TRIMS: Trajectory-Ranked Instruction Masked Supervision for Diffusion Language Models

Lingjie Chen*
UIUC

Ruizhong Qiu
UIUC

Yuyu Fan
Independent Researcher

Yanjun Zhao
UIUC

Hanghang Tong
UIUC

Abstract

Diffusion language models (DLMs) offer a promising path toward low-latency generation through parallel decoding, but their practical efficiency depends heavily on the decoding trajectory. In practice, this advantage often fails to fully materialize because standard training does not provide explicit supervision over token reveal order, creating a train–inference mismatch that leads to suboptimal decoding behavior. We propose **T**rajectory-**R**anked **I**nstruction **M**asked **S**upervision (**TRIMS**), a simple trajectory-guided supervised fine-tuning framework that injects trajectory supervision into standard Masked Diffusion Language Model (MDLM) training with minimal overhead. Instead of relying on costly DLM-based distillation, **TRIMS** uses lightweight signals from an autoregressive teacher to guide a trajectory-aware masking strategy, encouraging the model to learn more effective decoding orders. Experiments on LLaDA and Dream across math and coding benchmarks show that **TRIMS** significantly improves the accuracy-parallelism trade-off over both standard MDLM training and train-free acceleration baselines, while achieving competitive performance with prior distillation-based approaches at substantially lower training cost. Further analysis shows that **TRIMS** leads to better decoding trajectories, validating the effectiveness of trajectory-guided supervision for DLMs.

1 Introduction

Diffusion language models (DLMs) offer a promising alternative to autoregressive (AR) language models (Austin et al., 2021a; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Arriola et al., 2025). Unlike AR models, which generate tokens in a fixed left-to-right order, DLMs can generate tokens in arbitrary orders, enabling global iterative refinement (Wang et al., 2025a; Havasi et al., 2025). Moreover, the diffusion paradigm supports parallel token generation, which is a key advantage over AR models and has motivated a growing line of efficient decoding methods (Wu et al., 2026b;a; Ma et al., 2025; Ben-Hamu et al., 2025). Recent work on scaled DLMs has demonstrated their potential to achieve higher throughput and lower latency than AR models (Labs et al., 2025).

However, in practice, the benefits of parallel decoding do not always fully materialize. Models may fall back to next-token prediction or deliberately skip difficult tokens during inference (Ni et al., 2026; Fu et al., 2025), substantially limiting parallelism and wasting the parallel capacity afforded by bidirectional attention. Ideally, the model should learn to organize its decoding trajectory to maximize both parallelism and efficiency.

A key reason behind this limitation is the *mismatch between training and inference*. Existing DLMs are typically trained with a uniform random masking strategy (Sahoo et al., 2024; Shi et al., 2024). Under this training paradigm, the noise schedule is sample-agnostic, each token is masked with the same probability, and the model is trained by optimizing the

*Correspondence to lingjie7@illinois.edu.

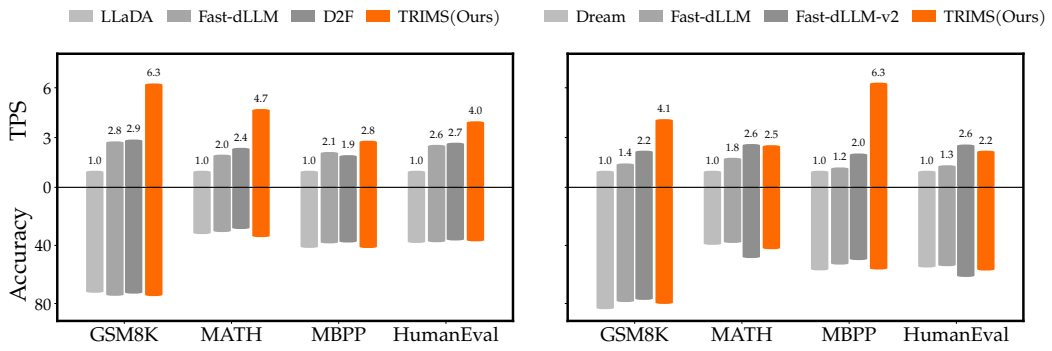


Figure 1: **TRIMS** improves the accuracy-parallelism trade-off on both LLaDA-Instruct (left) and Dream-Instruct (right), achieving higher TPS (tokens predicted per step) while maintaining competitive accuracy, with much lower training cost than distillation-based methods.

evidence lower bound (ELBO). As a result, the model does not receive explicit supervision about which tokens should be resolved earlier and which can be left to later steps. Recent works have begun exploring trajectory-aware training for DLMs, but many of them rely on costly trajectory sampling from the DLM itself for distillation (Chen et al., 2025; Qian et al., 2026). While effective, such methods inherit the high cost of diffusion sampling, limiting their practicality.

Motivated by this mismatch and the limitations of existing methods, we propose **T**rajectory-**R**anked **I**nstruction **M**asked **S**upervision (**TRIMS**), a simple trajectory-guided supervised fine-tuning framework for DLMs. The core idea is to replace the standard random masking strategy with a trajectory-aware masking strategy that injects trajectory information into the training process. Instead of collecting trajectory signals from costly diffusion sampling, **TRIMS** uses a much lighter alternative: an autoregressive teacher estimates token-level difficulty with a single teacher-forcing pass over the training data. We then discretize these difficulty scores into buckets and construct a *hard-to-easy* masking strategy. This design encourages the model to learn a hard-to-easy decoding order that improves parallelism while keeping the training pipeline simple and unchanged from standard MDLM training.

