al., 2025) and d3LLM (Qian et al., 2026) pursue this direction through trajectory distillation. Despite this promise, existing training-based trajectory-distillation methods face two distinct limitations. First, their trajectories are typically generated by the base model from

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

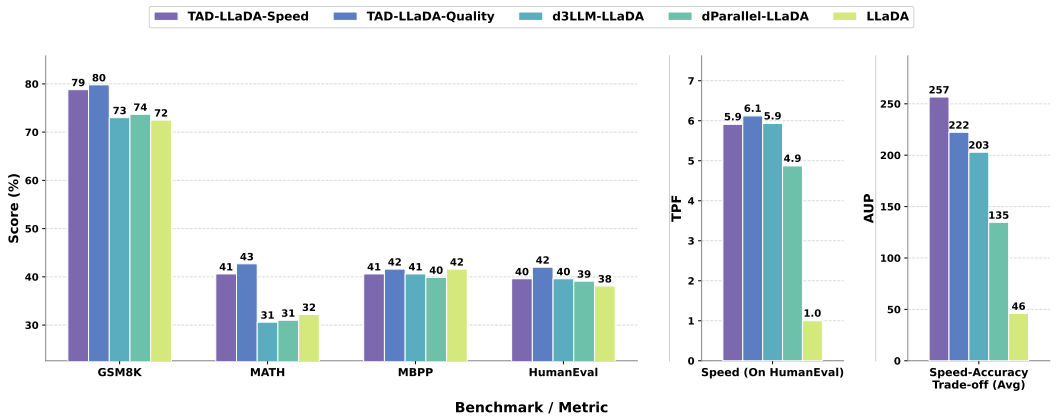


Figure 1: **TAD** achieves competitive accuracy on math and code benchmarks while substantially improving Accuracy Under Parallelism (AUP) (Qian et al., 2026) over acceleration baselines such as d3LLM (Qian et al., 2026) and dParallel (Chen et al., 2025).

the prompt alone. Consequently, distillation primarily reshapes existing decoding behavior without introducing knowledge beyond what the model already possesses, making it difficult to extend the model's original reasoning boundary. Second, existing methods either apply hard cross-entropy supervision to all masked tokens (Qian et al., 2026), forcing distant tokens to make early predictions, or supervise only tokens that will be decoded soon (Chen et al., 2025), discarding useful signals from distant tokens.

To address these limitations, we propose **TAD**, a **T**emporal-**A**ware trajectory self-**D**istillation framework that integrates two complementary ideas. First, we introduce *privileged-information-guided trajectory construction*. The frozen teacher is conditioned on additional privileged information, such as reference solutions or expert guidance (Zhao et al., 2026b; Hübotter et al., 2026; Shenfeld et al., 2026). The resulting trajectories carry supervision unavailable from prompt-only self-generation, enabling the student to extend its capabilities beyond the base model's original reasoning boundary. Second, motivated by the horizon-dependent teacher–student distributional gap observed in Figure 3a, *temporal-aware distillation* partitions masked tokens according to their time to reveal. TAD applies hard cross-entropy supervision to near tokens to encourage confident decoding, while matching the teacher's full distribution for distant tokens to avoid premature one-hot targets. Varying the temporal window naturally yields a Quality configuration for higher accuracy and a Speed configuration for more aggressive parallel decoding.

We evaluate TAD on mathematical reasoning and code generation benchmarks using LLaDA-8B-Instruct (Nie et al., 2025) and Dream-Instruct-7B (Ye et al., 2025). Experimental results demonstrate that TAD substantially pushes the Pareto frontier toward **fast and accurate dLLMs**. On LLaDA, TAD-Quality boosts MATH accuracy from 32.2% to 42.7% at 4.49 tokens per forward (TPF), while TAD-Speed achieves the highest Accuracy Under Parallelism (AUP) among competing acceleration baselines (Figure 1). These findings establish that dLLMs can achieve accelerated parallel decoding without sacrificing generation quality.

Our main contributions are as follows:

- We introduce **privileged-information-guided trajectory construction** for dLLMs, allowing training-only knowledge to shape decoding trajectories and be distilled into a student that operates without privileged information.
- We propose **temporal-aware trajectory self-distillation**, applying hard cross-entropy supervision to near tokens for confident parallel execution, and soft distribution supervision to distant tokens for long-horizon planning.
- We demonstrate on LLaDA and Dream that TAD consistently improves both task accuracy and decoding parallelism, achieving a substantially stronger accuracy–parallelism trade-off than existing acceleration methods.

2 PRELIMINARIES

Masked Diffusion Language Models (MDLMs). MDLMs (Austin et al., 2021a; Shi et al., 2024; Nie et al., 2025) formulate text generation as a forward corruption process followed by reverse denoising. Given a clean token sequence $\mathbf{x}_0 = (x_0^1, \ldots, x_0^L)$, the forward process constructs a partially masked sequence $\mathbf{x}_t$ by replacing each token with `[MASK]` with probability t, where $t \sim \mathcal{U}(0, 1)$:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \prod_{i=1}^{L} q(x_t^i \mid x_0^i), \quad q(x_t^i \mid x_0^i) = \begin{cases} 1 - t, & x_t^i = x_0^i, \\ t, & x_t^i = \texttt{[MASK]}. \end{cases} \tag{1}$$

The reverse process uses a model p_θ to predict all masked positions in parallel:

$$p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t) = \prod_{i : x_t^i = \texttt{[MASK]}} p_\theta(x_0^i \mid \mathbf{x}_t). \tag{2}$$

The model is trained by minimizing the negative log-likelihood over masked tokens (Nie et al., 2025):

$$\mathcal{L}(\theta) = -\mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_t} \left[\frac{1}{t} \sum_i \mathbf{1}[x_t^i = \texttt{[MASK]}] \log p_\theta(x_0^i \mid \mathbf{x}_t) \right]. \tag{3}$$

Trajectory Distillation. Trajectory distillation is widely used to improve parallel decoding in dLLMs (Chen et al., 2025; Qian et al., 2026; Zhang et al., 2026b). Let p_ϕ denote a pretrained teacher model and p_θ denote the student model. Given a prompt q, let $\mathbf{x}_0$ denote the target response and $\mathbf{x}_t$ denote a partially masked training state constructed from $\mathbf{x}_0$. A general trajectory-distillation objective trains the student to recover $\mathbf{x}_0$ from $\mathbf{x}_t$:

$$\mathcal{L}_{\text{traj}}(\theta) = -\mathbb{E}_{(\mathbf{x}_0, \mathbf{x}_t) \sim \mathcal{D}_{\text{traj}}} \left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t, q) \right], \tag{4}$$

where $\mathcal{D}_{\text{traj}}$ denotes the training pairs used for distillation.

Existing methods mainly differ in how $\mathbf{x}_t$ is constructed. We group the common designs into two types: (1) **Response-based reconstruction**, which first obtains a teacher-generated response $\mathbf{x}_0$ and constructs $\mathbf{x}_t$ by adding random noise to the response (Chen et al., 2025); and (2) **Trajectory-aware construction**, which records intermediate states from the teacher's decoding process and uses them as $\mathbf{x}_t$ for distillation (Qian et al., 2026; Zhang et al., 2026b). Both designs provide training states that better match the decoding process than standard diffusion training. However, such methods largely distill knowledge already present in the base model, failing to extend its capabilities, and they either enforce uniform supervision across all masked positions (Qian et al., 2026)or truncate supervision strictly to imminent tokens (Chen et al., 2025), leaving distant context unguided.

3 METHOD

We propose **TAD**, a **T**emporal-**A**ware trajectory self-**D**istillation framework that addresses the two limitations of existing trajectory distillation. First, we introduce **privileged-information-guided trajectory construction** to generate trajectories with additional information. Second, we introduce **temporal-aware self-distillation**, which applies different types of supervision to tokens according to when they will be decoded. Figure 2 provides an overview of the TAD framework.

3.1 PRIVILEGED-INFORMATION-GUIDED TRAJECTORY CONSTRUCTION

Existing trajectory distillation mainly distill knowledge already acquired by the base model. To provide stronger supervision, we introduce privileged information z_{priv} that is available during training, such as reference solutions, expert guidance, or few-shot examples. In our experiments, we use the ground-truth response a as z_{priv}.

Given a prompt q and privileged information z_{priv}, we let the teacher p_ϕ generate a fine-grained decoding trajectory by revealing exactly one token at each step. Following the standard diffusion notation, $\mathbf{x}_T$ denotes the initial masked state and $\mathbf{x}_0$ denotes the final decoded sequence. For any intermediate state $\mathbf{x}_t$, let $\mathcal{M}(\mathbf{x}_t) \triangleq \{i \mid x_t^i = \texttt{[MASK]}\}$ denote the set of its masked positions. At state $\mathbf{x}_t$, where $t \in \{T, T-1, \ldots, 1\}$, the teacher predicts all currently masked positions:

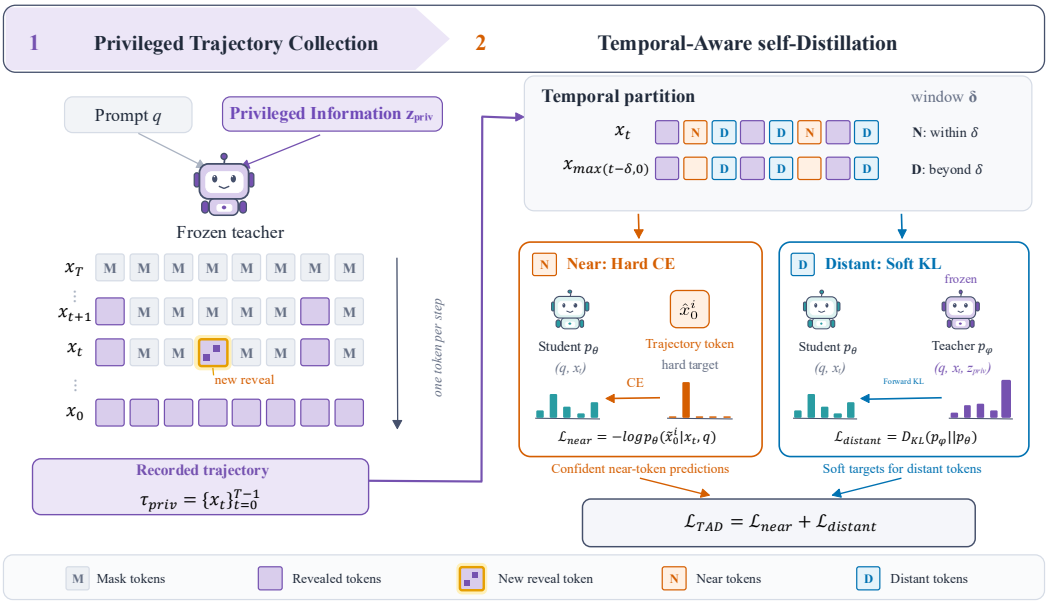


Figure 2: Overview of the TAD framework. **Left: Privileged-information-guided trajectory construction.** The teacher model p_ϕ receives prompt q together with privileged information and generates a token-by-token decoding trajectory. **Right: Temporal-aware self-distillation.** Masked tokens are divided into near and distant subsets according to the temporal window δ. The student model p_θ is trained with hard cross-entropy on near tokens and soft KL divergence on distant tokens.

