Mask-Aware Policy Gradients for Diffusion Language Models

Haran Raajesh,*[*] Kulin Shah,*[*] Adam Klivans, Philipp Krähenbühl
Department of Computer Science
The University of Texas at Austin
haranr@utexas.edu

Abstract

Reinforcement learning has proven effective for improving reasoning in large language models, but extending it to Masked Diffusion Language Models (MDLMs) remains challenging due to the intractability of the log-likelihood estimation. Existing approaches approximate this log-likelihood by modeling only the token predictions, ignoring the order in which positions are unmasked during generation. We observe that MDLM generation involves two decisions at each step: what tokens to place at each masked position and which positions to remask. We formalize this as a two-stage action MDP, showing that the policy gradient naturally decomposes into a token term and a masking term. Optimizing both terms achieves state-of-the-art results across mathematical reasoning and coding benchmarks, reaching **87.1%** on GSM8K and **53.4%** on MBPP.[1]

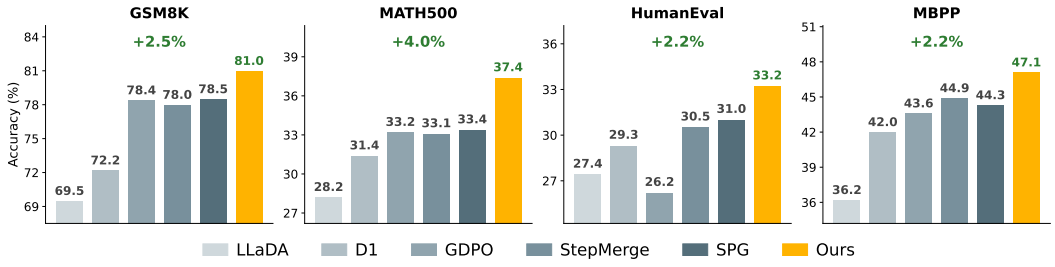

Figure 1: Test accuracy of our method and baseline methods on two mathematical reasoning and two code generation benchmarks. All methods use LLaDA-8B-Instruct as the base model and are evaluated with a generation length of 128. Full results are provided in Table 1.

1 Introduction

Reinforcement learning has emerged as a powerful tool for improving large language models (Guo et al., 2025; Jaech et al., 2024), with applications spanning instruction following (Ouyang et al., 2022a), alignment (Ouyang et al., 2022b), and complex reasoning (Guo et al., 2025; Jaech et al., 2024). At the core of these methods, policy gradient algorithms sample group rollouts autoregressively and estimate their log-likelihoods under the policy. The key to their success is the tractability and simplicity of the autoregressive log-likelihood: the generation trajectory is a sequence of token prediction actions, and the log-likelihood is their product.

Masked Diffusion Language Models (MDLMs) (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024) offer an alternative to autoregressive LLMs, replacing the fixed left-to-right generation order with iterative unmasking from a fully masked sequence (Hoogeboom et al., 2022; Kim et al., 2025). This enables parallel decoding and flexible generation orderings, and MDLMs

[*]Equal contribution.
[1]Code available at https://github.com/Haran71/mask-aware-policy-gradients