We evaluate **TRIMS** on two open-source DLMs, LLaDA-Instruct and Dream-Instruct, across four widely used math and coding benchmarks. The results show that **TRIMS** achieves up to 3× higher parallelism while maintaining similar accuracy compared to train-free acceleration baselines and standard MDLM training. Moreover, **TRIMS** introduces trajectory supervision with almost no additional cost over standard MDLM training, requiring only 3 hours on 8× A100 GPUs, while achieving performance comparable to distillation-based methods, which typically require costly diffusion sampling to generate separate distillation datasets for each target model. Ablation studies confirm the contribution of each component, and trajectory analysis shows that models trained with **TRIMS** exhibit more efficient decoding trajectories, validating the effectiveness of trajectory-guided supervision.

Our main contributions are as follows:

- We propose a trajectory-guided SFT framework that addresses the train–inference mismatch in DLMs by using token-level difficulty from an AR teacher to construct trajectory-aware supervision.
- We introduce a trajectory-aware masking strategy that simulates hard-to-easy decoding with minimal changes to standard MDLM training.
- We show that **TRIMS** achieves competitive performance with much lower overhead than prior trajectory optimization methods, requiring only a single AR forward pass and no costly diffusion sampling.

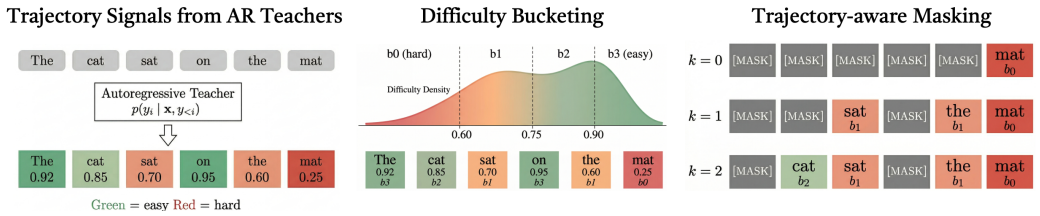


Figure 2: **TRIMS** consists of three stages: (1) an AR teacher estimates token-level difficulty scores, (2) scores are discretized into ordered buckets, and (3) bucket assignments drive trajectory-aware masking that simulates a hard-to-easy decoding order during training.

2 Preliminaries

Masked Diffusion Language Models (MDLMs). Unlike autoregressive language models that generate tokens in a strict left-to-right order, masked diffusion language models (MDLMs) formulate generation as an iterative denoising process over partially masked sequences (Austin et al., 2021a; Shi et al., 2024). Given a clean sequence $x_0 = (x_0^1, \ldots, x_0^L)$, the forward corruption process randomly replaces tokens with a special [MASK] symbol. At corruption level $t \in [0, 1]$, the corrupted sequence x_t is sampled as:

$$q(x_t \mid x_0) = \prod_{i=1}^{L} \left[(1-t)\, \mathbf{1}[x_t^i = x_0^i] + t\, \mathbf{1}[x_t^i = \texttt{[MASK]}] \right].$$

Here, $\mathbf{1}[\cdot]$ denotes the indicator function. The reverse model p_θ learns to recover tokens from the corrupted sequence. At each step, the model predicts all masked tokens in parallel:

$$p_\theta(x_0 \mid x_t) = \prod_{i : x_t^i = \texttt{[MASK]}} p_\theta(x_0^i \mid x_t).$$

The training objective minimizes the negative log-likelihood over masked positions:

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

Iterative Decoding and Trajectory. During inference, MDLMs decode sequences through iterative refinement rather than a single one-shot prediction. At intermediate decoding steps, some tokens have already been resolved while others remain masked for later prediction. This induces an implicit token reveal order during generation, which we refer to as the decoding trajectory. Different trajectories can lead to different levels of token predictability in later steps, and therefore to different accuracy–parallelism trade-offs. Our method builds on this view by introducing training signals that encourage more effective reveal orders.

3 Method

This section introduces the proposed **TRIMS**. We first describe how we derive token-level trajectory signals from an autoregressive teacher, and then show how these signals are incorporated into training through a trajectory-aware masking strategy. This design injects trajectory supervision into standard MDLM training with minimal overhead and avoids costly diffusion sampling. Figure 2 illustrates the overall three-stage pipeline of **TRIMS**.

3.1 Trajectory Signals from AR Teachers

To obtain trajectory supervision without expensive sampling, we construct token-level trajectory signals using an AR teacher model. Instead of explicitly observing decoding trajectories through model rollouts, we approximate token priority using prediction statistics from an AR model, which only requires a single forward pass over the training data.

Algorithm 1 Trajectory-Ranked Instruction Masked Supervision

```
// Differences from standard MDLM training are highlighted in brown.
```

Input: Training data $\mathcal{D}$ with bucket ids b_i, model x_θ with parameters θ, corruption process q, noise schedule α_t, and masking probabilities p_{context} and p_{future}
repeat
  Sample $(\mathbf{x}^{1:L}, \mathbf{b}^{1:L})$ i.i.d. from $\mathcal{D}$
  Sample $t \sim \mathcal{U}[0,1]$
  Compute $\alpha_t, \dot{\alpha}_t$
  Sample bucket threshold $k \sim \mathcal{U}\{0, K-1\}$
  Sample $\mathbf{z}_t^{1:L} \sim q(\mathbf{z}_t^{1:L} \mid \mathbf{x})$
  for $\ell = 1$ **to** L **do**
    if $b^\ell > k$ **then**
      mask z_t^ℓ with probability p_{future}
    else
      mask z_t^ℓ with probability p_{context}
    end if
  end for
  Compute $x_\theta(\mathbf{z}_t^{1:L})$
  $\mathcal{L}^{\text{MDM}}(\theta) \leftarrow \frac{\dot{\alpha}_t}{1-\alpha_t} \sum_{\ell=1}^{L} \delta_{z_t^\ell, m} \log \left\langle x_\theta^\ell(\mathbf{z}_t^{1:L}), x^\ell \right\rangle$
  Perform gradient descent step on $\mathcal{L}^{\text{MDM}}(\theta)$
until converged
Return θ

Given a prompt–completion pair (x, y), we run the AR teacher in teacher-forcing mode. For each completion token y_i, the teacher provides the conditional probability $p_T(y_i \mid x, y_{<i})$, from which we compute the token difficulty score as:

$$s_i = -\log p_T(y_i \mid x, y_{<i}),$$

where larger values indicate harder tokens for the teacher to predict.