$$p_\phi\left(x_0^i \mid \mathbf{x}_t, q, z_{\text{priv}}\right), \qquad i \in \mathcal{M}(\mathbf{x}_t). \tag{5}$$

We select the masked position with the highest prediction confidence:

$$m_t = \arg\max_{i \in \mathcal{M}(\mathbf{x}_t)} \max_v p_\phi\left(x_0^i = v \mid \mathbf{x}_t, q, z_{\text{priv}}\right), \tag{6}$$

where v denotes a token in the vocabulary. The corresponding token is selected as

$$\hat{x}^{m_t} = \arg\max_v p_\phi\left(x_0^{m_t} = v \mid \mathbf{x}_t, q, z_{\text{priv}}\right). \tag{7}$$

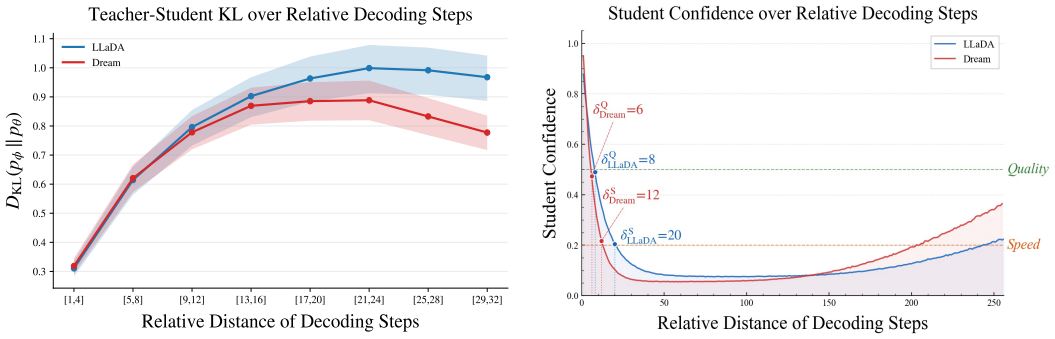
The next state $\mathbf{x}_{t-1}$ is obtained by replacing the mask at position m_t in $\mathbf{x}_t$ with $\hat{x}^{m_t}$ while keeping all other positions unchanged. Repeating this process gives the privileged decoding trajectory

$$\tau_{\text{priv}} = \{(\mathbf{x}_t, m_t, \hat{x}^{m_t})\}_{t=0}^{T-1}. \tag{8}$$

Unlike trajectories constructed only from the base model's own predictions, the privileged information affects the teacher's token predictions, reveal order, and intermediate states. The resulting trajectory τ_{priv} therefore records decoding states and token decisions shaped by the privileged information, which are then used for temporal-aware self-distillation.

3.2 Temporal-Aware Self-Distillation

Tokens masked in the same intermediate state can be revealed at very different future steps, and their prediction behavior varies systematically with this temporal distance. As shown in Figure 3a, the teacher–student KL divergence is small for tokens close to reveal but remains very large across distant intervals, suggesting that a single supervision type is not suitable for all masked tokens. We therefore partition tokens by temporal distance and apply different supervision to near and distant subsets.



(a) Teacher–student KL versus temporal distance.

(b) Student confidence versus temporal distance.

Figure 3: **Temporal diagnostics for temporal-aware supervision. (a)** Teacher–student KL over uniform four-step time-to-reveal intervals. The distributional gap is small for near tokens and remains high across distant intervals. **(b)** Student confidence falls over the early distances used for window selection; confidence thresholds of 0.5 and 0.2 define δ_{quality} and δ_{speed}, respectively. Together, these trends motivate hard CE for near tokens and soft KL for distant tokens.

At state $\mathbf{x}_t$, let $b_t^{(i)} \in \{0, 1\}$ indicate whether position i is masked, where $b_t^{(i)} = 1$ denotes a masked token. Given a temporal window δ, we define

$$\mathcal{M}_{\text{near}} = \left\{ i \mid b_t^{(i)} = 1 \wedge b_{\max(t-\delta,0)}^{(i)} = 0 \right\}, \tag{9}$$

$$\mathcal{M}_{\text{distant}} = \left\{ i \mid b_t^{(i)} = 1 \wedge b_{\max(t-\delta,0)}^{(i)} = 1 \right\}. \tag{10}$$

Thus, $\mathcal{M}_{\text{near}}$ contains masked tokens that will be revealed within the next δ reverse steps, while $\mathcal{M}_{\text{distant}}$ contains tokens that remain masked beyond this window.

Hard supervision for near tokens. For near tokens, we use the tokens recorded in the teacher trajectory as hard targets:

$$\mathcal{L}_{\text{near}} = -\mathbb{E}_{\mathbf{x}_t \sim \tau_{\text{priv}}} \left[\frac{1}{|\mathcal{M}_{\text{near}}|} \sum_{i \in \mathcal{M}_{\text{near}}} \log p_\theta \left(\hat{x}_0^i \mid \mathbf{x}_t, q \right) \right], \tag{11}$$

where $\hat{x}_0^i$ denotes the token for position i recorded in the teacher trajectory. This hard supervision increases the student's confidence on tokens that will be decoded soon, encouraging more tokens to be generated in each forward pass.

Soft supervision for distant tokens. For distant tokens, the hard trajectory target may depend on context that has not yet been revealed in $\mathbf{x}_t$. Directly forcing the student toward a one-hot target can therefore lead to an early decision with limited context. Let $p_\phi^j = p_\phi(x_0^j = \cdot \mid \mathbf{x}_t, q, z_{\text{priv}})$ and $p_\theta^j = p_\theta(x_0^j = \cdot \mid \mathbf{x}_t, q)$ denote the teacher and student token distributions at position j, respectively. We match these full distributions using

$$\mathcal{L}_{\text{distant}} = \mathbb{E}_{\mathbf{x}_t \sim \tau_{\text{priv}}} \left[\frac{1}{|\mathcal{M}_{\text{distant}}|} \sum_{j \in \mathcal{M}_{\text{distant}}} D_{\text{KL}} \left(p_\phi^j \parallel p_\theta^j \right) \right]. \tag{12}$$

This soft supervision preserves the teacher's uncertainty over distant tokens while still transferring useful information about future predictions. The final TAD objective combines the two losses:

$$\mathcal{L}_{\text{TAD}} = \mathcal{L}_{\text{near}} + \mathcal{L}_{\text{distant}}, \tag{13}$$

TAD therefore uses hard supervision to improve near-term prediction confidence and soft supervision to preserve information for distant tokens.

Choice of the Temporal Window δ. The temporal window δ controls the boundary between hard and soft supervision. We determine δ from the student's confidence profile over temporal distance. For each distance, we measure the average probability that the student assigns to the corresponding teacher trajectory token, and choose the distance at which this confidence falls below a predefined threshold. We use a threshold of 0.5 for the **Quality** setting and 0.2 for the **Speed** setting. As shown in Figure 3b, this gives $\delta_{\text{quality}} = 8/6$ and $\delta_{\text{speed}} = 20/12$ for LLaDA/Dream, respectively.

Table 1: Comparison of *TAD-LLaDA* with other LLaDA-based models. The best results among acceleration methods are highlighted in **bold** and the second best are underlined.

	GSM8K-CoT			MATH			MBPP			HumanEval			Average		
Method	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑
LLaDA	72.5	1.00	72.5	32.2	1.00	32.2	41.6	1.00	41.6	38.1	1.00	38.1	46.1	1.00	46.1
Fast-dLLM	73.0	2.81	180.8	31.9	2.24	50.4	38.7	2.12	56.5	37.9	2.57	54.2	45.4	2.43	85.5
D2F	73.3	2.75	178.4	29.0	2.66	47.2	39.1	2.16	53.4	40.5	2.62	61.0	45.5	2.55	85.0
dParallel	73.7	5.18	319.6	31.0	4.14	76.1	39.9	2.32	60.0	39.1	4.87	83.2	45.9	4.13	134.7
d3LLM	73.0	**9.42**	533.2	30.6	**5.32**	94.6	40.6	**4.16**	**87.6**	39.6	5.93	96.3	45.9	**6.21**	202.9
TAD-Q	**79.8**	6.31	502.5	**42.7**	4.49	190.4	**41.6**	3.35	79.0	**42.0**	**6.12**	**117.4**	**51.5**	5.07	222.3
TAD-S	78.8	8.46	**647.8**	40.6	5.14	**190.8**	40.6	3.44	80.6	39.6	5.91	107.2	49.9	5.74	**256.6**

Table 2: Comparison of *TAD-Dream* with other Dream-based models. The best results among acceleration methods are highlighted in **bold** and the second best are underlined.

	GSM8K-CoT			MATH			MBPP			HumanEval			Average		
Method	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑	Acc↑	TPF↑	AUP↑
Dream	83.9	1.00	83.9	39.7	1.00	39.7	57.1	1.00	57.1	55.1	1.00	55.1	58.9	1.00	58.9
Fast-dLLM	80.9	1.49	119.8	38.3	1.83	56.2	53.3	1.19	63.1	54.4	1.34	63.8	56.7	1.46	75.7
dParallel	**82.2**	2.96	239.2	38.9	2.94	77.9	55.3	2.21	106.6	54.3	2.51	97.3	57.6	2.66	130.2
d3LLM	81.1	**4.82**	**377.5**	38.1	3.92	97.2	55.6	**2.91**	**139.3**	57.3	**3.18**	128.7	58.0	3.70	185.6
TAD-Q	81.3	3.25	252.2	**42.9**	3.93	143.6	**56.8**	2.40	123.9	**64.0**	2.30	131.9	**61.2**	2.97	162.9
TAD-S	80.9	4.58	354.9	41.3	**5.30**	**161.3**	53.8	2.35	111.7	62.1	2.99	**147.7**	59.6	**3.81**	**193.9**

4 EXPERIMENT

4.1 EXPERIMENTAL SETUP

Training Data. We construct training data from public datasets containing prompts and reference answers. For LLaDA-8B-Instruct (Nie et al., 2025), we sample examples from the training splits of GSM8K (Cobbe et al., 2021), PRM12K (Lightman et al., 2023), NuminaMATH (LI et al., 2024) and a subset of KodCode (Xu et al., 2025b). Following Section 3.1, the reference answer is provided as privileged information when collecting teacher trajectories. We use a generation length of 256 and a block length of 32, and reveal one token at each decoding step while recording all intermediate states. After filtering invalid trajectories, we obtain approximately 26K training examples. We apply the same procedure to Dream-7B-Instruct (Ye et al., 2025) and obtain approximately 22K examples.

