Few-Step Diffusion Language Models via Trajectory Self-Distillation

Tunyu Zhang [*†1] **Xinxi Zhang** [*1] **Ligong Han** [†2 3] **Haizhou Shi** [1] **Xiaoxiao He** [1]

Zhuowei Li [1] **Hao Wang** [2 3] **Kai Xu** [2 3] **Akash Srivastava** [2 3] **Chengzhi Mao** [1]

Hao Wang [1] **Vladimir Pavlovic** [1] **Dimitris N. Metaxas** [†1]

Abstract

Diffusion large language models (DLLMs) have emerged as powerful generative models with the promise of fast text generation through parallel decoding. However, realizing this potential in practice remains challenging: reducing the number of decoding steps, typically causes a substantial degradation in output quality due to token factorization error. To alleviate this, we propose a self-distillation framework that trains a few-step student to match the *generative trajectory* of a full-step teacher. We theoretically and empirically show that trajectory-level supervision mitigates this factorization error, thereby enabling effective few-step decoding. We further incorporate Direct Discriminative Optimization (DDO), a reverse-KL objective that encourages mode-seeking toward the teacher's modes, yielding stronger performance on challenging reasoning tasks. Across reasoning and code-generation benchmarks, our method substantially narrows the gap between few-step and full-step decoding. The source code is available at https://github.com/Tyrion58/T3D.

1 Introduction

Inference-time efficiency is a central challenge in large language modeling, especially for real-time and compute-constrained applications [55, 28, 1]. Diffusion large language models (DLLMs) [21, 38, 29, 8, 50] offer a promising direction by enabling parallel token generation. However, existing DLLMs rely on long decoding chains consisting of many diffusion steps [32, 34, 29, 50], which significantly limits their efficiency gains. When decoding is made more aggressive by reducing the number of steps, these models struggle to accurately generate multiple tokens simultaneously [8].

Recent work [51, 6, 47, 31, 20, 54] has sought to accelerate diffusion large language models (DLLMs) and reduce their inference latency. One line of research focuses on system and decoding improvements, such as better decoding strategies [45, 7] and adapting KV caching [18, 27, 25]. Our work targets an orthogonal bottleneck: the model's internal prediction structure. In masked diffusion models, few-step decoding is fundamentally limited by the *mean-field* (token-factorized) parameterization [47, 51, 54]. As the number of decoding steps is reduced, this approximation becomes increasingly inaccurate, leading to a growing factorization error between few-step decoding and full-step decoding. As illustrated in Fig. 1, the error increases as each step is forced to predict more tokens, causing few-step predictions to deviate further from the full-step model and ultimately degrading generation quality. Previous self-distillation methods [51, 6] rely primarily on endpoint supervision from the teacher. We argue that this underuses the supervision available in the teacher's

[*]Equal Contribution. [1]Rutgers University. [2]Red Hat AI Innovation. [3]MIT-IBM Watson AI Lab. [†]Correspondence to: Tunyu Zhang <ty.zhang@rutgers.edu>, Ligong Han <ligong.han@rutgers.edu>, Dimitris N. Metaxas <dnm@cs.rutgers.edu>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

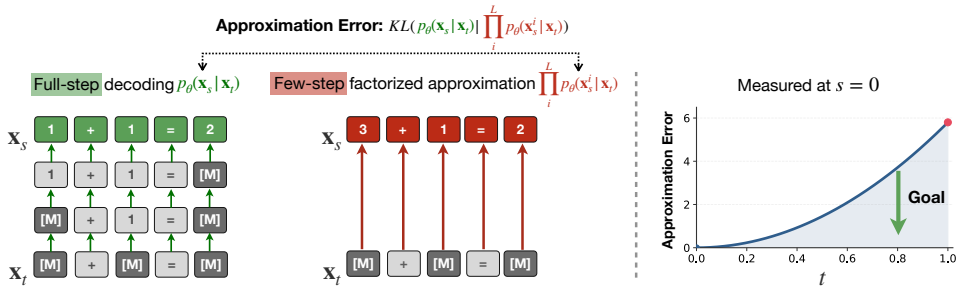

Figure 1: **Factorization error limits few-step decoding in MDLMs. (a)** Full-step decoding models the transition $p_\theta(\mathbf{x}_s \mid \mathbf{x}_t)$ through many denoising steps, while few-step decoding approximate with a token-factorized distribution $\prod_i^L p_\theta(\mathbf{x}_s^i \mid \mathbf{x}_t)$. This factorized approximation introduces error and can lead to degenerate generations. **(b)** We measured this approximation error on real MATH500 data for $s = 0$. The error increases sharply as t grows, showing that larger decoding jumps amplify factorization error. Our goal is to reduce this error and enable reliable few-step decoding.

full generative trajectory, which contains much richer information about the model's prediction structure than the endpoint alone.

Motivated by this, we propose **Trajectory Self-Distillation**, a principled self-distillation framework for effective few-step decoding in MDLMs. Our core idea is to distill a few-step student by matching the *generative trajectory* of the original full-step teacher, rather than supervising only the endpoint. This exposes the student to richer information about the teacher's prediction structure and allows it to better approximate full-step decoding under a limited step budget. Building on the analysis of ReDi [51], we further show theoretically that trajectory-level supervision reduces factorization error across intermediate reverse transitions. Crucially, our analysis also reveals why prior rectified-flow-style self-distillation does not carry over to MDLMs [21, 38, 29, 8, 50], the dominant regime of DLLMs: because the masked prior is deterministic, endpoint-based supervision is uninformative for reducing factorization error. In contrast, our method avoids this failure mode by reducing factorization error over the nontrivial decoding intervals that actually govern few-step generation.

To further improve few-step performance on complex reasoning tasks, we replace the standard forward-KL objective with *Direct Discriminative Optimization* (DDO) [56], which encourages the student to focus on the teacher's high-probability modes. Our intuition is that the mode-covering nature of forward KL can produce over-smoothed predictions and weaker trajectory alignment, while reverse-divergence objectives are inherently mode-seeking and thus yield sharper predictions. In addition, we introduce a *path-consistency* regularizer that places greater emphasis on early decoded tokens, which we find particularly helpful for reasoning.

We term our method Self-**T**rajectory **D**istillation via **DDO** (`T3D`), a simple self-distillation framework for few-step diffusion language modeling. We evaluate `T3D` on reasoning and code-generation benchmarks using both *SDAR* [8], a block-diffusion language model, and *LLaDA* [29], a full-diffusion language model. Across a broad range of decoding budgets and model families, `T3D` improves over prior few-step DLLM methods, with especially clear gains under aggressive decoding budgets. Beyond static few-step decoding, `T3D` also preserves full-step diffusion performance and remains effective under dynamic decoding. Together, these results show that trajectory self-distillation provides a practical route toward efficient few-step diffusion language modeling.

2 Related Work

Few-step Diffusion. Despite their remarkable success, diffusion models [49, 16] remain computationally expensive due to their iterative sampling process. Consistency Models [37, 36] accelerate generation by enforcing consistency across time, while flow-map-based methods [12, 4] reduce sampling cost by directly modeling state-to-state displacements. In practice, distillation-based variants often achieve stronger performance, which many attribute to their use of *teacher trajectories*. For example, Consistency Distillation [37] matches teacher intermediate states, CMT [17] bootstraps

training with teacher rollouts, and Re-MeanFlow [53] leverages teacher-rectified trajectories. Our work aims to bring this trajectory-based perspective to discrete diffusion language modeling.

Efficient Inference for Diffusion Language Models. Diffusion large language models (DLLMs) [21, 38, 29, 8, 50] have recently emerged as powerful generative models for text, but like their continuous counterparts, they require many iterative refinement steps during inference. One line of work improves efficiency through system- and decoding-level advances, such as KV caching [22, 18, 27], dynamic decoding [45], and block-structured diffusion generation [2, 8, 44, 41]. Another, orthogonal line of work aims to reduce the number of sampling steps directly. For example, EDLM [47] introduces an energy-based objective to reduce factorization error, dParallel [6] distills a few-step model by matching teacher rollouts, and ReDi [51] adopts a rectified-flow-style [24] distillation procedure. Our work addresses a missing piece in this literature: fully leveraging the supervision available throughout the denoising trajectory.

3 Background

3.1 Masked Diffusion Language Models (MDLMs)

In this work, we focus on masked diffusion language models (MDLMs) [32, 35], as they are the prominent paradigm for current large-scale diffusion language models [21, 38, 29, 8, 50].

MDLMs are diffusion-based generative models for discrete text sequences. Let p_{data} denote the data distribution. A data sample $\mathbf{x}_0 \sim p_{\text{data}}$ is a length-L token sequence $\mathbf{x}_0 = (\mathbf{x}_0^1, \ldots, \mathbf{x}_0^L)$, where $\mathbf{x}_0^i \in \mathcal{V}$ denotes a discrete token from a finite vocabulary $\mathcal{V}$ augmented with a special mask token $\boldsymbol{m}$.

The *forward (noising)* diffusion process is defined over continuous time $t \in [0, 1]$ and corrupts a sequence by independently masking tokens. The corruption distribution factorizes across tokens:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \prod_{i=1}^{L} q(\mathbf{x}_t^i \mid \mathbf{x}_0^i), \tag{1}$$

where the token-wise kernel $q(\mathbf{x}_t^i \mid \mathbf{x}_0^i)$ is governed by a monotonically decreasing noise schedule $\alpha_t \in [0, 1]$: at time t, $\mathbf{x}_t^i$ is preserved as $\mathbf{x}_0^i$ with probability α_t and replaced by the mask token $\boldsymbol{m}$ with probability $1 - \alpha_t$. We choose $\alpha_t = 1 - t$ following previous works [29, 32].