3.2 Difficulty Bucketing

Rather than using raw continuous difficulty scores, we discretize them into K ordered buckets via quantile-based thresholds. This choice is motivated by two considerations. First, quantile-based thresholds are invariant to the absolute scale of NLL scores, making the method robust to score distribution shifts across datasets or AR teachers. Second, equal-mass quantiles ensure each bucket contains roughly the same number of tokens, keeping all trajectory stages balanced and preventing any single stage from dominating training.

Specifically, we collect difficulty scores of all completion tokens across the training data and compute quantile thresholds that divide the score distribution into K equal-mass intervals. Each token is assigned a bucket id $b_i \in \{0, \ldots, K-1\}$ by its quantile rank, where harder tokens (higher s_i) receive lower bucket ids, placing them into earlier decoding stages.

3.3 Trajectory-aware Masking

Using the bucket assignments, we construct a trajectory-aware masking strategy that simulates iterative decoding during training. Our key intuition is to *resolve difficult tokens earlier in the decoding process*. This benefits parallel decoding in two ways: (1) once difficult tokens are determined, the remaining tokens often become more predictable, enabling more tokens to be generated in parallel; and (2) resolving difficult tokens early, while many positions remain masked, preserves sufficient flexibility for subsequent steps to refine the remaining tokens around these anchored positions.

Let b_i denote the bucket id of token y_i, where smaller bucket ids correspond to harder tokens. During training, we first sample a bucket threshold k uniformly from $\{0, \ldots, K-1\}$. Tokens satisfying $b_i \leq k$ are treated as already available context, while tokens with $b_i > k$

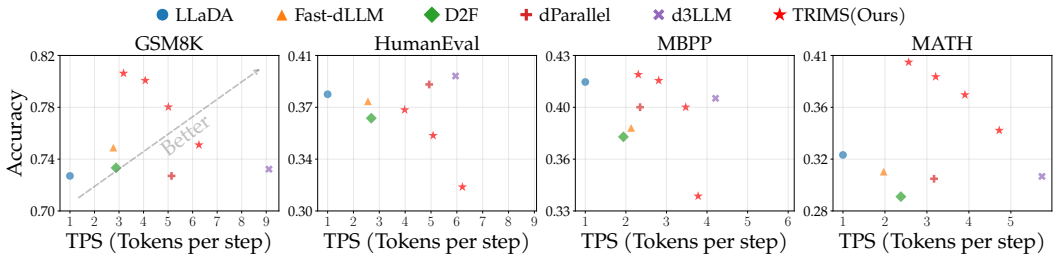


Figure 3: Accuracy-parallelism trade-off of **TRIMS** and baselines on LLaDA-Instruct. The upper-right region indicates better performance with higher accuracy and parallelism.

Method	GSM8K		MATH		MBPP		HumanEval	
	Acc(↑)	TPS(↑)	Acc(↑)	TPS(↑)	Acc(↑)	TPS(↑)	Acc(↑)	TPS(↑)
LLaDA	72.6	1.00	32.2	1.00	41.7	1.00	38.3	1.00
Fast-dLLM	74.7	2.77	30.8	1.97	38.6	2.13	37.8	2.56
D2F	73.2	2.88	28.7	2.38	38.0	1.94	36.6	2.69
dParallel	72.6	5.14	30.2	3.17	40.0	2.35	39.0	4.93
d3LLM	73.1	**9.11**	30.4	**5.74**	40.6	**4.21**	**39.6**	**5.95**
TRIMS	**74.9**	6.26	**34.3**	4.72	**41.8**	2.81	37.2	3.98

Table 1: Comparison of **TRIMS** with baseline methods on LLaDA-Instruct. TPS denotes tokens per step and higher values indicate better parallelism.

are treated as future tokens that should remain masked. To emphasize learning on these future tokens while maintaining robustness, we use different masking probabilities for the two groups. Formally, the masking probability m_i for token y_i is defined as:

$$m_i = \begin{cases} p_{\text{context}} & \text{if } b_i \leq k, \\ p_{\text{future}} & \text{if } b_i > k, \end{cases},$$

where $p_{\text{future}} = 0.95$ ensures future tokens are nearly always masked, and $p_{\text{context}} = 0.05$ introduces a small amount of masking on context tokens for robustness, preventing the model from overfitting to perfectly revealed context during training.

To reduce the mismatch with the pretraining stage and maintain training stability, we apply the trajectory-aware masking scheme to only 10% of the training data. The remaining data follow the standard MDLM masking process, and we keep the original ELBO objective unchanged. Algorithm 1 summarizes the overall training procedure of **TRIMS**.

4 Experiments

4.1 Experimental Details

In this section, we evaluate **TRIMS** on two open-source DLMs and analyze its effectiveness, efficiency, and impact on decoding trajectories.

Dataset Preparation. We use the s1K dataset (Muennighoff et al., 2025) for training. Reasoning data typically produces longer completions, making it a natural choice for evaluating the parallel decoding behavior of DLMs. For the AR teacher, we use Qwen3-8B (Yang et al., 2025a) to compute the token-level difficulty scores.