Training Details. We fine-tune all models using LoRA (Hu et al., 2022) with rank $r = 128$ and scaling factor $\alpha = 128$. To determine the temporal window δ, we randomly sample 500 trajectories from the collected trajectory data for each base architecture and follow the window-selection procedure described in Section 3.2. Specifically, we measure the student's confidence on teacher trajectory tokens as a function of their temporal distance to decoding, and select δ according to the confidence thresholds defined for the two operating modes. For TAD-Q, using a threshold of 0.5 gives $\delta = 8$ for LLaDA and $\delta = 6$ for Dream; for TAD-S, using a threshold of 0.2 gives $\delta = 20$ and $\delta = 12$, respectively. We fine-tune the models on the full collected training data using 8 NVIDIA DGX H200 GPUs. Additional implementation details are provided in Appendix C.

Evaluation Details. We evaluate our models on four widely used benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021b). Following prior work (Qian et al., 2026), we use 4-shot prompting for MATH and 3-shot prompting for LLaDA on MBPP, while all remaining evaluations are zero-shot. At inference time, we use a generation length of 256 and a block length of 32, entropy-based dynamic decoding with a threshold of 0.5, and the multi-block decoding strategy adopted by D2F (Wang et al., 2025a) and d3LLM (Qian et al., 2026). For each benchmark, we run the evaluation five times with different random seeds and report the average results. We report accuracy, tokens per forward (TPF), and Accuracy Under Parallelism (AUP) (Qian et al., 2026). Following d3LLM (Qian et al., 2026), for

Table 3: Ablation study on temporal-aware distillation objectives. "Hard CE" uses hard trajectory tokens as targets, while "Soft KL" matches the teacher's token distribution.

Objectives		MATH			HumanEval		
Near	Distant	TPF ↑	Acc ↑	AUP ↑	TPF ↑	Acc ↑	AUP ↑
Hard CE	None	4.39	40.3	<u>156.4</u>	5.47	34.9	82.5
Soft KL	Soft KL	2.08	<u>41.4</u>	90.0	2.90	35.4	55.4
Hard CE	Hard CE	**5.12**	35.7	147.9	**6.39**	<u>39.3</u>	<u>100.7</u>
Hard CE	**Soft KL**	<u>4.49</u>	**42.7**	**190.4**	<u>6.12</u>	**42.0**	**117.4**

each model and benchmark, we construct a two-point accuracy–parallelism curve using that model's accuracy at $\mathrm{TPF} = 1$ and its accuracy–TPF pair under the reported accelerated decoding setting, and compute AUP from these two points. More evaluation settings are provided in Appendix C.3.2.

Baselines. We compare against three groups of methods: (1) the original dLLMs, LLaDA-8B-Instruct (Nie et al., 2025) and Dream-7B-Instruct (Ye et al., 2025); (2) the training-free acceleration method Fast-dLLM (Wu et al., 2025b); and (3) training-based acceleration methods, including D2F (Wang et al., 2025a), dParallel (Chen et al., 2025), and d3LLM (Qian et al., 2026).

4.2 Main Results

TAD Improves Model Accuracy and Remains Competitive with RL Methods. As shown in Tables 1 and 2, TAD improves accuracy on both LLaDA and Dream. TAD-Q raises LLaDA's average accuracy from 46.1% to 51.5% and achieves 42.7% on MATH, compared with 31.0% for dParallel and 30.6% for d3LLM. On Dream, it reaches 61.2% average accuracy and 64.0% on HumanEval, outperforming both acceleration baselines. Notably, TAD is also competitive in accuracy with RL-based methods. For a controlled comparison with RL-enhanced dLLMs, we follow the d1 evaluation protocol on MATH500 and fix $\mathrm{TPF} = 2$. Figure 4 shows that TAD-Q reaches 36.4%, outperforming LLaDA (32.4%), d3LLM (Qian et al., 2026) (29.6%), LLaDA1.5 (Zhu et al., 2025) (33.2%), and d1 (Zhao et al., 2025) (36.0%). TAD-Q even retains +0.4% advantage over the RL-trained d1 model, demonstrating its competitiveness with RL-trained dLLMs.

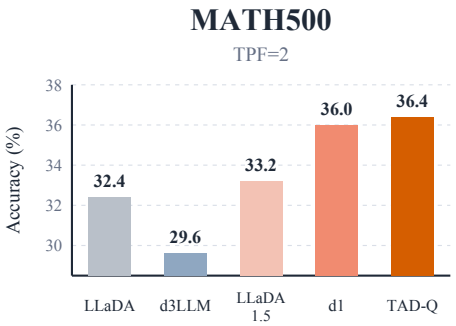


Figure 4: **MATH500 comparison at matched** $\mathrm{TPF} = 2$**.** Under the d1 protocol, TAD-Q improves over d1 by 0.4 points.

TAD Increases Parallelism and Achieves Better AUP. Beyond accuracy gains, TAD substantially increases decoding parallelism on both backbones. On LLaDA, TAD-Q and TAD-S achieve average TPF values of 5.07 and 5.74, respectively, compared with 4.13 for dParallel. Although d3LLM reaches a higher average TPF of 6.21, this comes with considerably lower accuracy (45.9% versus 49.9% for TAD-S). As a result, TAD-S achieves the highest average AUP of 256.6, clearly surpassing d3LLM (202.9) and dParallel (134.7). On Dream, TAD-S further increases the average TPF to 3.81, slightly exceeding d3LLM at 3.70, while also achieving the highest average AUP of 193.9 compared with 185.6 for d3LLM. Together, TAD-Q and TAD-S provide flexible operating points that favor either higher generation quality or more aggressive parallel decoding while consistently improving the overall accuracy–parallelism trade-off.

4.3 Ablation Study

We conduct ablations on LLaDA-8B-Instruct to isolate the contributions of trajectory construction and temporal-aware supervision.

Effect of Temporal-Aware Supervision. Table 3 compares different supervision strategies under a fixed temporal window of $\delta = 8$. Applying hard CE only to near tokens provides moderate parallelism,

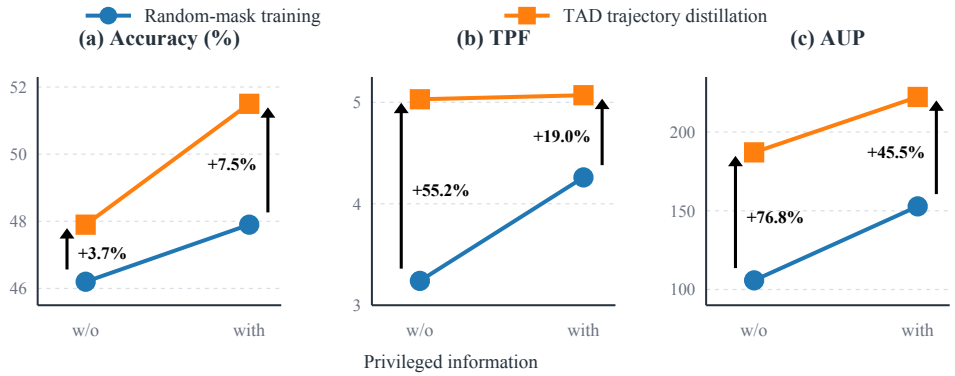


Figure 5: **Complementary effects of privileged-information-guided trajectory construction and temporal-aware distillation.** Each panel reports the average over four benchmarks on LLaDA-8B-Instruct. Privileged information improves both training strategies, while TAD provides the largest parallelism and AUP gains; combining them yields the best accuracy and AUP.

but leaving distant tokens unsupervised limits accuracy, especially on HumanEval (34.9%). Applying soft KL to all masked tokens preserves relatively strong MATH accuracy (41.4%), yet produces the lowest TPF and AUP on both benchmarks. Conversely, applying hard CE to both near and distant tokens achieves the highest TPF (5.12 on MATH and 6.39 on HumanEval), but its premature hard supervision reduces MATH accuracy to 35.7% and remains less accurate than our temporal-aware objective on HumanEval. Combining hard CE for near tokens with soft KL for distant tokens achieves the best accuracy and AUP on both benchmarks while retaining near-best TPF. These results validate that near and distant tokens require different forms of supervision to achieve a stronger accuracy–parallelism trade-off.

Effect of Privileged-Information-Guided Trajectories. Figure 5 ablates two factors: whether privileged information is used during trajectory collection, and whether the collected trajectories are trained with random masking or TAD. For a fair comparison, we collect the same number of trajectories with and without privileged information. Under random-mask supervision, privileged trajectories improve accuracy from 46.2% to 47.9% and AUP from 105.8 to 152.8, demonstrating the benefit of higher-quality trajectories. Without privileged information, replacing random masking with TAD increases TPF from 3.24 to 5.03 and AUP from 105.8 to 187.1, demonstrating the benefit of temporal-aware supervision. Combining both factors achieves the best result (51.5% accuracy, 5.07 TPF, and 222.3 AUP), confirming that they are complementary.

Effect of the Temporal Window δ. Figure 6 studies how the temporal window controls the quality–speed trade-off. A small window such as $\delta = 4$ limits hard supervision to tokens close to decoding and achieves the highest average accuracy of 52.1%. Increasing the window generally enables more aggressive parallel decoding: $\delta = 12$ reaches 5.17 TPF and 212.6 AUP, while $\delta = 20$ reaches 5.74 TPF and the highest AUP of 256.6 while retaining 49.9% accuracy. These results motivate using a smaller window for TAD-Q and a larger window for TAD-S.