Given a noisier sequence $\mathbf{x}_t$, the *reverse (denoising)* process learns to recover a cleaner sequence $\mathbf{x}_s$ at an earlier time $s < t$. This reverse transition is approximated by a neural network p_θ that also factorizes over tokens:

$$p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) \approx \prod_{i=1}^{L} p_\theta\left(\mathbf{x}_s^{(i)} \mid \mathbf{x}_t\right). \tag{2}$$

As shown in [32, 35], maximizing the evidence lower bound (ELBO) for MDLMs admits a remarkably simple form. Concretely, the optimization objective reduces to a masked-token cross-entropy objective:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\mathbf{x}_t \sim q(\cdot \mid \mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t)\right]. \tag{3}$$

3.2 Direct Discriminative Optimization (DDO)

Direct Discriminative Optimization (DDO) [56] is a GAN-inspired objective for likelihood-based generative models. Unlike standard GANs [13], which introduce an additional discriminator network, DDO implicitly parameterizes the discriminator using likelihood ratios. Consider a pretrained model $p_{\theta_{\text{ref}}}$ that supplies "fake" samples. To distinguish real data $\mathbf{x} \sim p_{\text{data}}$ from reference samples $\mathbf{x} \sim p_{\theta_{\text{ref}}}$, the optimal discriminator is:

$$d^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\theta_{\text{ref}}}(\mathbf{x})} = \sigma\left(\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\theta_{\text{ref}}}(\mathbf{x})}\right),$$

where $\sigma(\cdot)$ denotes the sigmoid function. DDO replaces the unknown p_{data} by parameterizing a discriminator through a learnable likelihood-based model p_θ:

$$d_\theta(\mathbf{x}) := \sigma\left(\log \frac{p_\theta(\mathbf{x})}{p_{\theta_{\text{ref}}}(\mathbf{x})}\right).$$

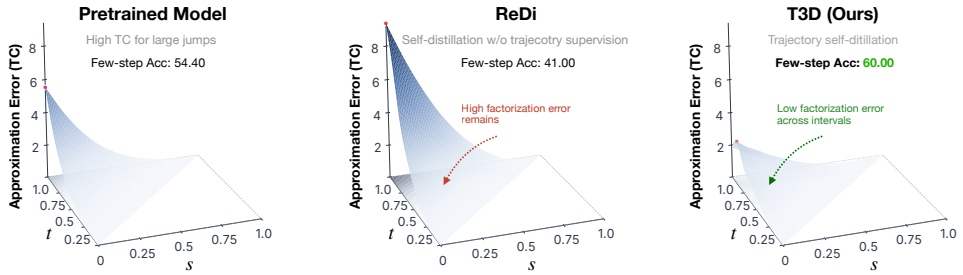

Figure 2: **Approximation Error Surface Across Decoding Interval.** We quantify factorization error using conditional total correlation (TC) on MATH500 with SDAR-4B-Chat, visualized over decoding intervals (s, t); larger intervals correspond to more aggressive few-step jumps. **(a)** The pretrained model exhibits high TC for large jumps, leading to degraded few-step accuracy. **(b)** ReDi-style self-distillation without trajectory supervision leaves high TC unresolved, thereby hurting few-step performance. **(c) T3D directly matches the teacher trajectory, substantially lowering TC across intervals and achieving strong few-step accuracy.**

Substituting this implicit discriminator into the GAN discriminator loss yields the DDO objective:

$$\min_{\theta} \mathcal{L}(\theta) = -\mathbb{E}_{x\sim p_{\text{data}}}\left[\log \sigma\left(\log \frac{p_\theta(\mathbf{x})}{p_{\theta_{\text{ref}}}(\mathbf{x})}\right)\right] - \mathbb{E}_{x\sim p_{\theta_{\text{ref}}}}\left[\log\left(1-\sigma\left(\log \frac{p_\theta(\mathbf{x})}{p_{\theta_{\text{ref}}}(\mathbf{x})}\right)\right)\right]. \quad (4)$$

With unlimited model capacity, [56] show that the global minimizer of the DDO objective above satisfies $p^*_\theta = p_{\text{data}}$.

4 Methods

4.1 Factorization Error: The Key Bottleneck in Few-Step MDLMs

As shown in Fig. 1, few-step decoding in MDLMs relies on the mean-field parameterization in Eq. 2, which factorizes the reverse transition across tokens. While necessary for tractability, this factorization introduces an approximation error that becomes more severe as the sampling budget is reduced, i.e., as the gap between s and t grows.

Following prior work [51], we quantify this error using Conditional Total Correlation (TC), defined as the expected KL divergence between the reverse transition and its token-factorized approximation:

$$TC_J(\mathbf{x}_s \mid \mathbf{x}_t) := \mathbb{E}_{\mathbf{x}_t}\left[\mathrm{KL}\Big(p(\mathbf{x}_s \mid \mathbf{x}_t) \,\Big\|\, \prod_{i=1}^{L} p(\mathbf{x}_s^i \mid \mathbf{x}_t)\Big)\right]. \quad (5)$$

Here, the Conditional TC is defined with respect to the joint distribution $J(\mathbf{x}_s, \mathbf{x}_t) = p(\mathbf{x}_t)\, p(\mathbf{x}_s \mid \mathbf{x}_t)$, induced by the marginal at time t and the reverse posterior.

As shown in Fig. 1 (b) and Fig. 2 (a), Conditional TC rises as the decoding interval becomes larger. This identifies factorization error as the central bottleneck in few-step MDLMs: when fewer steps are used, each step must model stronger cross-token dependencies, but the tokenwise factorization becomes increasingly inaccurate, leading to degraded generation quality.

4.2 Trajectory Self-Distillation

To overcome the factorization bottleneck in few-step MDLMs, we propose *trajectory self-distillation*, which trains a few-step student directly on teacher rollout trajectories.

Specifically, given a pretrained teacher model p_ϕ, we want to train a few-step student model p_θ initialized from p_ϕ by leveraging pairs of clean and intermediate states $(\mathbf{x}_0, \mathbf{x}_t)$ sampled along the teacher's generative trajectory $p_\phi^{\text{Tra}}(\mathbf{x}_{0:T})$:

$$p_\phi^{\text{Tra}}(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \quad (6)$$

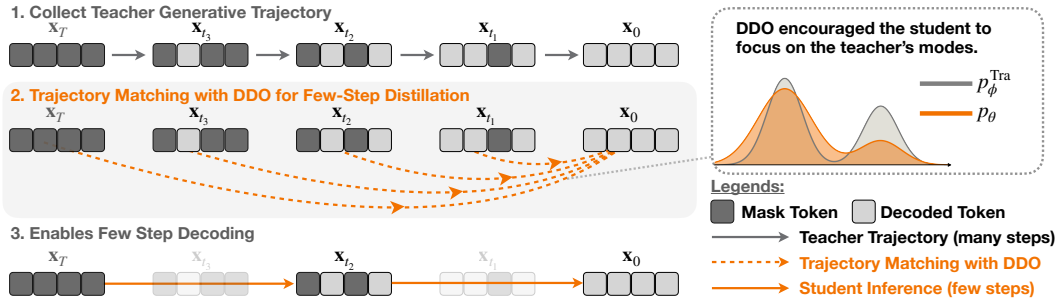

Figure 3: **Overview of T3D.** T3D first collects the teacher's full generative trajectory and distills it into a few-step student by matching the student outputs to this teacher trajectory. We use DDO as the trajectory-matching objective, which encourages the student to focus on the teacher's high-probability modes, thereby producing sharper and higher-quality predictions. After distillation, the student can skip intermediate states and perform efficient few-step decoding.

Then we define a forward-KL objective that trains the few-step student to match the teacher trajectory, leading to the following self-trajectory distillation loss:

$$\mathcal{L}_{\text{traj}}(\theta) = -\,\mathbb{E}_{p_\phi^{\text{Tra}}(\mathbf{x}_t)}\mathbb{E}_{\mathbf{x}_0\sim p_\phi^{\text{Tra}}(\mathbf{x}_0|\mathbf{x}_t)}\big[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t)\big]\,. \tag{7}$$

Intuitively, this formulation provides substantially richer supervision than endpoint-only distillation [6, 51], since it exposes the student to the teacher's prediction structure throughout the reverse process rather than only at the final target. More importantly, we show that trajectory self-distillation directly targets the source of few-step failure in MDLMs by reducing approximation error, building on the Conditional TC analysis of ReDi [51]:

Theorem 4.1 (Trajectory Distillation Induces Lower Conditional Total Correlation). *Let p_ϕ be a pretrained teacher model and p_θ a student model. Define the teacher trajectory joint distribution as $J_\phi(\mathbf{x}_s, \mathbf{x}_t)$ and the student-induced joint distribution as $J_\theta(\mathbf{x}_s, \mathbf{x}_t)$. Let θ^* be the optimal solution to Eqn. 7, and let J_{θ^*} denote the corresponding student joint distribution. Then, for any $s < t$, under mild assumptions, the following inequality holds:*

$$\mathbb{E}_t\big[TC_{J_{\theta^*}}(\mathbf{x}_s \mid \mathbf{x}_t)\big] \le \mathbb{E}_t\big[TC_{J_\phi}(\mathbf{x}_s \mid \mathbf{x}_t)\big]\,. \tag{8}$$

For proof, please see Appendix B.

We further show that trajectory-level supervision is particularly important for few-step distillation in MDLMs, where endpoint-only, rectified-flow-style [24] distillation methods such as ReDi [51] do not directly apply, because they provide no meaningful reduction in factorization error.

Corollary 4.2 (Endpoint-only Distillation Does Not Reduce Conditional Total Correlation for MDLMs). *In MDLMs [32, 35], the prior $p(\mathbf{x}_T) = \delta_{\mathbf{m}}$ is deterministic. Therefore, for any model, $p(\mathbf{x}_0 \mid \mathbf{x}_T) = q(\mathbf{x}_0)$, and consequently,*

$$TC_J(\mathbf{x}_0 \mid \mathbf{x}_T) = \mathrm{KL}\Big(q(\mathbf{x}_0) \,\Big\|\, \prod_{i=1}^{L} q(\mathbf{x}_0^i)\Big). \tag{9}$$

This quantity is a fixed constant of the data distribution and thus cannot be reduced. For proof, please see Appendix B.

As shown in Fig.2 (b), endpoint-only distillation fails to reduce Conditional TC. By contrast, Theorem 4.1 shows that trajectory self-distillation avoids this degeneracy by operating on intermediate states $\mathbf{x}_t$. Empirically, this yields a substantially lower TC surface across decoding intervals (Fig. 2 (c)), which translates into much stronger few-step decoding quality.

4.3 Improving Trajectory Alignment with DDO

The forward-KL objective in Eq. 7 provides a natural way to align the student with teacher trajectories and mitigate the trajectory-level mismatch underlying factorization error. However, as a mode-covering objective, it can still produce over-smoothed predictions and suboptimal alignment with

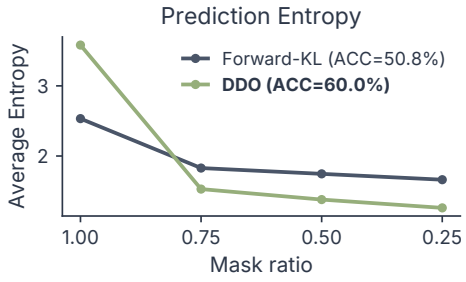

Figure 4: **Average prediction entropy during trajectory distillation.** We compare forward-KL and DDO on MATH500 with SDAR-4B-Chat. DDO maintains higher entropy at the fully masked stage but yields substantially lower entropy afterwards, suggesting broad early exploration followed by sharper refinement. This mode-seeking behavior improves trajectory alignment and reasoning accuracy.