Model and Training Details. In all experiments, we use LLaDA-Instruct (Nie et al., 2025) and Dream-Instruct (Ye et al., 2025) as the backbone models and implement **TRIMS** in the dLLM codebase (Zhou et al., 2026). We use a sequence length of 1,024, train for 30 epochs with a batch size 32 and the gradient accumulation factor 4, and set the learning rate at 1×10^{-4}. The loss is computed only on response tokens. We apply LoRA (Hu et al., 2022) with rank $r = 32$, scaling factor $\alpha = 32$, and the weight decay 0.1. We use a cosine learning

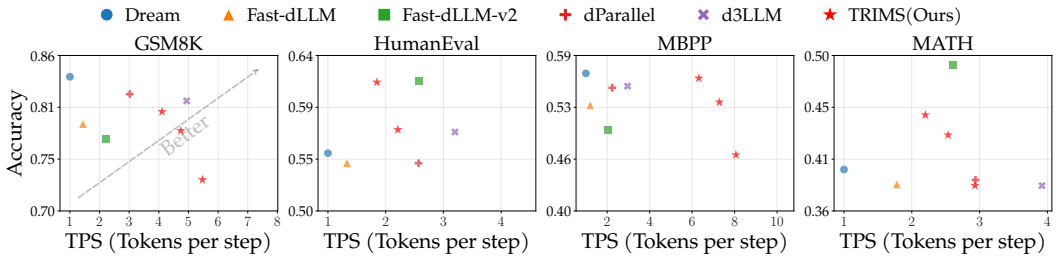


Figure 4: Accuracy-parallelism trade-off of **TRIMS** and baselines on Dream-Instruct.

Method	GSM8K		MATH		MBPP		HumanEval	
	Acc(↑)	TPS(↑)	Acc(↑)	TPS(↑)	Acc(↑)	TPS(↑)	Acc(↑)	TPS(↑)
Dream	**83.9**	1.00	39.6	1.00	**57.2**	1.00	55.2	1.00
Fast-dLLM	79.0	1.44	38.3	1.78	53.2	1.20	54.3	1.33
Fast-dLLM-v2	77.5	2.21	**48.7**	2.61	50.1	2.04	**61.7**	2.58
dParallel	82.1	3.02	38.7	2.94	55.4	2.24	54.3	2.57
d3LLM	81.4	**4.94**	38.2	**3.92**	55.6	2.96	57.1	**3.20**
TRIMS	80.3	4.11	42.6	2.54	56.6	**6.31**	57.3	2.21

Table 2: Comparison of **TRIMS** with baseline methods on Dream-Instruct. TPS denotes tokens per step and higher values indicate better parallelism.

rate schedule with warmup ratio 0.03. Training is conducted on 8×NVIDIA A100 80GB GPUs with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020) enabled.

Baselines. To evaluate the effectiveness of **TRIMS**, we compare it with four groups of baselines: (1) the original DLM backbones, LLaDA-Instruct (Nie et al., 2025) and Dream-Instruct (Ye et al., 2025); (2) train-free acceleration methods, Fast-dLLM (Wu et al., 2026b); (3) training-based acceleration methods, Fast-dLLM-v2 (Wu et al., 2026a) and D2F (Wang et al., 2025b); and (4) distillation-based methods, dParallel (Chen et al., 2025) and d3LLM (Qian et al., 2026). For evaluation, we report results on four widely used benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021b). These benchmarks assess core mathematical and coding abilities, and their relatively long solution sequences make them suitable for evaluating parallel decoding. We report both accuracy and parallelism, where parallelism is measured by tokens predicted per step (TPS). **TRIMS** uses the confidence-based sampling strategy from Fast-dLLM (Wu et al., 2026b).

4.2 Main Results

LLaDA Results. As shown in Table 1, **TRIMS** achieves strong parallelism gains across benchmarks, reaching 6.26 TPS on GSM8K and 4.72 TPS on MATH while maintaining competitive accuracy. The speedup on coding benchmarks is more modest, since the rigid syntactic structure of code makes tokens more predictable, leaving less room for trajectory supervision to improve parallel decoding. Compared with train-free baselines, **TRIMS** consistently improves parallelism while preserving accuracy. Compared with distillation-based methods, **TRIMS** achieves competitive results on the accuracy-parallelism trade-off. We note that d3LLM achieves higher raw TPS on some benchmarks; however, as shown in Table 3, this comes at the cost of 287 GPU hours of curation compute per target model and 93K training examples, nearly two orders of magnitude more data than **TRIMS**'s 1K samples. The TPS gap therefore reflects a deliberate resource-efficiency trade-off rather than a fundamental capability difference. Figure 3 shows the full accuracy-parallelism trade-off curve. The frontier of **TRIMS** lies mostly in the upper-right region, indicating a favorable balance between the accuracy and parallelism relative to all baselines.

Dream Results. As shown in Table 2, **TRIMS** achieves notable parallelism gains across benchmarks, with a particularly strong result on MBPP at 6.31 TPS, corresponding to up to 3× higher parallelism than the train-free baselines. One possible reason is that MBPP often contains relatively long prompts and test cases, where the trajectory learned by **TRIMS** can be more fully exploited. Compared with train-free baselines, **TRIMS** consistently improves parallelism while maintaining similar accuracy. Compared with distillation-based methods, **TRIMS** achieves competitive results on the accuracy-parallelism trade-off; any remaining TPS gap with d3LLM reflects the same resource-efficiency difference discussed above. Overall, the trend on Dream-Instruct is slightly different from that on LLaDA-Instruct. We hypothesize that this difference comes from Dream-Instruct being adapted from an autoregressive model (Gong et al., 2025), which may lead to different behavior after diffusion post-training. Figure 4 shows the full accuracy-parallelism trade-off curve. The frontier of **TRIMS** lies mostly in the upper-right region, indicating a favorable balance between the accuracy and parallelism relative to all baselines.