5 Related Work

5.1 Diffusion Large Language Models

Recent research has extended diffusion modeling from continuous domains (Croitoru et al., 2023) to discrete text generation (Austin et al., 2021a; Sahoo et al., 2024; Lou et al., 2024; Shi et al., 2024). Unlike traditional autoregressive models that rely on left-to-right sequential generation (Achiam et al., 2023; Guo et al., 2025; Yang et al., 2025), Diffusion Large Language Models (dLLMs) feature bidirectional context attention and parallel decoding capabilities (Li et al., 2025). Recent models, including the LLaDA series (Nie et al., 2025; Zhu et al., 2025; Bie et al., 2025; 2026), Dream (Ye et al., 2025), and SDAR (Cheng et al., 2025), achieve performance competitive with leading autoregressive models across various benchmarks. They also demonstrate advantages in reverse reasoning tasks that

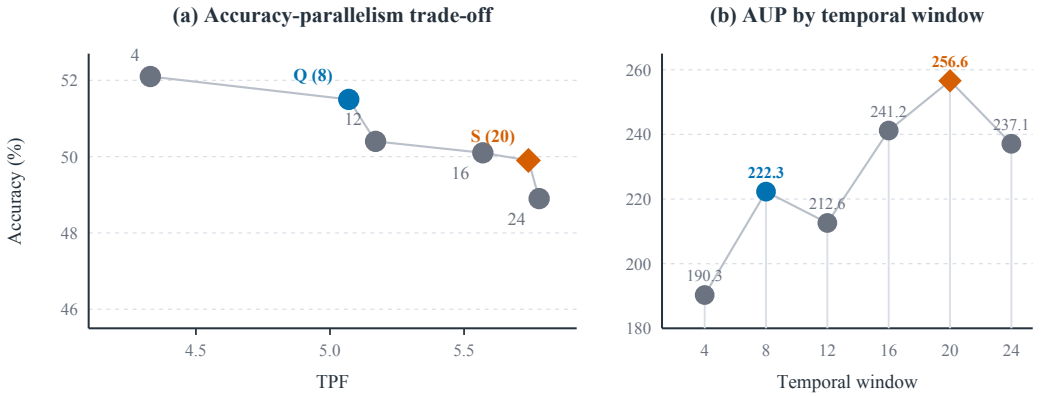


Figure 6: **Effect of the temporal window δ. Left:** Accuracy–parallelism operating points, where TAD-Q ($\delta = 8$) and TAD-S ($\delta = 20$) are highlighted. **Right:** AUP across temporal windows. A small window favors accuracy, while a larger window improves parallelism.

require global planning (Nie et al., 2025). Beyond these developments, the research community has increasingly focused on enhancing reasoning capabilities (Zhao et al., 2025; Wang et al., 2025b; Tang et al., 2025; Ou et al., 2025; Liu et al., 2026), building agent systems (Zhen et al., 2026; Zhao et al., 2026a), and accelerating inference (Lin et al., 2026) for dLLMs. In this paper, we focus on further accelerating dLLM inference by increasing the parallelism of these models.

5.2 Inference Acceleration for dLLMs

The inference cost of dLLMs arises from repeated forward evaluations and quality degradation when mutually dependent tokens are committed in parallel (Wu et al., 2025b; Kang et al., 2025). To reduce per-evaluation cost, recent studies (Liu et al., 2025; Ma et al., 2025; Jiang et al., 2025; Wu et al., 2025b)reuse temporally stable representations or restrict recomputation to selected regions (Liu et al., 2025; Ma et al., 2025; Jiang et al., 2025; Wu et al., 2025b). Block diffusion instead enable cache reuse explicitly across blocks (Arriola et al., 2025; Wu et al., 2025a). A complementary line of training-free methods (Hong et al., 2025; Wu et al., 2025b; Xu et al., 2025a; Mohamed et al., 2025; Peng et al., 2025; Ben-Hamu et al., 2026; Israel et al., 2026; Wei et al., 2026; Ye et al., 2026; Luo et al., 2026) reduces the number of evaluations by deciding where and how many tokens to reveal. These methods use confidence or lookahead signals (Wu et al., 2025b; Hong et al., 2025; Mohamed et al., 2025; Xu et al., 2025a), planner-guided unmasking (Peng et al., 2025), entropy-bounded selection (Ben-Hamu et al., 2026), or auxiliary autoregressive scoring (Israel et al., 2026). Although broadly compatible with pretrained dLLMs, these methods remain constrained by inference-time model predictions or auxiliary signals. Training-based approaches (Chen et al., 2025; Wang et al., 2025a; Kim et al., 2025; Qian et al., 2026; Zhang et al., 2026b; Liang et al., 2026; Bao et al., 2025; Hu et al., 2026; Asano et al., 2026) adapt the model or an auxiliary policy for aggressive parallel generation. TAD is most closely related to trajectory distillation and privileged-context self-distillation (Snell et al., 2022; Zhao et al., 2026b; Shenfeld et al., 2026; Hübotter et al., 2026). Unlike prompt-only trajectory construction, TAD conditions a frozen teacher on training-only targets, allowing privileged information to shape token predictions, reveal order, and intermediate states and further uses each token's time to reveal to apply hard supervision to near tokens and soft distribution matching to distant tokens.

6 Conclusion

We present TAD, a temporal-aware trajectory self-distillation framework that improves the accuracy-parallelism trade-off. The framework collects high-quality trajectories via a teacher conditioned on privileged information and partitions masked positions by their decoding steps, applying cross-entropy to near tokens for throughput and KL divergence loss to distant tokens for dependency preservation. Experiments on mathematical reasoning and code generation confirm that this design improves both accuracy and decoding speed, offering a practical path toward deploying efficient dLLMs.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing and for transcribing handwritten mathematical expressions into LaTeX. Generative AI was not used to formulate research ideas, mathematical claims or proofs, design experiments or methodology, implement methods, analyze data, or interpret experimental results. All AI-assisted edits and equation transcriptions were manually reviewed and verified by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our research does not involve human subjects, animal experiments, or the collection of sensitive personal data. The datasets used in this study are publicly available benchmarks that have been widely adopted in the research community and do not contain personally identifiable information. We have carefully considered the potential societal impacts of our method and believe that it does not pose direct risks of misuse or harmful applications. All computational experiments were conducted using institutional resources with appropriate permissions.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce the study throughout the paper and supplementary material. Section 3.1, Section 3.2, and Algorithm 1 specify the trajectory-construction procedure and training objective; Section 4 describes the datasets, model checkpoints, data construction, and evaluation protocol. Appendix C provides the exact prompt templates, hyperparameters, hardware, metric definition, and baseline decoding configurations, while Appendix D.1 reports means and standard deviations over five random seeds. The assumptions and proofs supporting the theoretical claims are given in Appendix A.1. Anonymous source code, including training and evaluation configurations and scripts, is available at https://anonymous.4open.science/r/TAD-B9D0/.

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 Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

Hikaru Asano, Tadashi Kozuno, Kuniaki Saito, and Yukino Baba. Where-to-unmask: Ground-truth-guided unmasking order learning for masked diffusion language models. *arXiv preprint arXiv:2602.09501*, 2026.

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.

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. *arXiv preprint arXiv:2509.25188*, 2025.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *Advances in Neural Information Processing Systems*, 38:55981–56007, 2026.

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.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. Dmax: Aggressive parallel decoding for dllms. *arXiv preprint arXiv:2604.08302*, 2026.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

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.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9): 10850–10869, 2023.

Guhao Feng, Yihan Geng, Jian Guan, Wei Wu, Liwei Wang, and Di He. Theoretical benefit and limitation of diffusion language model. *Advances in Neural Information Processing Systems*, 38: 24415–24459, 2026.

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.

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.

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

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

Yanzhe Hu, Yijie Jin, Pengfei Liu, Kai Yu, and Zhijie Deng. Lightningrl: Breaking the accuracy-parallelism trade-off of block-wise dllms via reinforcement learning. *arXiv preprint arXiv:2603.13319*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding. *Advances in neural information processing systems*, 38:52870–52888, 2026.

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, and Xu Yang. d̃2 cache: Accelerating diffusion-based llms via dual adaptive caching. *arXiv preprint arXiv:2509.23094*, 2025.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Il Koo, Nam Ik Cho, et al. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. *arXiv preprint arXiv:2510.04767*, 2025.

Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. Cdlm: Consistency diffusion language models for faster sampling. *arXiv preprint arXiv:2511.19269*, 2025.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

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

Yihao Liang, Ze Wang, Hao Chen, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Emad Barsoum, Zicheng Liu, and Niraj K Jha. Cd4lm: Consistency distillation and adaptive decoding for diffusion language models. *arXiv preprint arXiv:2601.02236*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Haokun Lin, Xinle Jia, Shaozhen Liu, Shujun Xia, Weitao Huang, Haobo Xu, Junyang Li, Yicheng Xiao, Xingrun Xing, Ziyu Guo, et al. Efficient diffusion language models: A comprehensive survey. *Authorea Preprints*, 2026.

Jiawei Liu, Xiting Wang, Yuanyuan Zhong, Defu Lian, and Yu Yang. Efficient and stable reinforcement learning for diffusion language models. *arXiv preprint arXiv:2602.08905*, 2026.

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. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32819–32848, 2024.

Xiangzhong Luo, Yilin An, Zhicheng Yu, Weichen Liu, and Xu Yang. Divide and conquer: Accelerating diffusion-based large language models via adaptive parallel decoding. *arXiv preprint arXiv:2602.23792*, 2026.

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

Amr Mohamed, Yang Zhang, Michalis Vazirgiannis, and Guokan Shang. Fast-decoding diffusion language models via progress-aware confidence schedules. *arXiv preprint arXiv:2512.02892*, 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.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled rl for diffusion llms emerges from a sequence-level perspective. *arXiv preprint arXiv:2512.03759*, 2025.

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling. *arXiv preprint arXiv:2502.03540*, 2025.

Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*, 2026.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*, 2025.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*, 2025a.

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

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles. In *International Conference on Learning Representations*, volume 2026, pp. 4340–4354, 2026.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025a.

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*, 2025b.

Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, et al. Lopa: Scaling dllm inference via lookahead parallel decoding. *arXiv preprint arXiv:2512.16229*, 2025a.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6980–7008, 2025b.

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.

Yushi Ye, Feng Hong, Huangjie Zheng, Xu Chen, Zhiyong Chen, Yanfeng Wang, and Jiangchao Yao. Rejection mixing: Fast semantic propagation of mask tokens for efficient dllm inference. *arXiv preprint arXiv:2602.22868*, 2026.