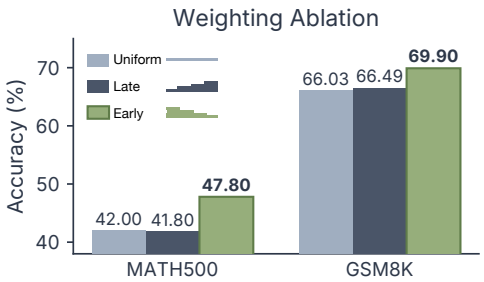

Figure 5: **Comparison of path-consistency weighting strategies for `T3D`.** We compare uniform weighting, late-token weighting, and our early-token weighting on MATH500 and GSM8K. Early-token weighting consistently achieves the best accuracy, suggesting that tokens decoded earlier in the trajectory are more critical under step compression, since their errors can propagate to later decoding steps.

teacher-generated trajectories. We argue this is harmful for complex reasoning tasks, where sharp decisions on high-probability continuations are often critical.

Motivated by this, we adopt *Direct Discriminative Optimization* (DDO) [56] to further improve the few-step quality for trajectory self-distillation. This GAN-inspired objective induces reverse-KL-like mode-seeking behavior without introducing an additional discriminator. It can be integrated into trajectory self-distillation with minimal modification, while encouraging the student to focus on the teacher's high-probability trajectories.

Formally, we define the trajectory-level DDO objective as:

$$\mathcal{L}_{\text{traj-DDO}}(\theta) = \mathbb{E}_{\mathbf{x}_t \sim p_\phi^{\text{Tra}}(\mathbf{x}_t)}\big[\, l(\theta)\,\big], \tag{10}$$

where the per-step DDO loss is:

$$l(\theta) = -\mathbb{E}_{\mathbf{x}_0 \sim p_\phi^{\text{Tra}}(\mathbf{x}_0|\mathbf{x}_t)}\left[\log \sigma\left(\beta \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_t)}{p_{\theta_{\text{ref}}}(\mathbf{x}_0|\mathbf{x}_t)}\right)\right] \tag{11}$$

$$- \mathbb{E}_{\mathbf{x}_0 \sim p_{\theta_{\text{ref}}}(\mathbf{x}_0|\mathbf{x}_t)}\left[\log\left(1 - \sigma\left(\beta \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_t)}{p_{\theta_{\text{ref}}}(\mathbf{x}_0|\mathbf{x}_t)}\right)\right)\right], \tag{12}$$

where $p_{\theta_{\text{ref}}}$ is a reference model that provides "fake" samples and is initialized from p_θ, and β is the coefficient that scales the log-likelihood ratio inside the sigmoid and controls the sharpness of the preference signal. The first term encourages the student to assign higher likelihood than the reference model to teacher-generated samples, while the second term penalizes overestimation of samples from the reference model. As illustrated in Fig. 4, DDO induces a desirable exploration–exploitation pattern along the decoding trajectory: it maintains higher entropy at the fully masked initial stage, allowing broader exploration, and produces substantially lower entropy afterwards, enabling sharper refinement around teacher-preferred modes. This sharper trajectory alignment translates into improved few-step reasoning.

Path-Consistency Regularization. We further introduce a lightweight *path-consistency* regularization that places larger weight on tokens decoded earlier in the trajectory, since errors at early steps are more likely to propagate under tight decoding budgets. Formally, given a fixed decoding budget B, let $\pi_i \in [B]$ denote the decoding step at which token $\mathbf{x}_0^i$ is generated, and define the step-dependent weight $w_i = \frac{B-\pi_i+1}{B}$. Then we define a token-level weighted path-consistency regularization loss as:

$$\mathcal{L}_{\text{path}}(\theta) = -\,\mathbb{E}_{p_\phi(\mathbf{x}_t)}\mathbb{E}_{\mathbf{x}_0 \sim p_\phi(\cdot|\mathbf{x}_t)}\left[\sum_i w_i \log p_\theta(\mathbf{x}_0^i \mid \mathbf{x}_t^{(i)})\right]. \tag{13}$$

This assigns larger training weight to earlier-decoded tokens while leaving the objective otherwise unchanged. As shown in Fig. 5, we compare against uniform weighting ($w_i = 1$) and a late-token schedule ($w_i = \pi_i/B$), which assigns larger weights to later-decoded tokens. Our early-token

Table 1: **Few-step accuracy comparison across baselines on SDAR-1.7B-Chat and SDAR-4B-Chat [8].** Few-step performance is evaluated using tokens-per-step (TokPS): for example, **Block Size**$= 4$ and **TokPS**$= 2$ means decoding uses blocks of 4 tokens while generating 2 tokens per diffusion step, resulting in $4/2 = 2$ diffusion steps per block. **SD** means Self-Distillation methods. **T3D is consistently among the strongest methods**, demonstrating the effectiveness of trajectory-level distillation for few-step generation.

TokPS	Method	SD	Block Size = 4				Block Size = 8				AVG. Gains (%)
			MATH500	GSM8K	MBPP	HumanEval	MATH500	GSM8K	MBPP	HumanEval	
SDAR-1.7B-Chat											
	Original Model	-	39.40	63.00	30.40	32.93	33.60	55.88	27.80	37.20	40.03	-
	SFT	✗	43.00	61.79	30.00	34.76	36.80	62.55	27.20	37.80	41.74	↑ 4.28
2	ReDi	✓	40.60	63.99	13.20	16.46	36.40	62.17	12.80	13.41	32.38	↓ 19.11
	dParallel	✓	43.40	68.23	22.20	24.39	45.20	67.70	23.20	**26.83**	40.14	↑ 0.29
	T3D (Ours)	✓	**47.00**	**70.96**	**27.20**	**30.49**	**47.80**	**68.84**	**26.60**	25.61	**43.06**	↑ **7.59**
	Original Model	-	5.00	13.34	10.60	12.20	4.80	12.74	10.20	10.37	9.91	-
	SFT	✗	22.40	36.62	6.20	5.49	20.00	39.65	4.40	7.93	17.84	↑ 80.05
4	ReDi	✓	15.00	32.45	3.40	5.49	12.80	29.72	4.00	4.88	13.47	↑ 35.95
	dParallel	✓	22.80	**45.26**	**10.20**	12.20	**25.40**	**42.91**	**10.40**	11.59	**22.60**	↑ **128.09**
	T3D (Ours)	✓	**25.60**	42.91	9.40	**15.24**	24.40	37.38	9.20	**14.02**	22.27	↑ 124.79
SDAR-4B-Chat											
	Original Model	-	54.40	78.77	34.20	49.39	49.60	72.33	33.40	46.95	52.38	-
	SFT	✗	54.60	54.60	26.80	37.20	54.44	77.41	25.60	29.88	46.76	↓ 10.73
2	ReDi	✓	41.00	73.62	20.00	21.95	23.60	71.87	19.20	23.17	36.80	↓ 29.74
	dParallel	✓	52.60	76.57	23.80	39.63	51.20	75.97	18.20	28.66	45.83	↓ 12.51
	T3D (Ours)	✓	**60.00**	**83.85**	**38.80**	**51.83**	**61.60**	**81.96**	**37.00**	**56.10**	**58.89**	↑ **12.43**
	Original Model	-	13.80	41.09	14.00	18.29	16.80	41.02	10.00	16.46	21.43	-
	SFT	✗	39.00	48.14	9.00	15.85	40.20	55.42	8.80	11.59	28.50	↑ 32.98
4	ReDi	✓	25.40	53.30	5.00	7.32	20.20	47.84	6.80	6.71	21.57	↑ 0.65
	dParallel	✓	34.20	45.94	13.20	20.73	40.80	53.83	9.60	20.12	29.80	↑ 39.05
	T3D (Ours)	✓	**47.80**	**69.90**	**22.60**	**23.78**	**44.80**	**63.99**	**21.20**	**23.17**	**39.66**	↑ **85.02**

weighting consistently performs best, indicating that early decoding decisions are more critical under tight step budgets because their errors are more likely to propagate through the remaining trajectory.

Final objective. Our full method, T3D (**T**rajectory self-**D**istillation via **DDO**), first collects teacher-generated trajectories and then trains a few-step student using DDO together with path-consistency regularization. The final training objective is

$$\mathcal{L}_{\text{T3D}}(\theta) = \mathcal{L}_{\text{traj-DDO}}(\theta) + \lambda\, \mathcal{L}_{\text{path}}(\theta), \tag{14}$$

where λ controls the strength of the path-consistency regularization. Figure 3 provides an overview of the framework, and the full training algorithm is given in Appendix A.

5 Experiments

5.1 Experimental Settings

Baselines. We compare T3D with representative few-step diffusion language model baselines: *ReDi* [51], *dParallel* [6], and *SFT* on real data as a supervised reference. For LLaDA experiments, we additionally include *CDLM* [20], which accelerates diffusion language models through system-level and training-based designs. All training-based baselines and T3D are trained until convergence.

Models and Benchmarks. We evaluate T3D on both block-diffusion and full-diffusion language models. For block diffusion, we use **SDAR-1.7B-Chat** and **SDAR-4B-Chat** [8]; for full diffusion, we use **LLaDA-8B-Instruct** [29]. We evaluate on four reasoning and code-generation benchmarks: **MATH500** [23], **GSM8K** [9], **MBPP** [3], and **HumanEval** [5]. These tasks require multi-step reasoning, making them sensitive to quality degradation under aggressive step compression.

Metrics. For few-step decoding and full-decoding preservation, we report **Accuracy**. For dynamic decoding, we additionally report throughput and averaged tokens per decoding steps. For LLaDA

Table 2: **Few-step accuracy comparison on LLaDA.** Following the same protocol as Table 1, we compare T3D with existing few-step decoding and self-distillation baselines under different TokPS settings. **T3D achieves the best average accuracy/extraction** at both TokPS $= 4$ and TokPS $= 8$.

TokPS	Method	MATH500	GSM8K	MBPP	HumanEval	AVG.	Gains (%)
4	Original Model	24.80	70.43	91.80	87.80	68.71	-
	ReDi	25.20	68.39	93.80	91.50	69.72	↑ 1.48
	dParallel	28.40	71.49	94.80	91.50	71.55	↑ 4.13
	CDLM	30.00	71.70	85.40	85.98	68.27	↓ 0.64
	T3D (Ours)	**30.40**	**75.89**	**98.20**	**94.50**	**74.75**	↑ **8.79**
8	Original Model	3.00	18.04	40.80	50.00	27.96	-
	ReDi	6.80	31.24	70.80	66.50	43.84	↑ 56.78
	dParallel	15.60	53.90	70.60	67.70	51.95	↑ 85.80
	CDLM	11.60	46.50	46.40	46.95	37.86	↑ 35.42
	T3D (Ours)	**25.20**	**70.13**	**86.60**	**73.80**	**63.93**	↑ **128.66**

coding tasks, we report **Extraction Rate (ER)**, following the limited executable-solution extraction ability of the base model.