Method	Supervision	Curation Compute	Train Data	Epochs
TRIMS	AR teacher	0.6 GPU Hours	1K	30
dParallel	DLM distillation	287 GPU Hours * N(model)	93K	6
d3LLM	DLM distillation	287 GPU Hours * N(model)	93K	6

Table 3: Training overhead comparison. All GPU hours are measured on NVIDIA A100-80GB. Distillation-based methods require expensive DLM sampling for dataset curation and are trained on substantially larger datasets. **TRIMS** needs almost no data curation cost with far less training data requirements.

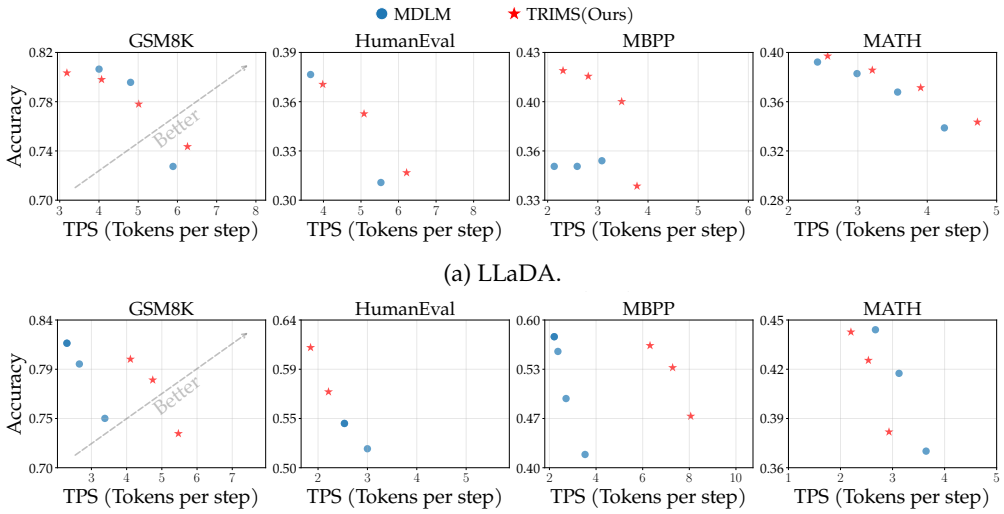
Training Overhead. Table 3 compares the training cost of **TRIMS** with distillation-based methods. The cost of **TRIMS** is close to standard MDLM training, requiring only a single teacher-forcing pass from an AR model over the training data. By contrast, distillation-based methods such as d3LLM and dParallel require additional diffusion sampling to construct distillation datasets. Unlike autoregressive decoding, DLM sampling relies on bidirectional attention and lacks full KV cache support and related low-level optimizations, leading to substantially higher time complexity per sampling step. Moreover, these methods typically need to construct a separate distillation dataset for each target model. In our setting, **TRIMS** uses only 1K training examples versus 93K for distillation-based methods, nearly two orders of magnitude less curated data, further highlighting the data efficiency of **TRIMS**. Details of the training cost calculation are provided in Appendix C.

4.3 Ablation Studies

We conduct ablation studies to isolate the contribution of each component in **TRIMS**, covering trajectory supervision, difficulty-based token ordering, bucket-based discretization, and difficulty metrics. Unless otherwise specified, all ablation experiments use the same training and evaluation settings as the main results.

Effect of Trajectory Supervision. We first evaluate whether introducing trajectory supervision improves performance over standard MDLM training. Figure 5 compares **TRIMS** with standard MDLM training on both LLaDA and Dream. **TRIMS** consistently outperforms the standard training setup on most benchmarks, showing that explicit trajectory supervision indeed improves the accuracy-parallelism trade-off. The gains are especially clear on coding benchmarks, including HumanEval and MBPP, where **TRIMS** achieves both higher accuracy and stronger parallelism. On LLaDA, the improvement is consistent across all benchmarks. On Dream, MATH shows similar accuracy but slightly lower parallelism, possibly because Dream-Instruct is adapted from an AR model already well-trained on mathematical tasks, making trajectory post-training slightly out-of-distribution for this domain.

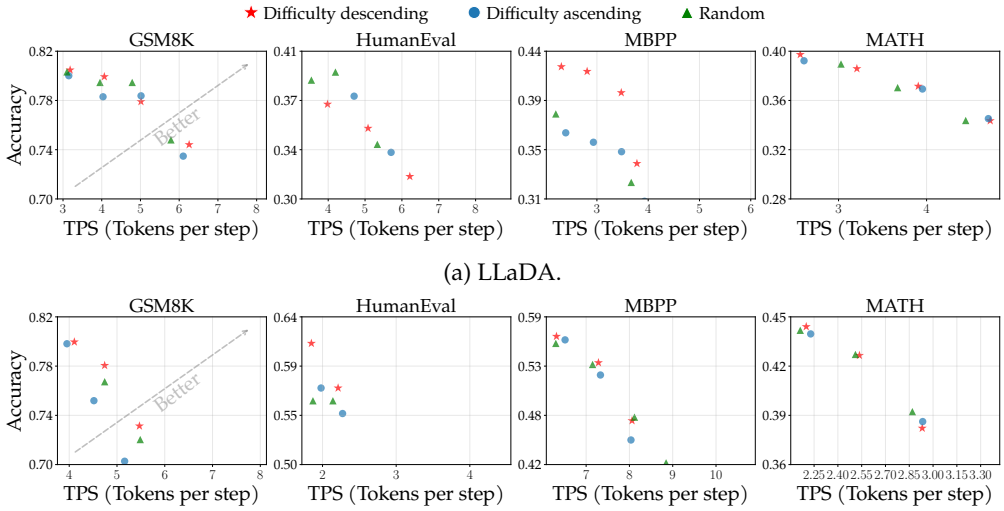
Effect of Trajectory Ordering. We compare three strategies for assigning tokens to buckets: difficulty-descending (harder tokens in earlier buckets), difficulty-ascending (easier tokens first), and random (uniform bucket assignment). Figure 6 shows the results of the



(a) LLaDA.