Shaorong Zhang, Longxuan Yu, Rob Brekelmans, Luhan Tang, Salman Asif, and Greg Ver Steeg. Generation order and parallel decoding in masked diffusion models: An information-theoretic perspective. *arXiv preprint arXiv:2602.00286*, 2026a.

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Vladimir Pavlovic, et al. T3d: Few-step diffusion language models via trajectory self-distillation with direct discriminative optimization. *arXiv preprint arXiv:2602.12262*, 2026b.

Jiahao Zhao, Shaoxuan Xu, Zhongxiang Sun, Fengqi Zhu, Jingyang Ou, Yuling Shi, Chongxuan Li, Xiao Zhang, and Jun Xu. Dllm-searcher: Adapting diffusion large language model for search agents. *arXiv preprint arXiv:2602.07035*, 2026a.

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

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026b.

Huiling Zhen, Weizhe Lin, Renxi Liu, Kai Han, Yiming Li, Yuchuan Tian, Hanting Chen, Xiaoguang Li, Xiaosong Li, Chen Chen, et al. Dllm agent: See farther, run faster. *arXiv preprint arXiv:2602.07451*, 2026.

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. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 7178–7218, 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.

A Proof of Theoretical Analysis

A.1 Privileged Information Raises the Distillation Capability Ceiling

We formalize two complementary effects of privileged information. First, making training-only information available enlarges the set of prediction rules accessible to the trajectory teacher and therefore cannot reduce the best achievable supervision quality. Second, the student inherits any gain realized by the privileged teacher up to its distillation error.

Let (q, a, z) follow the task distribution, where q is a prompt, a is the target response (or, more generally, the latent task state that determines correctness), and z is privileged information available only during trajectory construction. In our experiments, z contains the reference response a. Let $u(q, a, y) \in [0, 1]$ be a bounded utility for a model response y.

Let Π_0 denote the family of prompt-only prediction rules $\pi(\cdot \mid q)$ available to the trajectory generator, and let Π_{priv} denote the corresponding family $\pi(\cdot \mid q, z)$ when privileged information is available. Both families use the same model family and decoding mechanism. Because a privileged rule can always ignore z, every rule in Π_0 has an equivalent rule in Π_{priv}. Define their capability ceilings as

$$C_0^\star = \sup_{\pi \in \Pi_0} \mathbb{E}_{q,a} \mathbb{E}_{y \sim \pi(\cdot|q)}[u(q, a, y)], \tag{14}$$

$$C_{\mathrm{priv}}^\star = \sup_{\pi \in \Pi_{\mathrm{priv}}} \mathbb{E}_{q,a,z} \mathbb{E}_{y \sim \pi(\cdot|q,z)}[u(q, a, y)]. \tag{15}$$

Proposition A.1 (Capability-ceiling expansion)**.** *Access to privileged information cannot decrease the achievable capability ceiling:*

$$C_{\mathrm{priv}}^\star \geq C_0^\star. \tag{16}$$

The inequality is strict whenever the privileged information has positive task value within the prediction family, i.e., whenever some rule in Π_{priv} *attains expected utility strictly larger than* $C_0^\star$*.*

Proof. For any $\pi_0 \in \Pi_0$, construct a privileged rule $\widetilde{\pi}_0 \in \Pi_{\mathrm{priv}}$ by setting

$$\widetilde{\pi}_0(\cdot \mid q, z) = \pi_0(\cdot \mid q), \tag{17}$$

so that $\widetilde{\pi}_0$ ignores z. The two rules have identical expected utility. Hence the feasible prompt-only rules are embedded in the privileged family, and taking the supremum gives Eq. 16. If a privileged rule achieves utility greater than $C_0^\star$, taking the supremum over Π_{priv} makes the inequality strict. □

Proposition A.1 concerns the capability made available by the additional information. A particular frozen teacher must also exploit that information for the gain to be realized. To separate these two effects, define the prompt-only and privileged teacher distributions as

$$T_q^0 = p_\phi(\cdot \mid q), \qquad T_{q,z}^{\mathrm{priv}} = p_\phi(\cdot \mid q, z), \tag{18}$$

and define their expected utilities

$$C_0 = \mathbb{E}_{q,a} \mathbb{E}_{y \sim T_q^0}[u(q, a, y)], \tag{19}$$

$$C_{\mathrm{priv}} = \mathbb{E}_{q,a,z} \mathbb{E}_{y \sim T_{q,z}^{\mathrm{priv}}}[u(q, a, y)]. \tag{20}$$

Their realized privileged-information gain is

$$G_{\mathrm{priv}} = C_{\mathrm{priv}} - C_0. \tag{21}$$

We refer to C_0 and C_{priv} as the teacher-induced distillation ceilings: in the realizable, zero-distillation-error limit, they are the performances targeted by prompt-only and privileged distillation, respectively.

Let S_q^0 and S_q^{priv} be the student distributions obtained under the two strategies. The student never conditions on z. Define

$$J(S) = \mathbb{E}_{q,a,z} \mathbb{E}_{y \sim S_q}[u(q, a, y)] \tag{22}$$

and the corresponding distillation errors

$$\Delta_0 = \mathbb{E}_q \left[D_{\mathrm{KL}} \left(T_q^0 \parallel S_q^0 \right) \right], \tag{23}$$

$$\Delta_{\mathrm{priv}} = \mathbb{E}_{q,z} \left[D_{\mathrm{KL}} \left(T_{q,z}^{\mathrm{priv}} \parallel S_q^{\mathrm{priv}} \right) \right]. \tag{24}$$

Proposition A.2 (Transfer of the raised ceiling)**.** *The utility of each student differs from its teacher-induced distillation ceiling by at most its distillation error:*

$$\left|J(S^0) - C_0\right| \le \sqrt{\frac{\Delta_0}{2}}, \tag{25}$$

$$\left|J(S^{\mathrm{priv}}) - C_{\mathrm{priv}}\right| \le \sqrt{\frac{\Delta_{\mathrm{priv}}}{2}}. \tag{26}$$

Consequently, the privileged student satisfies the direct performance comparison

$$J(S^{\mathrm{priv}}) - J(S^0) \ge G_{\mathrm{priv}} - \sqrt{\frac{\Delta_{\mathrm{priv}}}{2}} - \sqrt{\frac{\Delta_0}{2}}. \tag{27}$$

Thus privileged trajectory construction guarantees strictly higher student task utility whenever

$$G_{\mathrm{priv}} > \sqrt{\frac{\Delta_{\mathrm{priv}}}{2}} + \sqrt{\frac{\Delta_0}{2}}. \tag{28}$$

In the realizable exact-distillation limit, $\Delta_0, \Delta_{\mathrm{priv}} \to 0$, the student's performance gain converges to G_{priv}.

Proof. For any two response distributions R and S and any utility bounded in $[0, 1]$,

$$|\mathbb{E}_{y\sim R}[u(q, a, y)] - \mathbb{E}_{y\sim S}[u(q, a, y)]| \le D_{\mathrm{TV}}(R, S). \tag{29}$$

Applying this inequality to the prompt-only teacher and student, then using Pinsker's inequality and Jensen's inequality, gives

$$\left|J(S^0) - C_0\right| \le \mathbb{E}_q D_{\mathrm{TV}}(T_q^0, S_q^0) \tag{30}$$

$$\le \mathbb{E}_q \sqrt{\frac{1}{2} D_{\mathrm{KL}}(T_q^0 \parallel S_q^0)} \tag{31}$$

$$\le \sqrt{\frac{\Delta_0}{2}}. \tag{32}$$

The same argument for the privileged teacher yields

$$\left|J(S^{\mathrm{priv}}) - C_{\mathrm{priv}}\right| \le \mathbb{E}_{q,z} D_{\mathrm{TV}}(T_{q,z}^{\mathrm{priv}}, S_q^{\mathrm{priv}}) \tag{33}$$

$$\le \sqrt{\frac{\Delta_{\mathrm{priv}}}{2}}. \tag{34}$$

Finally,

$$J(S^{\mathrm{priv}}) - J(S^0) = \big(J(S^{\mathrm{priv}}) - C_{\mathrm{priv}}\big) + G_{\mathrm{priv}} + \big(C_0 - J(S^0)\big) \tag{35}$$

$$\ge G_{\mathrm{priv}} - \sqrt{\frac{\Delta_{\mathrm{priv}}}{2}} - \sqrt{\frac{\Delta_0}{2}}, \tag{36}$$

which proves Eq. 27 and the strict-improvement condition in Eq. 28. When both distillation errors vanish, Eq. 25 gives $J(S^{\mathrm{priv}}) - J(S^0) = G_{\mathrm{priv}}$. □

Together, Propositions A.1 and A.2 distinguish capability availability from capability transfer. Privileged information first raises the best supervision ceiling whenever it has positive task value; when the frozen teacher realizes a positive gain G_{priv}, distillation transfers that gain to the student up to the two approximation terms. In our setting, the privileged information contains the reference response, and the teacher-accuracy results in Table 9 empirically establish a substantial positive G_{priv}.

B ALGORITHM

In this section, we provide the training procedure of TAD given the privileged decoding trajectories collected as described in Section 3.1. Algorithm 1 summarizes the temporal-aware self-distillation procedure used to optimize the student model.

Algorithm 1 Temporal-Aware Trajectory Self-Distillation (TAD)

Input: Pretrained base model p_{θ_0}, privileged trajectory dataset $\mathcal{D}_{\text{traj}} = \{(q, z_{\text{priv}}, \tau_{\text{priv}})\}$, temporal window δ.
Output: Optimized student model p_{θ^*}.