Training Data and Implementation. For self-distillation methods, we collect teacher-generated trajectories from the corresponding training sets: MATH [15] for mathematical reasoning and PrimeIntellect [19] for code generation. Unless otherwise specified, trajectories are generated with static decoding and low-confidence remasking. During T3D training, the DDO reference model is periodically updated from the current student, and we mix random tokens into training inputs to improve robustness. All trainable methods are fine-tuned using full-parameter training on $8\times$ NVIDIA A100-40GB GPUs. More implementation details, including trajectory construction, decoding settings, and training cost, are provided in Appendix C.

5.2 Improving Performance of Few-Step Decoding by Self-Distillation

Settings. We evaluate few-step decoding under high **Tokens Per Step (TokPS)** settings, where larger TokPS corresponds to more aggressive parallel decoding. For SDAR, we evaluate two block sizes, 4 and 8, with TokPS $= 2$ and 4. For LLaDA, we set the maximum generation length to 1024, use block size 32, and evaluate TokPS $= 4$ and 8. These settings cover both moderate and highly compressed decoding regimes.

Results. Table 1 and Table 2 report few-step accuracy across SDAR and LLaDA models. Overall, T3D is consistently among the strongest self-distillation methods and achieves the best average performance in most settings. The gains are especially pronounced under more aggressive decoding budgets, where competing methods often degrade substantially. These results show that T3D better preserves generation quality when the diffusion process is compressed to only a few steps.

Table 3: **Preserving diffusion performance under full decoding.** We revert few-step distilled models to full diffusion decoding using static decoding (one token per step) without additional training. Results are reported under block size 4 and 4 steps per block, showing that T3D preserves diffusion performance. **Bold** numbers denote the best result among self-distillation methods.

Method	MATH500	GSM8K	MBPP	HumanEval
SDAR-1.7B-Chat				
Original Model	59.40	80.59	45.20	59.76
SFT	52.00	73.09	44.20	60.37
ReDi	47.00	73.77	27.60	31.10
dParallel	0.40	0.23	34.60	43.29
TD	49.80	72.40	35.20	32.93
T3D **(Ours)**	**56.80**	**78.01**	**41.20**	**57.32**
SDAR-4B-Chat				
Original Model	68.00	89.84	58.60	71.95
SFT	60.20	86.05	50.20	69.51
ReDi	50.40	82.03	34.00	37.80
dParallel	13.20	2.88	34.00	48.17
TD	57.40	82.11	37.60	43.90
T3D **(Ours)**	**70.00**	**89.31**	**54.20**	**73.78**

5.3 Preserving Diffusion Performance under Full Decoding

Settings. In this experiment, we investigate whether few-step distillation leads to *diffusion property forgetting*, i.e., whether a model optimized for compressed decoding degrades when reverted to the original full diffusion process. To evaluate this, we take models distilled for few-step generation and

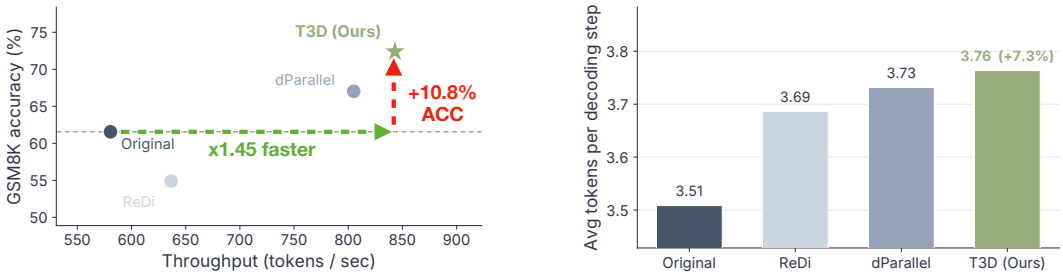

Figure 6: Dynamic decoding results on GSM8K using SDAR-4B-Chat. (a) Accuracy-throughput trade-off. T3D improves GSM8K accuracy by $+10.8\%$ over the original model while achieving $1.45\times$ higher throughput. (b) Average decoded tokens per step. T3D decodes more tokens per step under the same confidence threshold, indicating that it produces more confident predictions for adaptive decoding. Full results are provided in Appendix D.1.

directly restore them to full diffusion decoding using static decoding strategy, decoding one token per step without any additional training.

Results. Table 3 reports the results. Across both SDAR-1.7B-Chat and SDAR-4B-Chat, our methods preserve strong performance under full decoding. In particular, T3D achieves performance nearly identical to the original pretrained model on all benchmarks, and in some cases slightly outperforms it. In contrast, prior baselines such as ReDi and dParallel exhibit substantial degradation.

Discussions. These results indicate that trajectory self-distillation does not overfit to few-step decoding, but instead preserves the model's fine-grained denoising capability. Overall, **our approach enables few-step generation without sacrificing full diffusion performance**.

5.4 Experiments on Dynamic Decoding

Settings. Dynamic decoding [45, 48] adaptively determines how many tokens to decode at each step based on model confidence. Although T3D is trained under fixed static step budgets and our main experiments use static decoding for controlled comparison, we further evaluate whether the learned few-step model remains effective when combined with adaptive decoding strategy. All dynamic decoding experiments use block size 4, 4 steps per block, and a fixed confidence threshold of 0.9.

Results. Fig. 6 visualizes dynamic decoding on GSM8K, with full results reported in Appendix D.1. Under the same dynamic decoding rule, T3D improves the original model by $+10.8\%$ absolute accuracy while achieving $1.45\times$ higher throughput (Fig. 6 a). It also decodes more tokens per step on average (Fig. 6 b), suggesting that T3D produces more confident predictions and enables larger adaptive decoding steps. These results show that T3D remains effective beyond the static decoding regime used during training.

5.5 Ablation Study

We conduct a component-wise ablation to examine the contribution of each design in T3D. As shown in Table 4, trajectory distillation provides the main improvement over the original model, confirming the importance of matching teacher rollout trajectories under aggressive few-step decoding. Adding DDO further improves performance, suggesting that mode-seeking trajectory matching produces sharper predictions under tight decoding budgets. Finally, the path-consistency loss provides an additional gain by emphasizing early decoded tokens, which helps reduce error propagation. Full ablation results across more settings are provided in Sec. E.

Table 4: Component-wise ablation under aggressive few-step decoding. Results are averaged over four benchmarks using SDAR-4B-Chat with TokPS $= 4$ and block size $= 8$. Setting (c) corresponds to the full T3D objective.

Method		Acc.	Gains (%)
Original		16.80	–
(a)	+ TD ($\mathcal{L}_{\text{traj}}$ in Eqn. 7)	38.80	↑ 130.95
(b)	+ DDO ($\mathcal{L}_{\text{traj-DDO}}$ in Eqn. 10)	43.20	↑ 157.14
(c)	+ Path Loss ($\mathcal{L}_{\text{path}}$ in Eqn. 13)	**45.00**	↑ **167.86**

6 Conclusion

We presented T3D, a simple and effective framework for few-step diffusion language modeling based on *trajectory self-distillation*. Our key insight is that few-step decoding in MDLMs is bottlenecked by factorization error, and that the teacher's full generative trajectory provides much richer supervision than the endpoint alone for reducing this error. We further uncover a fundamental failure mode of prior endpoint-based, rectified-flow-style self-distillation in MDLMs, and show that T3D avoids this issue by distilling over intermediate decoding intervals, where the reverse process remains informative. Across reasoning and code-generation benchmarks, T3D consistently outperforms prior few-step DLLM methods, substantially narrowing the gap to full-step diffusion decoding.

Limitations. Our method has two inherent limitations. First, because it relies on self-distillation, student performance is ultimately bounded by teacher quality. Second, trajectory collection requires full-step teacher rollouts, incurring an offline cost that scales with dataset size and decoding budget.

Acknowledgement

We thank all reviewers, AC, and SAC for their valuable comments. Research partially funded by research grants to Metaxas from NSF: 2310966, 2235405, 2212301, 2003874, 1951890 and NIH 2R01HL127661. This research is also supported in part by Lambda, Inc.

Competing Interests. The authors declare no competing interests.

References

[1] K. Alizadeh, S. I. Mirzadeh, D. Belenko, S. Khatamifard, M. Cho, C. C. Del Mundo, M. Rastegari, and M. Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12562–12584, 2024.

[2] M. Arriola, A. Gokaslan, J. T. Chiu, Z. Yang, Z. Qi, J. Han, S. S. Sahoo, and V. Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

[3] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[4] N. M. Boffi, M. S. Albergo, and E. Vanden-Eijnden. Flow map matching. *arXiv preprint arXiv:2406.07507*, 2, 2024.

[5] M. Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] Z. Chen, G. Fang, X. Ma, R. Yu, and X. Wang. dparallel: Learnable parallel decoding for dllms. *arXiv preprint arXiv:2509.26488*, 2025.

[7] Z. Chen, G. Fang, X. Ma, R. Yu, and X. Wang. Dmax: Aggressive parallel decoding for dllms. *arXiv preprint arXiv:2604.08302*, 2026.

[8] S. Cheng, Y. Bian, D. Liu, L. Zhang, Q. Yao, Z. Tian, W. Wang, Q. Guo, K. Chen, B. Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

[9] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] J. Deschenaux and C. Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, pages 5007–5045, 2025.

[11] J. Gai, G. Zeng, H. Zhang, and A. Raghunathan. Differential smoothing mitigates sharpening and improves llm reasoning. *arXiv preprint arXiv:2511.19942*, 2025.

[12] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[14] S. Hayakawa, Y. Takida, M. Imaizumi, H. Wakaki, and Y. Mitsufuji. Distillation of discrete diffusion through dimensional correlations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 22259–22297. PMLR, 2025.

[15] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[17] Z. Hu, C.-H. Lai, Y. Mitsufuji, and S. Ermon. Cmt: Mid-training for efficient learning of consistency, mean flow, and flow map models. *arXiv preprint arXiv:2509.24526*, 2025.

[18] Z. Hu, J. Meng, Y. Akhauri, M. S. Abdelfattah, J.-s. Seo, Z. Zhang, and U. Gupta. Accelerating diffusion language model inference via efficient kv caching and guided diffusion. *arXiv preprint arXiv:2505.21467*, 2025.

[19] S. Jaghouar, J. M. Ong, M. Basra, F. Obeid, J. Straube, M. Keiblinger, E. Bakouch, L. Atkins, M. Panahi, C. Goddard, et al. Intellect-1 technical report. *arXiv preprint arXiv:2412.01152*, 2024.

[20] M. Kim, C. Xu, C. Hooper, H. Singh, B. Athiwaratkun, C. Zhang, K. Keutzer, and A. Gholami. Cdlm: Consistency diffusion language models for faster sampling. *arXiv preprint arXiv:2511.19269*, 2025.

[21] I. Labs, S. Khanna, S. Kharbanda, S. Li, H. Varma, E. Wang, S. Birnbaum, Z. Luo, Y. Miraoui, A. Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

[22] T. Li, M. Chen, B. Guo, and Z. Shen. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*, 2025.

[23] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[24] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[25] Z. Liu, Y. Yang, Y. Zhang, J. Chen, C. Zou, Q. Wei, S. Wang, and L. Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025.