(b) Dream.

Figure 5: Effect of trajectory supervision compared with standard MDLM training. **TRIMS** consistently improves the accuracy-parallelism trade-off on most benchmarks, with especially clear gains on coding tasks.



(a) LLaDA.

(b) Dream.

Figure 6: Effect of trajectory ordering under random, hard-to-easy, and easy-to-hard supervision. Hard-to-easy achieves the best overall performance, while all orderings improve over standard MDLM training.

trajectory-ordering comparison. The difficulty-descending setting achieves the best overall performance on most benchmarks, although the margin over the other two settings is sometimes modest. Notably, comparing Figure 5 and Figure 6, even random ordering improves over standard MDLM training on most benchmarks, indicating that the bucket-based masking structure itself provides value through trajectory-aware supervision, regardless of ordering. The difficulty-descending ordering yields an additional boost by providing a more informative training signal that aligns with the natural hard-to-easy decoding pattern. This decomposition is also consistent with concurrent work (Ma et al., 2026).

Effect of Bucket Granularity. We next study another core module in the method: the bucket-based masking mechanism. We vary the number of difficulty buckets $K \in \{4, 8, 16\}$. As shown in Figure 8 (Appendix A), $K = 8$ achieves the best performance, with $K = 4$ and $K = 16$ being slightly worse. This is possibly because too many buckets lead to overly fine-grained trajectory stages that are hard for the model to distinguish, while too few buckets merge tokens of meaningfully different difficulty into the same stage, losing discriminative signal. Overall, the performance difference across bucket numbers is modest, indicating robustness to this hyperparameter.

Effect of Difficulty Metrics. We compare negative log-likelihood (NLL) and entropy as difficulty metrics from the autoregressive teacher. As shown in Figure 9 (Appendix B), the two metrics produce comparable results on mathematical benchmarks, while NLL yields a noticeably better accuracy-parallelism trade-off on coding tasks (HumanEval and MBPP). A possible explanation is that coding tasks have more rigid syntactic structure, where NLL's token-specific confidence signal can more precisely distinguish structurally critical tokens from predictable ones, leading to a more informative bucket assignment. Overall, the difference remains modest, indicating that **TRIMS** is robust to the choice of difficulty metric.

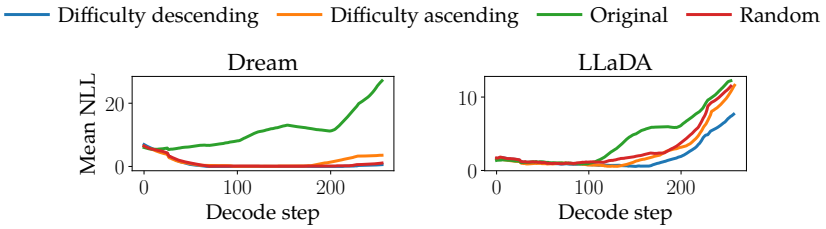


Figure 7: Per-step token NLL during decoding across **TRIMS** variants and original DLM baselines, showing how uncertainty evolves over the generation process.

4.4 Trajectory Analysis

To better understand how **TRIMS** improves the accuracy-parallelism trade-off, we analyze the decoding trajectories of the trained models. Specifically, we study whether the learned trajectories reduce token uncertainty more efficiently during sampling.

Implementation Details. We use Qwen3-8B (Yang et al., 2025a) to compute the token-level NLL of generated sequences at each decoding step, and we evaluate on 40 randomly sampled examples from GSM8K (Cobbe et al., 2021). We compare four settings: the original DLM models (LLaDA-Instruct (Nie et al., 2025) and Dream-Instruct (Ye et al., 2025)) and models trained with three **TRIMS** masking strategies: difficulty-descending, difficulty-ascending, and random ordering. We set `max_new_tokens` and `steps` to 256, and keep all other generation hyperparameters the same as in the main evaluation.

Results. As shown in Figure 7, all three **TRIMS** variants exhibit consistently lower NLL than the original models throughout decoding, except at the very early steps where all settings start from similar high-uncertainty states. This gap indicates that **TRIMS** reduces token uncertainty faster during decoding, leading to more efficient trajectories. Among the variants, the difficulty-descending setting yields the lowest NLL across most steps, consistent with our design intuition: by resolving difficult tokens first, the model anchors high-uncertainty positions early, making the remaining tokens easier to predict in parallel at later steps.

5 Conclusion

In this paper, we present **T**rajectory-**R**anked **I**nstruction **M**asked **S**upervision (**TRIMS**), a simple yet effective trajectory guided supervised fine tuning framework for DLMs that

mitigates the train inference mismatch by incorporating token level difficulty signals and a trajectory aware masking strategy. Experiments demonstrate that **TRIMS** consistently improves the accuracy and parallelism trade off over standard MDLM training and acceleration baselines, achieving performance competitive with distillation based methods while requiring substantially lower training cost and data. Our results further suggest that lightweight trajectory supervision leads to more effective decoding behaviors, highlighting its potential as a practical and scalable approach for enhancing parallel decoding in DLMs. Future work includes extending this framework to larger scale models and exploring richer forms of trajectory signals.