Initialize student: $\theta \leftarrow \theta_0$
Initialize teacher as a copy of the initial student: $\phi \leftarrow \theta_0$
Freeze teacher parameters ϕ
while not converged **do**
  Sample a batch of $(q, z_{\text{priv}}, \tau_{\text{priv}})$ from $\mathcal{D}_{\text{traj}}$
  Sample an intermediate state $\mathbf{x}_t$ from τ_{priv}, where $t \in \{T, \dots, 1\}$
  $t' \leftarrow \max(t - \delta, 0)$
  # 1. Temporal-Aware Partition
  Construct the near-token subset

$$\mathcal{M}_{\text{near}} \leftarrow \left\{ i \mid b_t^{(i)} = 1 \wedge b_{t'}^{(i)} = 0 \right\}$$

  Construct the distant-token subset

$$\mathcal{M}_{\text{distant}} \leftarrow \left\{ i \mid b_t^{(i)} = 1 \wedge b_{t'}^{(i)} = 1 \right\}$$

  Retrieve the trajectory token $\hat{x}_0^i$ for each $i \in \mathcal{M}_{\text{near}}$
  # 2. Student and Teacher Predictions
  Compute student distributions $p_\theta(x_0^i = \cdot \mid \mathbf{x}_t, q)$ for all masked positions
  Compute teacher distributions $p_\phi(x_0^j = \cdot \mid \mathbf{x}_t, q, z_{\text{priv}})$ for $j \in \mathcal{M}_{\text{distant}}$ {frozen teacher}
  # 3. Temporal-Aware Distillation

$$\mathcal{L}_{\text{near}} \leftarrow -\frac{1}{|\mathcal{M}_{\text{near}}|} \sum_{i \in \mathcal{M}_{\text{near}}} \log p_\theta \left(\hat{x}_0^i \mid \mathbf{x}_t, q \right)$$

$$\mathcal{L}_{\text{distant}} \leftarrow \frac{1}{|\mathcal{M}_{\text{distant}}|} \sum_{j \in \mathcal{M}_{\text{distant}}} D_{\text{KL}} \left(p_\phi \left(x_0^j = \cdot \mid \mathbf{x}_t, q, z_{\text{priv}} \right) \parallel p_\theta \left(x_0^j = \cdot \mid \mathbf{x}_t, q \right) \right)$$

  $\mathcal{L}_{\text{TAD}} \leftarrow \mathcal{L}_{\text{near}} + \mathcal{L}_{\text{distant}}$
  # 4. Student Optimization
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{TAD}}$
end while
return p_{θ^*}

C More Implementation Details

C.1 Prompt Templates

We use separate prompt templates for the teacher and student in TAD. Here, {q} denotes the problem and {a} denotes its reference answer. The teacher receives both the problem and the reference answer when constructing privileged trajectories, whereas the student receives only the problem.

```
Problem: {q}
Here is the reference answer: {a}.
After understanding the reference solution, please try to solve this
problem using your own approach.
Answer:
```

Listing 1: Teacher prompt for privileged trajectory construction.

```
Problem:{q}
, Answer:
```

Listing 2: Student prompt for TAD.

C.2 Training Details

We apply TAD to two dLLMs: LLaDA-8B-Instruct and Dream-7B-Instruct. Both models are fine-tuned using LoRA with DeepSpeed ZeRO-Stage 2 on 8×H200 GPUs, using bfloat16 mixed precision throughout. The total training time is 20 hours for each model. The detailed training hyperparameters are summarized in Table 4.

Table 4: Training hyperparameters for TAD.

Hyperparameter	**TAD-LLaDA**	**TAD-Dream**
Training		
Optimizer	AdamW	AdamW
Learning Rate	2×10^{-5}	2×10^{-5}
LR Scheduler	Constant	Constant
Warmup Steps	0	0
Weight Decay	0.01	0.01
Max Gradient Norm	1.0	1.0
Training Epochs	1	1
Per-Device Batch Size	4	4
Gradient Accumulation Steps	8	8
Global Batch Size	256	256
Precision	bfloat16	bfloat16
DeepSpeed Stage	ZeRO-2	ZeRO-2
Max Sequence Length	Dynamic	Dynamic
LoRA		
Rank (r)	128	128
Alpha (α)	128	128
Dropout	0.05	0.05
Bias	None	None
Target Modules	q, k, v, o, gate, up, down_proj	
Distillation		
Temperature (τ)	1.0	1.0

C.3 Evaluation Details

C.3.1 Evaluation Metrics

We adopt the AUP (Accuracy Under Parallelism) Qian et al. (2026) score as our primary evaluation metric. AUP is defined as a weighted area under the accuracy–parallelism curve, where parallelism is measured by Tokens Per Forward pass (TPF). Given a set of parallelism-accuracy pairs $S = \{(\rho_i, y_i)\}_{i=1}^{m}$ sorted by increasing parallelism $\rho_1 < \rho_2 < \cdots < \rho_m$, AUP is computed as:

$$\text{AUP} \triangleq \rho_1 y_1 + \sum_{i=2}^{m} (\rho_i - \rho_{i-1}) \left(\frac{y_i W(y_i) + y_{i-1} W(y_{i-1})}{2} \right), \tag{37}$$

where the weighting function $W(y) = \min(e^{-\alpha(1-y/y_{\max})}, 1)$ penalizes accuracy degradation relative to the best accuracy $y_{\max}$ achieved on that task. We use $\alpha = 3$ as the default penalty factor. A minimum accuracy threshold $y_{\min} = y_1 - 5$ is applied to exclude regimes of severe accuracy degradation. AUP rewards methods that increase parallelism without sacrificing accuracy, while suppressing contributions from low-accuracy regimes. Importantly, AUP is hardware-independent since it relies on TPF rather than tokens per second (TPS), providing a fair comparison of algorithmic parallelism across different hardware setups.

C.3.2 Evaluation Configurations

For fair comparison, we follow the decoding configurations used by each baseline method. Table 5 summarizes the detailed inference settings. Unless otherwise specified, all methods are evaluated

Table 5: Decoding configurations used for different methods during evaluation.

Method	Decoding Configuration
Fast-dLLM	Confidence threshold = 0.9; dual cache.
dParallel	Entropy threshold = 0.5.
D2F	Confidence threshold = 0.9; block_add_threshold= 0.7; decoded_token_threshold= 0.95.
d3LLM	entropy_threshold= 0.5; block_add_threshold= 0.1; decoded_token_threshold= 0.95; multi_block_kv_cache;
TAD (all)	entropy_threshold= 0.5; block_add_threshold= 0.1; decoded_token_threshold= 0.95; no KV cache.
Common Settings	`lm-evaluation-harness`; generation length = 256; block size = 32; five independent runs (mean ± std).

with `lm-evaluation-harness`, a generation length of 256, a block size of 32. Each evaluation is repeated five times with different random seeds, and we report the mean and standard deviation.

D More Experiment Results

D.1 Results Across Random Seeds

Tables 6 and 7 supplement the main results in Tables 1 and 2, respectively. For every method and benchmark, we report the mean and standard deviation over five independent random seeds. The mean values are identical to those reported in the main text. The Average columns are computed from the benchmark-level means and are therefore shown without standard deviations.

Table 6: Five-seed results for *TAD-LLaDA* and LLaDA-based baselines. Per-benchmark entries are reported as mean ± standard deviation. The best results among acceleration methods are highlighted in **bold** and the second best are underlined.

(a) Accuracy (%)

Method	GSM8K-CoT	MATH	MBPP	HumanEval	Average
LLaDA	72.5±0.3	32.2±0.3	41.6±0.3	38.1±0.2	46.1
Fast-dLLM	73.0±0.3	31.9±0.3	38.7±0.2	37.9±0.2	45.4
D2F	73.3±0.2	29.0±0.3	39.1±0.3	40.5±0.1	45.5
dParallel	73.7±0.1	31.0±0.3	39.9±0.1	39.1±0.2	45.9
d3LLM	73.0±0.2	30.6±0.2	40.6±0.2	39.6±0.3	45.9
TAD-Q	**79.8±0.2**	**42.7±0.1**	**41.6±0.2**	**42.0±0.2**	**51.5**
TAD-S	78.8±0.2	40.6±0.1	40.6±0.2	39.6±0.2	49.9

(b) Tokens Per Forward (TPF)

Method	GSM8K-CoT	MATH	MBPP	HumanEval	Average
LLaDA	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00
Fast-dLLM	2.81±0.08	2.24±0.09	2.12±0.08	2.57±0.07	2.43
D2F	2.75±0.09	2.66±0.09	2.16±0.09	2.62±0.07	2.55
dParallel	5.18±0.07	4.14±0.11	2.32±0.09	4.87±0.08	4.13
d3LLM	**9.42±0.07**	**5.32±0.10**	**4.16±0.07**	5.93±0.12	**6.21**
TAD-Q	6.31±0.06	4.49±0.10	3.35±0.07	**6.12±0.09**	5.07
TAD-S	8.46±0.11	5.14±0.12	3.44±0.07	5.91±0.11	5.74

(c) Accuracy Under Parallelism (AUP)

Method	GSM8K-CoT	MATH	MBPP	HumanEval	Average
LLaDA	72.5±0.3	32.2±0.3	41.6±0.3	38.1±0.2	46.1
Fast-dLLM	180.8±5.4	50.4±1.4	56.5±1.1	54.2±0.8	85.5
D2F	178.4±4.9	47.2±0.7	53.4±1.0	61.0±0.9	85.0
dParallel	319.6±4.3	76.1±2.2	60.0±1.2	83.2±0.7	134.7
d3LLM	533.2±4.9	94.6±1.5	**87.6±1.3**	96.3±1.6	202.9
TAD-Q	502.5±6.8	190.4±3.5	79.0±1.3	**117.4±1.2**	222.3
TAD-S	**647.8±7.9**	**190.8±3.6**	80.6±1.0	107.2±1.6	**256.6**

Table 7: Five-seed results for *TAD-Dream* and Dream-based baselines. Per-benchmark entries are reported as mean ± standard deviation. The best results among acceleration methods are highlighted in **bold** and the second best are underlined.