[26] K. Lu and T. M. Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. https://thinkingmachines.ai/blog/on-policy-distillation.

[27] X. Ma, R. Yu, G. Fang, and X. Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

[28] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, H. Jin, T. Chen, and Z. Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Computing Surveys*, 58(1):1–37, 2025.

[29] S. Nie, F. Zhu, Z. You, X. Zhang, J. Ou, J. Hu, J. Zhou, Y. Lin, J.-R. Wen, and C. Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

[30] A. Petrenko, B. Lipkin, K. Chen, E. Wijmans, M. F. Cusumano-Towner, R. Giryes, and P. Kraehenbuehl. Entropy-preserving reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[31] Y.-Y. Qian, J. Su, L. Hu, P. Zhang, Z. Deng, P. Zhao, and H. Zhang. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *arXiv preprint arXiv:2601.07568*, 2026.

[32] S. Sahoo, M. Arriola, Y. Schiff, A. Gokaslan, E. Marroquin, J. Chiu, A. Rush, and V. Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

[33] K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[34] Y. Schiff, S. S. Sahoo, H. Phung, G. Wang, S. Boshar, H. Dalla-torre, B. P. de Almeida, A. M. Rush, T. PIERROT, and V. Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[35] J. Shi, K. Han, Z. Wang, A. Doucet, and M. Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[36] Y. Song and P. Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[37] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023.

[38] Y. Song, Z. Zhang, C. Luo, P. Gao, F. Xia, H. Luo, Z. Li, Y. Yang, H. Yu, X. Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

[39] Teknium. OpenHermes 2.5: An open dataset of synthetic data for generalist LLM assistants. `https://huggingface.co/datasets/teknium/OpenHermes-2.5`, 2023.

[40] C. Wang, Z. Li, J. Bai, Y. Zhang, S. Cui, Z. Zhao, and Y. Wang. Arbitrary entropy policy optimization breaks the exploration bottleneck of reinforcement learning. *arXiv preprint arXiv:2510.08141*, 2025.

[41] X. Wang, C. Xu, Y. Jin, J. Jin, H. Zhang, and Z. Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*, 2025.

[42] Y. Wang, L. Yang, B. Li, Y. Tian, K. Shen, and M. Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025.

[43] G. Wolfer and S. Watanabe. Geometric aspects of data-processing of markov chains. *Transactions of Mathematics and Its Applications*, 8(1):tnae001, 2024.

[44] C. Wu, H. Zhang, S. Xue, S. Diao, Y. Fu, Z. Liu, P. Molchanov, P. Luo, S. Han, and E. Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025.

[45] C. Wu, H. Zhang, S. Xue, Z. Liu, S. Diao, L. Zhu, P. Luo, S. Han, and E. Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

[46] T. Xie, S. Xue, Z. Feng, T. Hu, J. Sun, Z. Li, and C. Zhang. Variational autoencoding discrete diffusion with enhanced dimensional correlations modeling. In *International Conference on Learning Representations*, pages 39109–39132, 2026.

[47] M. Xu, T. Geffner, K. Kreis, W. Nie, Y. Xu, J. Leskovec, S. Ermon, and A. Vahdat. Energy-based diffusion language models for text generation. *arXiv preprint arXiv:2410.21357*, 2024.

[48] H. Yang, R. Hu, Z. Sun, R. Zhou, Y. Cai, and Y. Wang. Wavefrontdiffusion: Dynamic decoding schedule for improved reasoning. *arXiv preprint arXiv:2511.19473*, 2025.

[49] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023.

[50] J. Ye, Z. Xie, L. Zheng, J. Gao, Z. Wu, X. Jiang, Z. Li, and L. Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

[51] J. Yoo, W. Kim, and S. Hong. Redi: Rectified discrete flow. *arXiv preprint arXiv:2507.15897*, 2025.

[52] J. Yoo, W. Kim, C. Lee, and S. Hong. Infinite mask diffusion for few-step distillation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 149532–149558. PMLR, 2026.

[53] X. Zhang, S. Tan, Q. Nguyen, Q. Dao, L. Han, X. He, T. Zhang, A. Mrdovic, and D. Metaxas. Flow straighter and faster: Efficient one-step generative modeling via meanflow on rectified trajectories. *arXiv preprint arXiv:2511.23342*, 2025.

[54] Y. Zhang, A. Schwing, and Z. Zhao. Variational masked diffusion models. *arXiv preprint arXiv:2510.23606*, 2025.

[55] R. Zhen, J. Li, Y. Ji, Z. Yang, T. Liu, Q. Xia, X. Duan, Z. Wang, B. Huai, and M. Zhang. Taming the titans: A survey of efficient llm inference serving. *arXiv preprint arXiv:2504.19720*, 2025.

[56] K. Zheng, Y. Chen, H. Chen, G. He, M.-Y. Liu, J. Zhu, and Q. Zhang. Direct discriminative optimization: Your likelihood-based visual generative model is secretly a gan discriminator. *arXiv preprint arXiv:2503.01103*, 2025.

[57] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, 2023.

[58] Y. Zhu, X. Wang, S. Lathuilière, and V. Kalogeiton. Di [m] o: Distilling masked diffusion models into one-step generator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18606–18618, 2025.

Appendix

A Algorithm . . . 15

B Proof of Theoretical Analysis . . . 15

C Implementation Details. . . . 16

C.1 Mixture of Random Tokens . . . 16

C.2 Multi-Round and Self-Play Update . . . 17

C.3 Prompts . . . 17

C.4 Accelerated Inference . . . 17

C.5 Other Implementation Details . . . 17

D Additional Experiments . . . 18

D.1 Additional Results on Dynamic Decoding . . . 18

D.2 Robustness to Decoding Temperature . . . 19

D.3 Answer Extraction and Task Accuracy . . . 19

D.4 Useful Exploration under Reverse-KL Training . . . 20

D.5 Experiments with Multiple Seeds . . . 21

D.6 Cross-Domain Generalization to WinoGrande . . . 21

D.7 Open-Ended Instruction Following . . . 22

D.8 Sensitivity to Teacher Quality . . . 22

E Ablation Study . . . 22

E.1 The Effectiveness of λ in Training Objective . . . 22

E.2 Preserving Full-Step Diffusion Properties . . . 23

E.3 Ablation Study on Few-Step Generation . . . 23

E.4 Which Part of the Trajectory Matters? . . . 24

F Additional Related Work . . . 25

A Algorithm

In this section, we describe the training algorithm of T3D. Algorithm 1 provides the pseudocode of the full training procedure, while Fig. 3 presents a high-level overview of the method for better conceptual understanding.

Algorithm 1 T3D Training

Require: Teacher model p_ϕ
Require: Student model p_θ (initialized from teacher)
Require: Path regularization weight λ, DDO coefficient β, reference update interval K
Sample trajectory pairs $(\mathbf{x}_0, \mathbf{x}_t) \sim p_\phi$
Initialize reference model $p_{\theta_{ref}} \leftarrow \text{StopGrad}(p_\theta)$ and step counter $n \leftarrow 0$
repeat
  Compute trajectory DDO loss $\mathcal{L}_{\text{traj}-\text{DDO}}$with $p_{\theta_{ref}}$ and β
  Compute path consistency loss $\mathcal{L}_{\text{path}}$
  Update student model using

$$\mathcal{L} = \mathcal{L}_{\text{traj}-\text{DDO}} + \lambda \mathcal{L}_{\text{path}}$$

  $n \leftarrow n + 1$
  if $n \bmod K = 0$ **then**
    Update reference model $p_{\theta_{ref}} \leftarrow \text{StopGrad}(p_\theta)$
  end if
until convergence
output p_θ

B Proof of Theoretical Analysis

In this section, we provide detailed proofs for the theoretical results presented in the main paper. Our analysis focuses on understanding the behavior of trajectory self-distillation under few-step decoding and its effect on the factorization properties of the reverse diffusion process.

Assumption B.1. Following [51], we assume that the trained student model p_{θ^*} attains the optimum of the MDLM objective:

$$\forall\, t \in [0,1], \quad p_{\theta^*} = \arg\min_{p_\theta} \ \mathrm{KL}\Big(p_\phi(\mathbf{x}_s \mid \mathbf{x}_t) \,\Big\|\, p_\theta(\mathbf{x}_s \mid \mathbf{x}_t)\Big). \tag{15}$$

Assumption B.2. Let P be the family of T-step decoding processes. We assume that $\forall t \in [T]$, $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ lies within the log-convex hull of P.

Lemma B.3 (**Pythagorean Inequality for KL Divergence** [43])**.** *Let $\mathcal{Q}$ be a log-convex set. If $q^* = \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(p\|q)$ and $r \in \mathcal{Q}$, then*

$$\mathrm{KL}(p\|r) \geq \mathrm{KL}(p\|q^*) + \mathrm{KL}(q^*\|r).$$

Theorem B.4 (**Trajectory Distillation Induces Lower Conditional Total Correlation**)**.** *Let p_ϕ be a pretrained teacher model and p_θ a student model. Define the teacher trajectory joint distribution as $J_\phi(\mathbf{x}_s, \mathbf{x}_t)$ and the student-induced joint distribution as $J_\theta(\mathbf{x}_s, \mathbf{x}_t)$. Let θ^* be the optimal solution to Eqn. 7, and let J_{θ^*} denote the corresponding student joint distribution. Then, for any $s < t$, under mild assumption, the following inequality holds:*

$$\mathbb{E}_t[TC_{J_{\theta^*}}(\mathbf{x}_s \mid \mathbf{x}_t)] \leq \mathbb{E}_t\big[TC_{J_\phi}(\mathbf{x}_s \mid \mathbf{x}_t)\big]\,. \tag{16}$$

Proof.