6 Acknowledgement

This work is supported by NSF (2324770). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021a.

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

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, 2025.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

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

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

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

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, 2024.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations*, 2023.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, 2025.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Variable length discrete flow matching with sequence-level edit operations. In *Advances in Neural Information Processing Systems*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, 2021.

Zachary Horvitz, Raghav Singhal, Hao Zou, Carles Domingo-Enrich, Zhou Yu, Rajesh Ranganath, and Kathleen McKeown. Rethinking reasoning with masked diffusion models. In *International Conference on Learning Representations*, 2026. OpenReview: 2mkYB5KUwa.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Daniel Israel, Tian Jin, Ellie Cheng, Guy Van den Broeck, Aditya Grover, Suvinay Subramanian, and Michael Carbin. Planned diffusion. In *International Conference on Learning Representations*, 2026.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025. URL https://arxiv.org/abs/2506.17298.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Pengxiang Li, Dilxat Muhtar, Lu Yin, Tianlong Chen, and Shiwei Liu. Why diffusion language models struggle with truly parallel (non-autoregressive) decoding? *arXiv preprint arXiv:2602.23225*, 2026.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2024.

Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*, 2025.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Linrui Ma, Yufei Cui, Kai Han, and Yunhe Wang. Mask is what dllm needs: A masked data training paradigm for diffusion llms, 2026. URL https://arxiv.org/abs/2603.15803.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. In *Advances in Neural Information Processing Systems*, 2025.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. In *Advances in Neural Information Processing Systems*, 2022.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

John Nguyen, Marton Havasi, Tariq Berrada, Luke Zettlemoyer, and Ricky T. Q. Chen. OneFlow: Concurrent mixed-modal and interleaved generation with edit flows. *arXiv preprint arXiv:2510.03506*, 2025.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Why arbitrary order limits reasoning potential in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, 2025.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, 2025.

Julianna Piskorz, Alvaro Correia, Cristina Pinneri, Motasem Alfarra, Risheek Garrepalli, and Christos Louizos. Masks can be distracting: On context comprehension in diffusion language models. In *International Conference on Learning Representations*, 2026. OpenReview: CdJwNTisx1.

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

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, 2020.

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

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *International Conference on Learning Representations*, 2023.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. Spg: Sandwiched policy gradient for masked diffusion language models. In *International Conference on Learning Representations*, 2026. OpenReview: 18j5Q49GwN.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems*, 2025a.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing, 2025b. URL https://arxiv.org/abs/2508.09192.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. In *International Conference on Learning Representations*, 2026a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations*, 2026b.

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

Jingyi Yang, Guanxu Chen, Xuhao Hu, and Jing Shao. Taming masked diffusion language models via consistency trajectory reinforcement learning with fewer decoding step. *arXiv preprint arXiv:2509.23924*, 2025b.

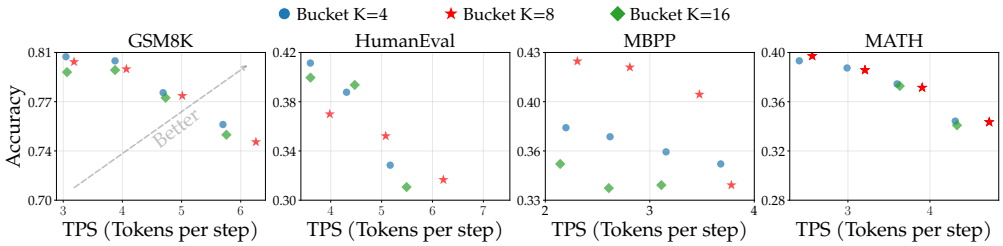
Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis, 2025. URL https://arxiv.org/abs/2502.01718.

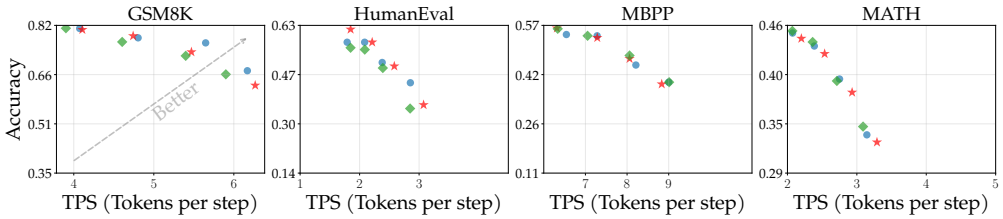
Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. In *International Conference on Learning Representations*, 2025.

Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling, 2026. URL https://arxiv.org/abs/2602.22661.

A Effect of Bucket Granularity



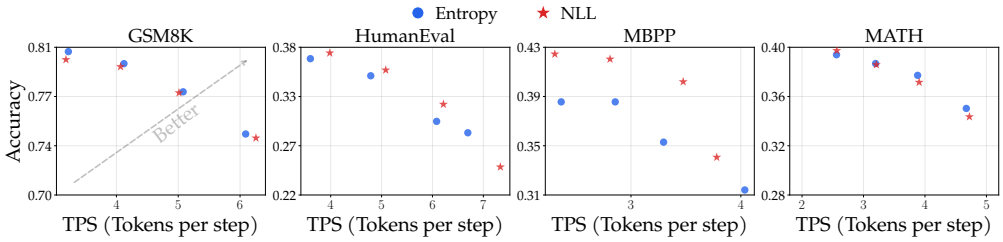
(a) LLaDA.