(a) Accuracy (%)

Method	GSM8K-CoT	MATH	MBPP	HumanEval	Average
Dream	83.9±0.3	39.7±0.3	57.1±0.3	55.1±0.2	58.9
Fast-dLLM	80.9±0.3	38.3±0.3	53.3±0.2	54.4±0.2	56.7
dParallel	**82.2±0.1**	38.9±0.3	55.3±0.1	54.3±0.2	57.6
d3LLM	81.1±0.2	38.1±0.2	55.6±0.2	57.3±0.3	58.0
TAD-Q	81.3±0.2	**42.9±0.1**	**56.8±0.2**	**64.0±0.2**	**61.2**
TAD-S	80.9±0.2	41.3±0.1	53.8±0.2	62.1±0.2	59.6

(b) Tokens Per Forward (TPF)

Method	GSM8K-CoT	MATH	MBPP	HumanEval	Average
Dream	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00
Fast-dLLM	1.49±0.08	1.83±0.09	1.19±0.08	1.34±0.07	1.46
dParallel	2.96±0.07	2.94±0.11	2.21±0.09	2.51±0.08	2.66
d3LLM	**4.82±0.07**	3.92±0.10	**2.91±0.07**	**3.18±0.12**	3.70
TAD-Q	3.25±0.11	3.93±0.12	2.40±0.07	2.30±0.11	2.97
TAD-S	4.58±0.06	**5.30±0.10**	2.35±0.07	2.99±0.09	**3.81**

(c) Accuracy Under Parallelism (AUP)

Method	GSM8K-CoT	MATH	MBPP	HumanEval	Average
Dream	83.9±0.3	39.7±0.3	57.1±0.3	55.1±0.2	58.9
Fast-dLLM	119.8±6.2	56.2±1.8	63.1±2.8	63.8±2.0	75.7
dParallel	239.2±5.7	77.9±2.7	106.6±3.6	97.3±1.9	130.2
d3LLM	**377.5±5.6**	97.2±2.0	**139.3±3.4**	128.7±3.9	185.6
TAD-Q	252.2±7.9	143.6±3.5	123.9±3.1	131.9±5.8	162.9
TAD-S	354.9±5.8	**161.3±2.3**	111.7±3.0	**147.7±3.7**	**193.9**

D.2 Accuracy–Parallelism Operating Points

Figure 7 compares three measured accuracy–parallelism operating points for dParallel, d3LLM, TAD-Q, and TAD-S on each LLaDA benchmark. The gains of TAD are most pronounced on the mathematical reasoning tasks, where both TAD variants preserve substantially higher accuracy as parallelism increases. On the code benchmarks, TAD-Q favors accuracy, while TAD-S provides a more aggressive operating point, illustrating the complementary quality- and speed-oriented behavior of the two configurations.

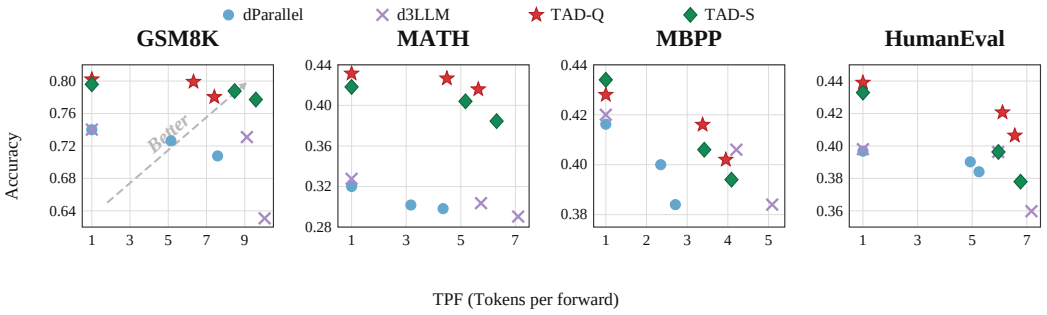


Figure 7: **Accuracy–parallelism operating points on LLaDA-8B-Instruct.** Each method is evaluated at three decoding operating points on GSM8K-CoT, MATH, MBPP, and HumanEval. Darker, larger markers denote our TAD variants, while lighter markers denote the distillation baselines.

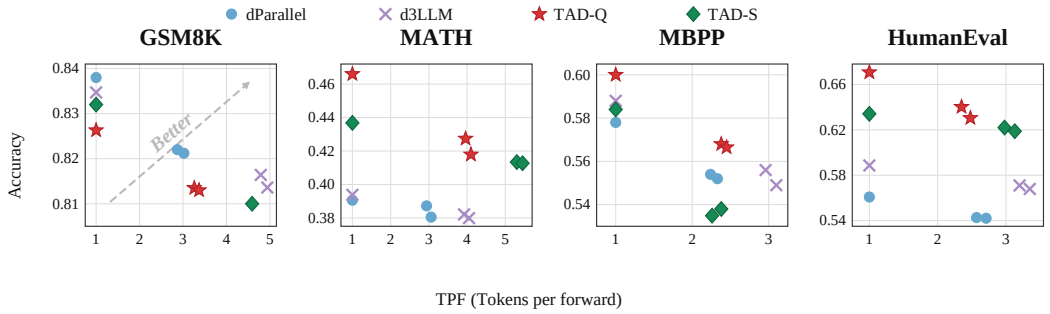


Figure 8: **Accuracy–parallelism operating points on Dream-7B-Instruct.** The four panels show the provided decoding operating points on GSM8K-CoT, MATH, MBPP-Instruct, and HumanEval-Instruct. Darker, larger markers denote our TAD variants, while lighter markers denote the distillation baselines.

Figure 8 presents the corresponding comparison on Dream-7B-Instruct. TAD yields its clearest gains on MATH and HumanEval-Instruct, where both variants maintain substantially higher accuracy as TPF increases. TAD-Q also favors accuracy on MBPP-Instruct, whereas d3LLM reaches a more aggressive parallel operating point. On GSM8K-CoT, the four methods exhibit a closer accuracy–parallelism trade-off.

D.3 Matched-Data Baseline Comparison

Experimental protocol. The results in Tables 1 and 2 use the publicly available checkpoints of dParallel and d3LLM, whose original training data differ from the data used by TAD. To control for this difference, we retrain both baselines on exactly the same examples used to train TAD. The same prompt mixture, reference-derived trajectory corpus, LoRA configuration, number of training epochs, and total optimization budget are used across the matched-data models.

Each baseline retains its original method-specific training procedure. In particular, dParallel applies its certainty-forcing objective to responses from the shared corpus, whereas d3LLM constructs its training states using the recorded teacher reveal order and applies its original curriculum-based trajectory-distillation objective. TAD applies the temporal-aware hard–soft objective described in Section 3.2. The student model receives only the problem prompt at inference time in all cases. We evaluate every model under the same benchmark prompts and generation settings used in the main experiments.

D.4 Impact of Privileged Information on Trajectory Collection

In this section, we investigate the quality of the intermediate trajectories collected during the data construction phase. Specifically, we evaluate the generative accuracy of the teacher models under two distinct rollout conditions: (1) standard self-generation, where the model is conditioned solely on the input question without access to the ground truth (w/o GT), and (2) generation with privileged information, where the model receives both the input question and the ground-truth response.

As presented in Table 9, the inclusion of ground-truth information yields a substantial and consistent improvement in trajectory accuracy across all evaluated benchmarks for both LLaDA-8B-Instruct and Dream-7B-Instruct architectures. For instance, on the GSM8K Training Set, the privileged information guidance improves LLaDA's accuracy from 68.88% to 89.16%, and Dream's from 75.17% to 94.79%. This performance confirms that leveraging ground truth as privileged information is beneficial to sample the high-quality intermediate states required for effective temporal-aware self-distillation.

Table 8: **Comparison with baselines retrained on the same data as TAD.** "Public" denotes the existing baseline checkpoints reported in Tables 1 and 2. "Matched" denotes models retrained using exactly the same training examples and the same optimization budget as TAD, while retaining each baseline's original training objective. TAD results are included for reference. All methods are evaluated using the same prompts, generation length, and block length. The detailed decoding protocols are shown in C.3.2.

		GSM8K-CoT		MATH		MBPP		HumanEval		Average	
Method	**Training data**	Acc↑	TPF↑	Acc↑	TPF↑	Acc↑	TPF↑	Acc↑	TPF↑	Acc↑	TPF↑
LLaDA-8B-Instruct backbone											
dParallel	Public	73.7	5.18	31.0	4.14	39.9	2.32	39.1	4.87	45.9	4.13
dParallel	Matched	76.9	5.21	36.3	4.52	40.6	2.79	38.0	5.63	48.0	4.54
d3LLM	Public	73.0	9.42	30.6	5.32	40.6	4.16	39.6	5.93	45.9	6.21
d3LLM	Matched	77.2	**8.81**	36.5	**5.22**	41.0	**4.47**	39.8	**6.33**	48.6	**6.21**
TAD-Q	Matched	**79.8**	6.31	**42.7**	4.49	**41.6**	3.35	**42.0**	6.12	**51.5**	5.07
TAD-S	Matched	78.8	8.46	40.6	5.14	40.6	3.44	39.6	5.91	49.9	5.74
Dream-7B-Instruct backbone											
dParallel	Public	82.2	2.96	38.9	2.94	55.3	2.21	54.3	2.51	57.6	2.66
dParallel	Matched	81.7	2.84	41.0	3.11	55.5	2.04	57.8	2.82	59.0	2.70
d3LLM	Public	81.1	4.82	38.1	3.92	55.6	2.91	57.3	3.18	58.0	3.70
d3LLM	Matched	80.7	**4.65**	39.8	4.13	56.0	**3.06**	61.5	**3.18**	59.5	3.76
TAD-Q	Matched	**81.3**	3.25	**42.9**	3.93	**56.8**	2.40	**64.0**	2.30	**61.2**	2.97
TAD-S	Matched	80.9	4.58	41.3	**5.30**	53.8	2.35	62.1	2.99	59.6	**3.81**

Table 9: Performance comparison of dLLMs with and without Ground Truth (GT).

Model	**Condition**	**GSM8K**	**KodCode**
LLaDA-8B-Instruct	w/o GT	68.88	8.22
	with GT	**89.16**	**77.28**
Dream-7B-Instruct	w/o GT	75.17	14.05
	with GT	**94.79**	**75.08**

D.5 Throughput Analysis

To further validate the practical deployment efficiency of the TAD framework, we conduct a wall-clock throughput analysis on the GSM8K-CoT benchmark using NVIDIA H200 GPUs. We measure the generation speed in tokens per second (TPS) and compare it against the base models and strong acceleration baselines. The results for LLaDA and Dream architectures are presented in Table 10.

Results on LLaDA Architecture As shown in Table 10, both TAD variants substantially accelerate LLaDA while improving its generation accuracy. TAD-LLaDA-Q reaches 336.4 TPS, corresponding to a $10.8\times$ speedup over LLaDA, and achieves the highest accuracy of 79.8%. TAD-LLaDA-S further increases throughput to 376.9 TPS ($12.1\times$) while retaining 78.8% accuracy. Although d3LLM attains the highest raw throughput at 401.2 TPS, TAD-LLaDA-S preserves 93.9% of this throughput while improving accuracy by 5.8 percentage points (78.8% vs. 73.0%). Together with the TPF results in Table 1, where d3LLM exhibits higher algorithmic parallelism than TAD-LLaDA-S (9.42 vs. 8.46 TPF), the wall-clock measurements show a consistent throughput ordering and highlight TAD's more favorable accuracy–efficiency trade-off.