$$TC_{J_\phi}(\mathbf{x}_s \mid \mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_t \sim p_\phi(\mathbf{x}_t)} \left[\mathrm{KL}(p_\phi(\mathbf{x}_s \mid \mathbf{x}_t) \Big\| \prod_{i=1}^{L} p_\phi(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big) \right] \tag{17}$$

$$\geq \mathbb{E}_{\mathbf{x}_t \sim p_\phi(\mathbf{x}_t)} \Bigg[\mathrm{KL}\Big(p_\phi(\mathbf{x}_s \mid \mathbf{x}_t) \Big\| p_{\theta^*}(\mathbf{x}_s \mid \mathbf{x}_t) \Big) + \mathrm{KL}\Big(p_{\theta^*}(\mathbf{x}_s \mid \mathbf{x}_t) \Big\| \prod_{i=1}^{L} p_\phi(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big) \Bigg] \tag{18}$$

$$\geq \mathbb{E}_{\mathbf{x}_t \sim p_\phi(\mathbf{x}_t)} \left[\mathrm{KL}\Big(p_{\theta^*}(\mathbf{x}_s \mid \mathbf{x}_t) \Big\| \prod_{i=1}^{L} p_\phi(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big) \right] \tag{19}$$

$$= \mathbb{E}_{\mathbf{x}_t \sim p_\phi(\mathbf{x}_t)} \Bigg[\mathrm{KL}\Big(p_{\theta^*}(\mathbf{x}_s \mid \mathbf{x}_t) \Big\| \prod_{i=1}^{L} p_{\theta^*}(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big) + \sum_{i}^{N} \mathrm{KL}\Big(p_{\theta^*}(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big\| p_\phi(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big) \Bigg] \tag{20}$$

$$\geq \mathbb{E}_{\mathbf{x}_t \sim p_\phi(\mathbf{x}_t)} \left[\mathrm{KL}\Big(p_{\theta^*}(\mathbf{x}_s \mid \mathbf{x}_t) \Big\| \prod_{i=1}^{L} p_{\theta^*}(\mathbf{x}_s^i \mid \mathbf{x}_t) \Big) \right] \tag{21}$$

$$= TC_{J_{\theta^*}}(\mathbf{x}_s \mid \mathbf{x}_t). \tag{22}$$

The first inequality follows from Assumption B.1, which equates optimizing trajectory self-distillation with minimizing the expected KL divergence. The result then follows by applying Lemma B.3 and Assumption B.2. All assumptions invoked here are inherited from [51]. □

Corollary B.5 (Endpoint-only Distillation Does Not Reduce Conditional Total Correlation). *In MDLMs, the prior $p(\mathbf{x}_T) = \delta_{\mathbf{m}}$ is deterministic. Therefore, for any model, $p(\mathbf{x}_0 \mid \mathbf{x}_T) = q(\mathbf{x}_0)$, and consequently,*

$$TC_J(\mathbf{x}_0 \mid \mathbf{x}_T) = \mathrm{KL}\Big(q(\mathbf{x}_0) \Big\| \prod_{i=1}^{L} q(\mathbf{x}_0^i) \Big). \tag{23}$$

This quantity is a fixed constant of the data distribution and thus cannot be reduced by endpoint-only distillation.

Proof. Since $p(\mathbf{x}_T) = \delta_{\mathbf{m}}$, the joint is uniquely $J(\mathbf{x}_0, \mathbf{x}_T) = q(\mathbf{x}_0)\, \delta_{\mathbf{m}}(\mathbf{x}_T)$, so $p(\mathbf{x}_0 \mid \mathbf{x}_T) = q(\mathbf{x}_0)$ regardless of the model, and the Conditional TC reduces to the unconditional Total Correlation of the data distribution. □

C Implementation Details.

In this section, we provide implementation details of our method and experimental setup.

C.1 Mixture of Random Tokens

As described in Sec. 5.1, we replace some mask tokens with random tokens sampled from the vocabulary $\mathcal{V}$ uniformly. This design is inspired by recent work on one-step discrete generative modeling for images [58], where mixing mask tokens with uniformly sampled tokens is shown to improve training stability and robustness. Formally, let $\mathbf{x} = (x_1, \ldots, x_L)$ denote a token sequence of length L, and let $\mathcal{V}$ denote the vocabulary. For each position i, we introduce a binary replacement indicator $r_i \sim \mathrm{Bernoulli}(p_{\mathrm{rand}})$, where p_{rand} is the probability of replacing a mask token with a random token.

C.2 Multi-Round and Self-Play Update

In our loss function Eqn. 11, we introduce a reference model $p_{\theta_{\text{ref}}}$, which is initialized from the student model p_θ. Following the setup of prior work [56], we adopt a multi-round refinement strategy for training. Formally, this process can be written as:

$$\text{Round } n: \quad \cdots \rightarrow \underbrace{p_{\theta^*_{n-1}}}_{\text{Reference}} \rightarrow \underbrace{\sigma\left(\beta \log \frac{p_{\theta_n}}{p_{\theta^*_{n-1}}}\right)}_{\text{Discriminator}}$$

$$\text{Round } n+1: \quad \rightarrow \underbrace{p_{\theta^*_n}}_{\text{Reference}} \rightarrow \cdots$$

where θ^*_{n-1} denotes the best-performing student model obtained in round n. In each round, the reference model serves as a fixed generator. In our experiments, we update the reference model every 10 global steps ($K = 10$ in Algorithm 1), which corresponds to one round in our training schedule. We set $\beta = 0.1$ in all experiments. With $\beta = 1$, Eqn. 11 is exactly the DDO objective in Eqn. 4 applied at each intermediate state $\mathbf{x}_t$, with teacher trajectory samples $\mathbf{x}_0 \sim p_\phi^{\text{Tra}}(\mathbf{x}_0 \mid \mathbf{x}_t)$ as real data.

Intuition. Since the reference model is a recent copy of the student, its samples reflect what the student currently generates. DDO thus acts as a discriminator between teacher and student samples: it raises the likelihood of outputs that the teacher generates but the student under-weights, and lowers the likelihood of outputs that the student generates but the teacher does not, concentrating the student on the teacher's high-probability modes. This plays a similar role to low-temperature or guided sampling at inference time, but is trained directly into the student, which is particularly useful for few-step decoding, where fewer steps remain to revise uncertain predictions.

C.3 Prompts

In this section, we present the prompts used in our experiments. These prompts are used to query the model and generate responses, which are then collected as trajectories for training.

Prompt For Math Reasoning

[User]: `{problem}`. Please reason step by step, and put your final answer within `boxed{}`.
You are a precise math problem solver. Solve the given math problem step by step.
[Assistant]:

Prompt For Code Generation

[User]: This is the problem: `{problem}`. Place your code within a single Python code block ```` ```python``` ````. Do not include more than one code block.
[Assistant]:

C.4 Accelerated Inference

For all SDAR-series experiments, rollouts are performed using `JetEngine` [2], a vLLM-style inference framework tailored for diffusion language models. `JetEngine` is a lightweight yet high-performance inference engine designed for SDAR models and other block-wise diffusion decoding architectures. It supports both dense and MoE models, as well as Tensor Parallel distributed inference, and achieves significant speedups compared to naive inference implementations.

C.5 Other Implementation Details

Baselines. We compare against *ReDi* [51], which learns from teacher-generated clean samples $\boldsymbol{x}_0$ paired with randomly corrupted noisy samples $\boldsymbol{x}_t$. We also include *dParallel* [6], which maximizes the transition probability from fully masked sequences to teacher-generated clean sequences. *SFT* is trained on real data and serves as a supervised reference rather than a self-distillation baseline. For LLaDA, we additionally evaluate the official *CDLM* [20] checkpoint under our setting.

[2] https://github.com/Labman42/JetEngine

Training Data. For self-distillation methods, we collect model-generated responses on the MATH training set [15] for mathematical reasoning and the PrimeIntellect dataset [19] for code generation. For SFT, we use data derived from Bespoke-Stratos-17k [21]. Following prior work [41], we use open-source collections pre-filtered to a maximum sequence length of 600 tokens.

Trajectory Construction. We prompt the teacher model to answer questions from the corresponding training sets and collect its generated trajectories. To improve data quality, we use low-confidence remasking [29, 42] with static decoding, using block size 4 and 4 steps per block. Teacher rollouts are sampled with temperature 0.1. We collect one trajectory per prompt and apply no correctness filtering, so all rollouts are used regardless of whether the final answer is correct; we study the effect of teacher quality in Appendix D.8. To recover the generation trajectory, we record the decoding order of tokens in the final clean sequence, following prior work [42]. Given a clean sequence and its decoding order, intermediate states $\mathbf{x}_t$ are constructed by masking tokens according to the recorded order. We also mix random tokens into the input for training robustness, following previous work [58].

Training Cost. All trainable methods are trained with full-parameter fine-tuning on $8\times$ NVIDIA A100-40GB GPUs. For SDAR-4B-Chat, trajectory collection takes approximately 1.5 hours with `JetEngine` acceleration[3], and T3D training takes approximately 8 hours. Under the same hardware setting, dParallel and ReDi require roughly 4–5 hours. Thus, T3D introduces a modest additional offline training cost from DDO, while the resulting inference speedup applies at deployment time.

D Additional Experiments

This section provides additional experiments that complement the main results. We report full dynamic decoding results, analyze full-step diffusion preservation, evaluate robustness across multiple seeds, and examine the generalization of T3D to WinoGrande. We further report results under greedy decoding and different temperatures, analyze answer extraction, evaluate open-ended instruction following on MT-Bench, and study the sensitivity of T3D to teacher quality.

D.1 Additional Results on Dynamic Decoding

We provide the full dynamic decoding results in Table 5. For reference, we additionally report the Forward-KL variant of T3D, denoted as FKL, which corresponds to the objective in Eqn. 7. All experiments use SDAR-4B-Chat with a block size of 4, 4 steps per block, a confidence threshold of 0.9, and a temperature of 0.1. In addition to accuracy, we report throughput, latency, average decoding steps, and output length to characterize the efficiency–quality trade-off.

Overall, T3D consistently achieves strong performance under dynamic decoding. On MATH500 and GSM8K, T3D improves both accuracy and throughput over the original model, showing that trajectory self-distillation remains effective even when the number of decoded tokens is chosen adaptively at inference time. On code-generation benchmarks, T3D also substantially improves efficiency and maintains competitive accuracy. These results support the conclusion in the main text that T3D is compatible with adaptive decoding strategies, although it is trained under static decoding budgets.

The gains of T3D are larger on math than on code. We attribute this to two factors. First, math solutions are longer and span more decoding blocks, so factorization errors accumulate over more steps and trajectory distillation has more room to help, whereas code outputs are shorter and more templated. Second, the teacher is stronger on math than on code, providing higher-quality trajectories for distillation.

Greedy decoding. The results above use a temperature of 0.1, since the official SDAR inference engine (`JetEngine`) did not support greedy decoding at the time of our experiments: setting the temperature to 0 causes a division by zero in its sampling stage. After fixing this issue, we rerun dynamic decoding on MATH500 with greedy decoding under the same protocol (block size 4, 4 steps per block, and confidence threshold 0.9). As shown in Table 6, T3D achieves the highest accuracy,

[3] `https://github.com/Labman42/JetEngine`

Table 5: Dynamic decoding results with block size 4, 4 steps per block, confidence threshold 0.9, and temperature 0.1. We report throughput (TPS), per-sample latency (Latency), average decoding steps and sequence length (Avg Steps and Avg Length), and accuracy (Acc). **Bold** numbers indicate the best performance among baseline methods. All experiments are done using SDAR-4B-Chat model.