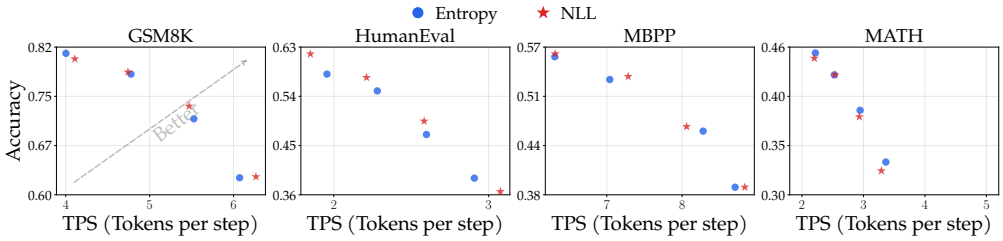
(b) Dream.

Figure 8: Effect of bucket granularity with $K \in \{4, 8, 16\}$ difficulty buckets. $K = 8$ achieves the best performance, while all settings yield similar results, indicating robustness to this hyperparameter.

B Effect of Difficulty Metrics



(a) LLaDA.



(b) Dream.

Figure 9: Comparison of NLL and entropy as difficulty metrics. The two metrics produce comparable results on math benchmarks, while NLL yields a noticeably better accuracy-parallelism trade-off on coding tasks.

C Training Cost Calculation

We estimate the training cost of distillation-based methods (d3LLM (Qian et al., 2026) and dParallel (Chen et al., 2025)) based on their reported dataset construction procedure, which we summarize below for completeness. Note that d3LLM uses the same distillation dataset as dParallel, so the curation cost is shared.

The dataset is constructed by generating teacher trajectories using a semi-autoregressive remasking strategy (Qian et al., 2026): for each prompt, the teacher produces a response trajectory, which is split into contiguous blocks. Training examples are then created by masking an "active block" while keeping earlier blocks as context and later blocks fully masked, simulating intermediate decoding states for certainty-forcing distillation.

For LLaDA-8B-Instruct (Nie et al., 2025), prompts are sampled from GSM8K (Cobbe et al., 2021), PRM12K (Lightman et al., 2024), and a subset of Numina-Math (Li et al., 2024). The teacher generates trajectories with sequence length 256 and block length 32, followed by filtering for correctness, yielding approximately 92K samples. For Dream-7B-Instruct (Ye et al., 2025), the same strategy is applied with additional code data from a subset of Ace-Code (Zeng et al., 2025) ($\sim$10K prompts). Notably, for Dream, standard random masking is used during training instead of semi-autoregressive masking, as the latter risks degenerating toward autoregressive behavior (Qian et al., 2026).

All training tokens in these methods are self-generated without external target completions. All GPU hour estimates are based on NVIDIA A100 80GB GPUs. By contrast, **TRIMS** uses only 1K existing prompt-response pairs with no trajectory generation, making the curation cost negligible.

D Related Work

Diffusion Language Models. Diffusion models, originally developed for continuous domains (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021), have been extended to discrete text via absorbing-state (Austin et al., 2021a) and uniform-state (Hoogeboom et al., 2021) formulations. Continuous-time extensions (Campbell et al., 2022), score-based (Sun et al., 2023; Meng et al., 2022), and ratio-based (Lou et al., 2024) objectives further unify the theory. Masked diffusion language models (MDLMs) simplify the forward process to independent token masking, with recent work clarifying equivalences and simplifying training (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025; Zheng et al., 2025). Alternative directions include continuous diffusion in embedding space (Li et al., 2022; Gong et al., 2023; Dieleman et al., 2022; Lin et al., 2023), flow matching and edit-based methods (Gat et al., 2024; Havasi et al., 2025; Nguyen et al., 2025), block diffusion that interpolates between AR and diffusion decoding (Arriola et al., 2025), and hybrid AR–diffusion architectures for speculative drafting (Christopher et al., 2025), planned generation (Israel et al., 2026), or unified draft-and-verify passes (Liu et al., 2025). More recent works have scaled DLMs substantially. DiffuLLaMA (Gong et al., 2025) demonstrates that pretrained AR backbones can be adapted into DLMs through continual pretraining. LLaDA and Dream further show that DLMs can reach billion-parameter regimes and instruction-following settings, making them competitive on reasoning and coding tasks (Nie et al., 2025; Ye et al., 2025). However, a growing body of analysis highlights that the practical benefits of parallel decoding are still limited: current DLMs exhibit strong locality bias and sensitivity to appended mask tokens (Horvitz et al., 2026; Piskorz et al., 2026), and standard supervision can induce autoregressive-like decoding dynamics even in nominally parallel DLMs (Li et al., 2026). These findings motivate methods that explicitly improve the decoding trajectory during training.

Trajectory Optimization via RL and Distillation. A closely related line of work attempts to improve diffusion decoding trajectories directly. Distillation-based methods such as dParallel (Chen et al., 2025) and d3LLM (Qian et al., 2026) train DLMs to follow more aggressive parallel decoding schedules, often by imitating or compressing sampled trajectories so that more tokens can be resolved at earlier denoising steps. Reinforcement-

learning-based approaches have also been proposed for DLMs. CJ-GRPO (Yang et al., 2025b) introduces consistency-aware trajectory reinforcement learning for fewer-step decoding, while SPG (Wang et al., 2026) develops a policy-gradient estimator tailored to DLMs with intractable likelihoods. These methods demonstrate that decoding trajectories are a critical lever for improving the accuracy–parallelism trade-off. However, they typically rely on online diffusion sampling, sampled pseudo-trajectories, or trajectory-level optimization, which introduce substantial computational overhead.