Results on Dream Architecture On Dream, TAD-Dream-Q achieves 205.7 TPS ($5.1\times$ over the base model) with 81.3% accuracy, while TAD-Dream-S reaches 288.4 TPS ($7.2\times$) with 80.9% accuracy. d3LLM Qian et al. (2026) provides the highest throughput at 295.0 TPS, only 2.3% above TAD-Dream-S, consistent with its higher TPF (4.82 vs. 4.58). Thus, TAD-Dream-S delivers throughput

close to the strongest acceleration baseline, whereas TAD-Dream-Q offers a more conservative operating point that retains accuracy while still providing a substantial wall-clock speedup.

Table 10: Throughput comparison of *TAD-LLaDA* and *TAD-Dream* on GSM8K-CoT using H200 GPUs. We report tokens per second (TPS) and accuracy (%). Speedup ratios relative to the respective base models (LLaDA and Dream) are shown in parentheses.

Method	H200 TPS ↑	Acc (%) ↑
LLaDA	31.2 (1.0×)	72.5
Fast-dLLM-LLaDA	137.3 (4.4×)	73.0
D2F-LLaDA	121.1 (3.9×)	73.3
dParallel-LLaDA	214.3 (6.9×)	73.7
d3LLM-LLaDA	**401.2** (12.9×)	73.0
TAD-LLaDA-Q	336.4 (10.8×)	**79.8**
TAD-LLaDA-S	376.9 (12.1×)	78.8

Method	H200 TPS ↑	Acc (%) ↑
Dream	40.1 (1.0×)	83.9
Fast-dLLM-Dream	117.8 (2.9×)	80.9
Fast-dLLM-v2	169.3 (4.2×)	77.5
dParallel-Dream	189.4 (4.7×)	82.2
d3LLM-Dream	**295.0** (7.4×)	81.1
TAD-Dream-Q	205.7 (5.1×)	81.3
TAD-Dream-S	288.4 (7.2×)	80.9

D.6 Case Studies

We present three representative examples from MATH500 for which LLaDA-8B-Instruct and d3LLM produce incorrect answers, whereas TAD-Q reaches the correct result. All methods use a generation length of 256. LLaDA uses token-by-token decoding, while d3LLM and TAD-Q use multi-block decoding with a confidence threshold of 0.5. NFE denotes the number of forward evaluations required to complete the response. Incorrect final answers are shown in red and correct final answers in green; NFE is highlighted in red for ease of comparison.

D.6.1 Circular Seating (MATH500 id 32)

Model: LLaDA-8B-Instruct
Generation Length: 256
Question: In how many ways can 8 people sit around a round table if Pierre, Rosa, and Thomas all want to sit together? Two seatings are considered the same if one is a rotation of the other.

1. Treat Pierre, Rosa, and Thomas as a single "block" or "super person" since they want to sit together. This reduces the problem to arranging seven "people" around a round table.
2. In a circular arrangement, one position can be fixed to avoid counting rotations as different arrangements. Therefore, we can fix the "super person" in one position.
3. We then have six remaining people to arrange in the remaining six positions. The number of ways to arrange them is 6!.
4. Within the "super person," Pierre, Rosa, and Thomas can be arranged in 3! ways.
5. Therefore, the total number of arrangements is $6! \times 3!$.

Calculating this gives

$$6! = 720, \qquad 3! = 6, \qquad 720 \times 6 = 4320.$$

Thus, the answer is $\boxed{4320}$.

NFE: 256

Model: d3LLM-LLaDA
Generation Length: 256
Question: In how many ways can 8 people sit around a round table if Pierre, Rosa, and Thomas all want to sit together? Two seatings are considered the same if one is a rotation of the other.

1. First, treat Pierre, Rosa, and Thomas as a single "block" or "super person" since they must sit together. This reduces the problem to arranging seven "people" around a round table.
2. The number of ways to arrange seven "people" around a round table is $(7-1)! = 6!$.
3. Within the block, Pierre, Rosa, and Thomas can be arranged in $3!$ ways.
4. Therefore, the total number of arrangements is $6! \times 3!$.
5. Calculating the values gives

$$6! = 720, \qquad 3! = 6, \qquad 6! \times 3! = 720 \times 6 = 4320.$$

Thus, the number of ways is $\boxed{4320}$.

NFE: 53

Model: TAD-Q
Generation Length: 256
Question: In how many ways can 8 people sit around a round table if Pierre, Rosa, and Thomas all want to sit together? Two seatings are considered the same if one is a rotation of the other.

To solve this problem, we need to consider the constraints of the round table and the requirement that Pierre, Rosa, and Thomas must sit together.

1. **Treat Pierre, Rosa, and Thomas as a single unit.** Since they must sit together, we can consider them as one unit. This reduces the problem to arranging six units: the unit containing Pierre, Rosa, and Thomas, together with the other five people.
2. **Arrange the six units around a round table.** The number of arrangements is

$$(6-1)! = 5! = 120$$

because rotations are considered identical.
3. **Arrange Pierre, Rosa, and Thomas within their unit.** There are $3! = 6$ possible internal arrangements.
4. **Combine the arrangements.** Multiplying the two counts gives

$$120 \times 6 = 720.$$

Thus, the answer is $\boxed{720}$.

NFE: 48

D.6.2 FIBONACCI SERIES (MATH500 ID 213)

Model: LLaDA-8B-Instruct
Generation Length: 256
Question: Compute $\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}}$, where F_n denotes the nth Fibonacci number, $F_0 = 0$, and $F_1 = 1$.

To solve the problem, we start by analyzing the given series:

$$\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}}.$$

Let the series be denoted by S. To simplify the problem, we use properties of Fibonacci numbers and their ratios. Specifically, we use the claimed result

$$\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}} = \frac{1}{2}.$$

This result is stated to follow from properties of Fibonacci numbers and their ratios. Therefore, the final answer is reported as $\boxed{1}$.

NFE: 256

Model: d3LLM-LLaDA
Generation Length: 256
Question: Compute $\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}}$, where F_n denotes the nth Fibonacci number, $F_0 = 0$, and $F_1 = 1$.

To compute the sum, we begin by examining the general term

$$\frac{F_{n+1}}{F_n F_{n+2}}.$$

Using the Fibonacci recurrence $F_{n+2} = F_{n+1} + F_n$, the term is rewritten as

$$\frac{F_{n+1}}{F_n F_{n+2}} = \frac{F_{n+1}}{F_n (F_{n+1} + F_n)} = \frac{F_{n+1}}{F_n F_{n+1}} + \frac{1}{F_n F_{n+1}}.$$

Next, we consider the convergence of the series. The following is a well-known identity for Fibonacci numbers:

$$\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}} = \frac{1}{2}.$$

Therefore, the generated answer is $\boxed{\frac{1}{2}}$.

NFE: 54

Model: TAD-Q
Generation Length: 256
Question: Compute $\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}}$, where F_n denotes the nth Fibonacci number, $F_0 = 0$, and $F_1 = 1$.

To solve the problem, we need to compute the infinite series

$$\sum_{n=1}^{\infty} \frac{F_{n+1}}{F_n F_{n+2}}.$$

Using the Fibonacci recurrence, we have $F_{n+1} = F_{n+2} - F_n$. Therefore, the general term can be written as

$$\frac{F_{n+1}}{F_n F_{n+2}} = \frac{F_{n+2} - F_n}{F_n F_{n+2}} = \frac{1}{F_n} - \frac{1}{F_{n+2}}.$$

Hence, the partial sum telescopes:

$$\sum_{n=1}^{N} \left(\frac{1}{F_n} - \frac{1}{F_{n+2}} \right) = \frac{1}{F_1} + \frac{1}{F_2} - \frac{1}{F_{N+1}} - \frac{1}{F_{N+2}}.$$

Since $F_1 = F_2 = 1$ and $F_n \to \infty$, taking $N \to \infty$ gives

$$\frac{1}{F_1} + \frac{1}{F_2} = 1 + 1 = 2.$$

Therefore, the sum is $\boxed{2}$.

NFE: 51

D.6.3 Cube Roots of Unity (MATH500 ID 480)

Model: LLaDA-8B-Instruct
Generation Length: 256
Question: Let ω be a complex number such that $\omega^3 = 1$. Find all possible values of $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$.

Given that ω is a complex number such that $\omega^3 = 1$, we know that ω is a primitive cube root of unity. The primitive cube roots of unity are $\omega = e^{2\pi i/3}$ and $\omega^2 = e^{-2\pi i/3}$.
We need to find the value of the expression $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$.
First, recall that for any primitive cube root of unity ω, the expression $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$ is known to be equal to -1. This is a well-known result from the theory of complex roots of unity.
Thus, the value of $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$ is -1.
Therefore, the only possible value of the expression $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$ is $\boxed{-1}$.

NFE: 256

Model: d3LLM-LLaDA
Generation Length: 256
Question: Let ω be a complex number such that $\omega^3 = 1$. Find all possible values of $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$.

Given that ω is a complex number such that $\omega^3 = 1$, we need to find all possible values of the expression $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$.

First, recall that ω is a primitive cube root of unity. Therefore, $\omega^3 = 1$ and $\omega + \omega + \omega^2 = 0$.

We start by considering the expression:

$$\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$$

We can rewrite this expression using the fact that $\omega + \omega + \omega^2 = 0$:

$$\frac{1}{1+\omega} + \frac{1}{1+\omega^2} = \frac{1}{\omega} + \frac{1}{\omega^2}$$

Since $\omega^3 = 1$, we have:

$$\frac{1}{\omega} + \frac{1}{\omega^2} = 0$$

Thus, the possible value of the expression is $\boxed{0}$.

NFE: 49

Model: TAD-Q
Generation Length: 256
Question: Let ω be a complex number such that $\omega^3 = 1$. Find all possible values of $\frac{1}{1+\omega} + \frac{1}{1+\omega^2}$.

To solve the problem, we need to find the value of the expression under the condition $\omega^3 = 1$. First, rewrite the expression using a common denominator:

$$\frac{1}{1+\omega} + \frac{1}{1+\omega^2} = \frac{1+\omega^2+1+\omega}{(1+\omega)(1+\omega^2)}.$$

The numerator is $2 + \omega + \omega^2$. Using $\omega^3 = 1$, the denominator expands to

$$(1+\omega)(1+\omega^2) = 1 + \omega + \omega^2 + \omega^3 = 2 + \omega + \omega^2.$$

No cube root of unity equals -1, so the original denominators are nonzero. Thus,

$$\frac{1}{1+\omega} + \frac{1}{1+\omega^2} = \frac{2+\omega+\omega^2}{2+\omega+\omega^2} = 1.$$

Therefore, the final answer is $\boxed{1}$.

NFE: 44