Dataset	Method	TPS↑	Latency↓	Avg Steps↓	Avg Length	Acc↑
MATH500	Original	657.72	1.10	196.19	721.90	39.00
	ReDi	715.71	1.04	198.24	757.05	27.00
	dParallel	692.08	0.95	170.22	653.98	45.80
	FKL	693.85	0.97	177.99	678.55	44.00
	T3D (Ours)	**791.23**	**0.66**	**137.95**	525.50	**49.40**
GSM8K	Original	580.60	0.43	71.12	249.52	61.56
	ReDi	636.58	0.49	84.63	311.99	54.89
	dParallel	805.02	0.39	83.23	310.58	67.02
	FKL	696.99	0.47	89.78	330.82	62.40
	T3D (Ours)	**843.05**	**0.37**	**83.03**	312.48	**72.40**
MBPP	Original	262.66	0.36	27.25	93.64	23.40
	ReDi	298.83	0.21	17.11	62.57	10.00
	dParallel	215.65	0.63	36.03	135.16	8.40
	FKL	**314.99**	0.31	26.43	98.80	9.80
	T3D (Ours)	313.18	**0.19**	**16.94**	61.62	**23.60**
HumanEval	Original	175.48	0.73	36.56	127.54	33.54
	ReDi	163.77	0.47	21.23	76.75	10.00
	dParallel	130.34	0.48	17.41	62.19	23.78
	FKL	216.39	0.29	17.15	62.10	23.17
	T3D (Ours)	**222.68**	**0.26**	**16.21**	58.10	**29.27**

Table 6: Dynamic decoding results on MATH500 under greedy decoding, with block size 4, 4 steps per block, and confidence threshold 0.9. All experiments use SDAR-4B-Chat.

Method	Acc↑	Avg Length	Avg Steps↓	Tokens/Step↑	Wall Time (s)↓	Latency (ms)↓
ReDi	39.00	787.3	423.8	**3.33**	165.9	331.8
dParallel	56.60	605.6	204.2	3.16	126.5	253.1
T3D (Ours)	**57.00**	548.9	**172.7**	3.19	**122.7**	**245.4**

the fewest decoding steps, and the lowest latency. Compared with dParallel, T3D improves accuracy while requiring 15.4% fewer decoding steps.

D.2 Robustness to Decoding Temperature

We further evaluate few-step decoding under different sampling temperatures, including greedy decoding, using SDAR-4B-Chat on MATH500 with block size 4. As shown in Table 7, T3D achieves the best accuracy at every temperature under both TokPS $= 2$ and TokPS $= 4$. The gap is largest at high temperature under TokPS $= 4$, where the original model drops to 0.00 accuracy while T3D still reaches 37.20.

D.3 Answer Extraction and Task Accuracy

Since DLLMs decode the tokens of a block in parallel, producing well-formed outputs (e.g., a `\boxed{}` answer or an executable code block) becomes harder under aggressive few-step decoding, and extraction failures directly lower accuracy. To separate format compliance from task-solving ability, we report the answer extraction rate (ER) together with accuracy.

LLaDA coding. Table 8 reports execution accuracy and ER on MBPP and HumanEval with LLaDA-8B-Instruct. T3D achieves the highest ER in all settings, and its advantage grows under TokPS $= 8$,

Table 7: Few-step accuracy on MATH500 under different decoding temperatures, using SDAR-4B-Chat with block size 4.

TokPS	Method	Greedy	$T = 0.1$	$T = 0.5$	$T = 1.0$
2	Original	55.60	56.20	47.40	27.20
	ReDi	39.00	40.80	36.80	36.00
	dParallel	52.00	52.20	47.80	48.80
	T3D (Ours)	**60.00**	**62.20**	**57.60**	**55.40**
4	Original	16.60	14.20	4.40	0.00
	ReDi	24.80	24.60	23.20	17.20
	dParallel	34.60	36.20	34.80	23.60
	T3D (Ours)	**48.20**	**45.40**	**46.60**	**37.20**

Table 8: Execution accuracy (Acc) and extraction rate (ER) on coding benchmarks with LLaDA-8B-Instruct.

TokPS	Method	MBPP		HumanEval	
		Acc	ER	Acc	ER
4	Original	15.2	91.8	17.1	87.8
	ReDi	6.0	93.8	15.9	91.5
	dParallel	**18.8**	94.8	18.3	91.5
	CDLM	16.8	85.4	**21.3**	86.0
	T3D (Ours)	16.8	**98.2**	18.3	**94.5**
8	Original	6.6	40.8	6.1	50.0
	ReDi	6.0	70.8	4.3	66.5
	dParallel	6.2	70.6	6.1	67.7
	CDLM	5.2	46.4	**10.4**	47.0
	T3D (Ours)	**6.6**	**86.6**	9.2	**73.8**

where it improves the MBPP ER of the original model from 40.8 to 86.6. In execution accuracy, T3D is competitive with the strongest baselines.

MATH500. We further compute the accuracy among successfully extracted samples on MATH500, using SDAR-4B-Chat with block size 4, TokPS $= 2$, and greedy decoding. As shown in Table 9, all methods reach comparable extraction rates (90–97%) in this setting, and T3D still achieves the highest accuracy when restricted to extracted samples (61.73 vs. 57.65 for dParallel and 43.14 for ReDi). The gains of T3D therefore come from improved problem solving under parallel decoding, not only from better format compliance.

D.4 Useful Exploration under Reverse-KL Training

A potential concern is that the reverse-KL training objective may reduce model diversity, which could harm generation quality on open-ended tasks. However, entropy reduction is not unique to T3D; it is common across many post-training methods, including fine-tuning, RL, and distillation [11, 40, 30]. More importantly, recent work [26] suggests that lower-entropy, mode-seeking objectives can be beneficial for reasoning, since they concentrate probability mass on coherent solution paths rather than diffuse alternatives.

We provide three additional analyses to examine whether T3D suffers from diversity collapse.

- **Entropy and output diversity.** As shown in Fig. 6 (b), T3D does not exhibit uniform diversity collapse relative to Forward-KL. Instead, it shows a stage-wise exploration–exploitation pattern: higher entropy at early decoding stages (mask ratio $= 1.0$), indicating broader exploration, and lower entropy at later stages, enabling sharper refinement. This behavior is desirable for reasoning, where the model should explore possible solution paths early and refine toward a coherent answer later.

Table 9: Extraction rate (ER) and accuracy on MATH500, using SDAR-4B-Chat with block size 4, TokPS $= 2$, and greedy decoding. Acc (extracted) is computed only over successfully extracted samples.

Method	ER (%)	Acc (all)	Acc (extracted)
ReDi	90.4 (452/500)	39.00	43.14
dParallel	90.2 (451/500)	52.00	57.65
T3D (Ours)	**97.2** (486/500)	**60.00**	**61.73**

Table 10: Pass@k results on MATH500 using SDAR-4B-Chat with block size 4 and TokPS 2. T3D achieves larger gains as k increases, suggesting that it preserves output diversity and benefits from test-time scaling.

Model	pass@5	pass@10	pass@20
Teacher	81.9	85.6	88.2
Forward-KL	53.1	63.3	71.6
T3D (ours)	66.0	74.2	80.4

- **Exploration behavior on reasoning tasks.** As shown in Table 10, we compare pass@k on MATH500 for the teacher, Forward-KL baseline, and T3D. T3D outperforms Forward-KL at every k. Moreover, the gap between T3D and the teacher narrows as k increases, indicating that T3D preserves meaningful exploration ability rather than collapsing to a narrow set of outputs.
- **Open-ended generation.** On MT-Bench (Appendix D.7), T3D achieves the highest score among few-step methods and halves the severe repetition rate of the original model.

Overall, these results suggest that T3D does not simply reduce diversity in an indiscriminate way. Instead, it preserves useful exploration for reasoning while promoting sharper refinement during decoding. This helps explain why T3D benefits from test-time scaling and consistently improves over Forward-KL under larger pass@k budgets.

D.5 Experiments with Multiple Seeds

To evaluate the stability of T3D, we repeat the main few-step experiments on MATH-500 and MBPP with three random seeds. As shown in Table 11, T3D consistently outperforms all baselines across both benchmarks. On MATH-500, T3D achieves an average accuracy of 46.27 with a standard deviation of only 0.31, indicating both strong performance and low variance across seeds. On MBPP, T3D obtains an average accuracy of 21.87, substantially outperforming the original model and prior few-step DLLM baselines. These results suggest that the gains of T3D are stable and not due to seed-specific variation.

D.6 Cross-Domain Generalization to WinoGrande

Our main experiments evaluate T3D on math and coding benchmarks, which test structured reasoning and executable generation under aggressive few-step decoding. To examine whether T3D generalizes beyond its training domain, we directly evaluate the checkpoints trained on MATH on WinoGrande [33], a commonsense reasoning benchmark, without any task-specific fine-tuning. The model is prompted to think step by step and place its final answer in `\boxed{}`.

As shown in Table 12, the original model reaches 68.51% accuracy under full-step decoding, but under TokPS $= 4$ it can no longer produce well-formed answers: only 1.4% of its responses can be extracted, which limits its accuracy to 1.0%. T3D recovers the extraction rate to 55.6% and achieves the highest accuracy among few-step methods, improving over both dParallel and ReDi. This suggests that the benefits of T3D are not restricted to the math and coding domains it is trained on.

Table 11: Multi-seed results on MATH-500 and MBPP. We report accuracy over three seeds, together with the mean, standard deviation, and sample variance.

Method	MATH-500					MBPP				
	Seed 1	Seed 2	Seed 3	Mean ± Std	Var.	Seed 1	Seed 2	Seed 3	Mean ± Std	Var.
Original	14.80	14.20	12.60	13.87 ± 1.14	1.29	13.00	15.00	13.40	13.80 ± 1.06	1.12
ReDi	24.20	24.20	23.00	23.80 ± 0.69	0.48	6.40	6.80	6.60	6.60 ± 0.20	0.04
dParallel	37.80	38.80	36.20	37.60 ± 1.31	1.72	7.00	6.20	7.00	6.73 ± 0.46	0.21
T3D (ours)	46.20	46.00	46.60	**46.27 ± 0.31**	**0.09**	21.60	21.60	22.40	**21.87 ± 0.46**	0.21

Table 12: Results on WinoGrande with checkpoints trained on MATH. We report the extraction rate and accuracy (%). Under TokPS $= 4$, T3D achieves the highest extraction rate and accuracy among few-step methods.

Model	Original (full-step)	Original	dParallel	ReDi	T3D (ours)
TokPS	1	4	4	4	4
Extraction Rate	100.0	1.4	29.0	49.0	**55.6**
Accuracy	68.51	1.0	17.4	29.7	**31.5**

D.7 Open-Ended Instruction Following

Beyond benchmarks with near-deterministic answers, we evaluate T3D on open-ended instruction following. We train each method on SDAR-4B-Chat using prompts from OpenHermes [39] and evaluate on MT-Bench [57] with a GPT-4 judge. Few-step methods use greedy decoding with block size 8 and 4 steps per block, and the full-step model is reported as an upper bound. We additionally report the percentage of responses with severe repetition.

As shown in Table 13, T3D achieves the highest average score among few-step methods and halves the severe repetition rate of the original model ($6.2\% \rightarrow 3.1\%$). This indicates that the mode-seeking DDO objective does not induce degenerate behavior such as severe repetition in open-ended generation.

D.8 Sensitivity to Teacher Quality

In our main experiments, we collect one teacher trajectory per prompt without correctness filtering, so about 62% of the training trajectories end with a correct answer. To study whether higher-quality teacher data helps, we construct a filtered dataset: for each problem, we sample the teacher eight times at temperature 0.8 and keep a verified-correct trajectory whenever one is available, which raises the fraction of correct trajectories to 81%. As shown in Table 14, training T3D on the filtered dataset does not meaningfully change few-step accuracy. This suggests that T3D mainly benefits from learning the teacher's denoising process, rather than from the correctness of its final outputs alone.

E Ablation Study

In this section, we present ablation studies for our proposed `T3D`. In Appendix E.1, we analyze the effect of the regularization coefficient λ. In Appendix E.2, we examine how different components of our method contribute to preserving the full diffusion decoding behavior. Finally, in Appendix E.3, we present ablations under few-step generation settings to evaluate the contribution of each component to the overall performance of our method. In addition, in Appendix E.4, we decompose trajectory supervision into teacher-generated samples and teacher decoding order.

E.1 The Effectiveness of λ in Training Objective

We conduct an ablation study on the regularization weight λ in Eqn. 14. We run these experiments using the **SDAR-4B-Chat** model and evaluate it on **MATH500** benchmark. Table 15 reports performance under different decoding configurations with varying Tokens Per Step (TokPS), block sizes, and decoding steps.

Table 13: MT-Bench results (GPT-4 judge) on SDAR-4B-Chat. Few-step methods use greedy decoding with block size 8 and 4 steps per block. Severe Rep. denotes the percentage of responses with severe repetition.

Model	Turn 1↑	Turn 2↑	Average↑	Severe Rep. (%)↓
Full-step (upper bound)	8.50	7.18	7.84	0.6
Original	6.19	4.72	5.46	6.2
ReDi	6.10	**5.10**	5.60	5.6
dParallel	5.28	4.85	5.07	5.6
T3D (Ours)	**6.21**	**5.10**	**5.66**	**3.1**

Table 14: Effect of teacher-trajectory filtering on SDAR-4B-Chat with block size 8. + *Filtering* trains T3D on verified-correct trajectories selected from eight teacher samples per problem.

Block Size	Steps/Block	GSM8K		MATH500	
		T3D	+ Filtering	T3D	+ Filtering
8	2	63.99	64.75	44.80	46.00
8	4	81.96	80.14	61.60	62.00

Results. Overall, moderate regularization consistently yields the best or near-best performance across most settings. In particular, $\lambda = 0.2$ achieves the strongest results in the majority of configurations, especially under more aggressive few-step decoding regimes (e.g., higher TokPS). In contrast, a smaller regularization weight ($\lambda = 0.05$) is often insufficient to stabilize training, while overly strong regularization ($\lambda = 0.5$) can lead to degraded performance in several settings. Based on these observations, we fix $\lambda = 0.2$ for all experiments reported in the main results.

E.2 Preserving Full-Step Diffusion Properties

We first examine whether few-step distillation preserves the original full-step diffusion behavior. After training each variant under the few-step distillation setting, we revert the model to the original full-step diffusion decoding process without any additional fine-tuning. This evaluation tests whether the learned model still retains the fine-grained denoising capability of the pretrained diffusion model.

Results. As shown in Table 16, directly applying few-step distillation can substantially degrade full-step diffusion behavior. Both TD and DDO alone perform poorly when the distilled model is reverted to the original full-step decoding process, indicating that optimizing only for compressed decoding may damage the model's fine-grained denoising capability.

Adding the path-consistency loss substantially improves preservation under full-step decoding. For TD, adding $\mathcal{L}_{\text{path}}$ improves accuracy from 22.00 to 58.00, showing that path-level supervision helps retain intermediate denoising behavior. The full T3D objective achieves the best result, reaching 69.00 accuracy and slightly surpassing the original model. These results suggest that T3D improves few-step decoding while preserving the intrinsic diffusion behavior of the pretrained model.

E.3 Ablation Study on Few-Step Generation

We further study how each component affects few-step generation performance. We evaluate SDAR-4B-Chat on MATH500 with block size 8 under two decoding budgets: 4 decoding steps per block and 2 decoding steps per block. The latter corresponds to a more aggressive few-step decoding regime.

Results. As shown in Table 17, trajectory-level distillation is the key factor behind the improvement in few-step generation. TD improves over the original model under both decoding budgets, especially in the more aggressive setting with only 2 decoding steps per block, where accuracy increases from 16.80 to 38.80. This supports our main claim that matching teacher trajectories helps reduce the factorization error induced by large denoising jumps.

Table 15: Ablation study on the effect of the regularization weight λ under different decoding configurations. We report the model performance across varying Tokens Per Step (TokPS), block sizes, and decoding steps. All experiments are done using MATH500 dataset.

TokPS	Block Size	Decoding Steps	$\lambda = 0.05$	$\lambda = 0.2$	$\lambda = 0.5$
1	4	4	67.80	69.00	69.20
1	8	8	62.60	64.80	65.40
2	8	4	57.20	58.60	56.20
4	4	1	47.00	47.20	46.00
4	8	2	40.20	45.20	42.00
8	8	1	7.20	7.60	6.20

Table 16: Ablation results under full-step diffusion decoding on MATH500. All variants are trained for few-step distillation and then evaluated by reverting to the original full-step diffusion process with block size 4 and decoding steps 4 per block.

Method	Objective / Variant	Acc.
Original	–	68.00
SFT	Supervised Fine-Tuning	60.20
ReDi	Endpoint-Style Distillation	50.40
TD	$\mathcal{L}_{\text{traj}}$	22.00
TD + Path Loss	$\mathcal{L}_{\text{traj}} + \lambda \mathcal{L}_{\text{path}}$	58.00
DDO	$\mathcal{L}_{\text{traj-DDO}}$	12.00
T3D (Ours)	$\mathcal{L}_{\text{traj-DDO}} + \lambda \mathcal{L}_{\text{path}}$	**69.00**

DDO further improves few-step performance by replacing the forward-KL trajectory objective with a mode-seeking trajectory-matching objective. Under the aggressive setting with 2 decoding steps per block, the full T3D objective performs best, reaching 45.00 accuracy. This suggests that the benefit of each component becomes more apparent as the decoding budget becomes tighter.

Overall, these results show that trajectory supervision provides the main gain, DDO improves the quality of trajectory matching, and path consistency further stabilizes generation under compressed decoding.

E.4 Which Part of the Trajectory Matters?

Compared with training on ground-truth data, trajectory self-distillation introduces two factors: (1) *teacher-generated samples*, i.e., the student is trained on outputs sampled from the teacher rather than on ground-truth responses; and (2) *teacher decoding order*, i.e., the intermediate states follow the teacher's actual decoding trajectory, including its token-reveal order and mask patterns, rather than independently sampled random masks. Crossing these two factors gives a 2×2 ablation: random order with ground-truth samples (SFT), random order with generated samples (ReDi), generated order with ground-truth samples, and generated order with generated samples (T3D). The third variant keeps the teacher's reveal order but applies it to ground-truth responses, which is closely related to the pseudo-trajectories of d3LLM [31].

We use SDAR-4B-Chat with block size 4 and TokPS $= 2$. As shown in Table 18, neither factor alone is sufficient. ReDi uses teacher samples but remains far behind T3D when the reveal order is randomized, and using the teacher order with ground-truth samples also fails to recover the gain. T3D performs best because it preserves the coupling between each teacher sample and the trajectory that produced it. From the diffusion perspective, this supervises the joint intermediate transitions required for large decoding jumps; from the language-model perspective, it resembles on-policy distillation, where the intermediate states, reveal decisions, and generated tokens all come from the same teacher rollout.

Table 17: Component-wise ablation on few-step generation using SDAR-4B-Chat on MATH500. We evaluate block size 8 with two decoding budgets: 4 and 2 decoding steps per block. Higher accuracy is better.

Method	Objective / Variant	BS = 8 DS = 4	BS = 8 DS = 2
Original	–	49.60	16.80
SFT	Supervised fine-tuning	54.44	40.20
ReDi	Endpoint-style distillation	23.60	20.20
TD	$\mathcal{L}_{\text{traj}}$	52.60	38.80
TD + Path Loss	$\mathcal{L}_{\text{traj}} + \lambda\mathcal{L}_{\text{path}}$	49.40	37.20
DDO	$\mathcal{L}_{\text{traj-DDO}}$	52.22	43.20
T3D (Ours)	Full objective	**60.60**	**45.00**

Table 18: Decomposing trajectory supervision into teacher-generated samples and teacher decoding order, using SDAR-4B-Chat with block size 4 and TokPS = 2.

Method	Teacher Samples	Teacher Order	MATH500	GSM8K
Original Multi-Step Teacher	–	–	54.40	78.77
SFT	✗	✗	54.60	54.60
ReDi	✓	✗	41.00	73.62
Generated Order + Ground Truth	✗	✓	32.60	62.77
T3D (Ours)	✓	✓	**60.00**	**83.85**

F Additional Related Work

Few-step distillation of masked diffusion models. SDTT [10] trains the student to match the teacher's multi-step predictive distributions from the same noisy state, which requires the teacher's output distributions during training, while T3D only requires the teacher's decoded tokens and their decoding order. d3LLM [31] applies the teacher's unmasking order to ground-truth responses to form pseudo-trajectories and targets decoding multiple blocks in parallel, whereas T3D trains on the teacher's actual trajectories to reduce the number of steps within each block. Another line of work reduces factorization error by changing the model itself: Di4C [14] distills the teacher into a mixture model that captures dimensional correlations, VADD [46] introduces latent variables into the denoiser, and the concurrent IMDM [52] replaces the deterministic mask state with a stochastic one. T3D keeps the standard MDLM unchanged and is complementary to these approaches.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims match the paper's contributions, theoretical analysis, and empirical results, without going beyond the demonstrated scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our method, particularly those inherent to self-distillation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We state the assumptions and provide the full proofs in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide thorough implementation details and experimental settings in the main text and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The T3D implementation, data preparation instructions, training and evaluation instructions, and a distilled SDAR-4B-Chat checkpoint are publicly available on GitHub.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide all training and evaluation details in the main text and appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report multi-run results for key experiments (e.g., Math500 and MBPP with SDAR) and include the corresponding variance statistics in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the compute setup and report training cost estimates based on 8×A100 GPUs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, the research complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This paper focuses on a methodological advance in few-step diffusion language modeling and does not include a dedicated discussion of broader societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release new foundation models or datasets, but studies a training framework built on existing open-source models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit the original assets used in the paper and explicitly specify their licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the T3D implementation and a distilled SDAR-4B-Chat checkpoint. The repository provides installation, data preparation, training, and evaluation instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.