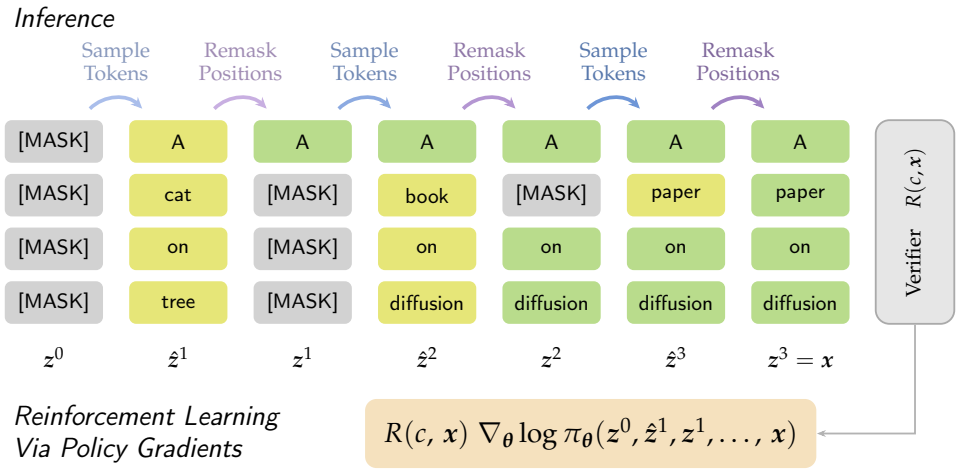

Figure 2: Overview of our approach. Given a prompt c, a masked diffusion language model π_θ generates text by iteratively predicting tokens and remasking positions over T denoising steps, producing an extended trajectory that captures both what tokens were predicted and, through the remasking step, which positions were unmasked. A verifier scores the final output with reward $R(c, x)$, and we compute policy gradients over this full trajectory, jointly optimizing token predictions and position selection.

have proven effective across text generation, code generation, mathematical reasoning, and protein sequence generation (Lou et al., 2024; Sahoo et al., 2024; Nie et al., 2025). However, this flexibility comes at a cost: unlike autoregressive models, MDLMs lack a tractable log-likelihood estimation and are instead trained via an evidence lower bound (ELBO). This makes adapting policy gradient algorithms to MDLMs significantly more challenging.

Existing methods approximate the log-likelihood using ELBO variants (Zhao et al., 2025b; Rojas et al., 2026; Wang et al., 2026) or generation trajectories (Wang et al., 2025a), focusing on the token prediction component. However, the MDLM generation trajectory has a richer structure than its autoregressive counterpart. While an autoregressive trajectory is a sequence of token prediction actions at fixed positions, an MDLM trajectory involves two decisions at each step: what token to place at the masked positions and which of these to remask.[2] The trajectory log-likelihood therefore decomposes into a product of both token prediction and remasking probabilities. Since the policy determines the unmasking order, this richer trajectory structure provides an additional signal for policy optimization that prior ELBO- and trajectory-based methods do not exploit. See Figure 2 for an overview.

We formalize the unmasking order as part of the policy. We model MDLM generation as a two-stage action MDP: at each step, the model selects a token for each masked position and then decides which positions to remask. This factorization yields a policy gradient that decomposes into two terms, one reinforcing token predictions and one reinforcing the ordering. The position term is computed from the model's own logits at no cost beyond the token-likelihood computation, and is compatible with any standard policy gradient algorithm. From a theoretical perspective, we show that ignoring the position component of the policy gradient can miss directions that improve expected return. Empirically, jointly optimizing both terms leads to consistent improvements over prior methods, achieving gains of up to **4.0%** on MATH500, **2.9%** on HumanEval, and **2.5%** on GSM8K and MBPP (Figure 1).

Our contributions are as follows:

[2]We use the term *remasking* following LLaDA (Nie et al., 2025): at each step the model predicts tokens at all masked positions, and those not selected to remain revealed are returned to `[MASK]`. Such positions were never actually unmasked, so this differs from self-correction methods that revert previously committed tokens.

Algorithm 1 Autoregressive Generation

Require: prompt $\boldsymbol{c}$

1: **for** $k = 1$ to n **do**
   ▷ *Sample next token*
2: $x_k \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}_{<k}, \boldsymbol{c})$
3: **end for**
4: **return** $\boldsymbol{x}_{1:n}$

Algorithm 2 MDLM Generation

Require: prompt $\boldsymbol{c}$, sequence length n
1: $\boldsymbol{z}^0 \leftarrow [\texttt{[MASK]}]^n$
2: **for** $t = 1 \dots T$ **do**
   ▷ *Predict masked positions*
3: $\hat{\boldsymbol{z}}^t \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{z}^{t-1}, \boldsymbol{c})$
   ▷ *Remask heuristically*
4: $\boldsymbol{z}^t \leftarrow \text{remask}(\hat{\boldsymbol{z}}^t, \boldsymbol{z}^{t-1})$
5: **end for**
6: **return** $\boldsymbol{z}^T$

- We formalize MDLM generation as a two-stage action MDP and show that the policy gradient decomposes into a token term and a masking term, providing theoretical grounding for optimizing the unmasking order.
- We derive position log-probabilities as a softmax over unmasking scores from the model's own logits, requiring no additional parameters or architectural changes, and no forward passes beyond those already used for the token likelihood.
- We achieve state-of-the-art results across mathematical reasoning and code generation benchmarks.

2 Preliminaries

A language model $\pi_{\boldsymbol{\theta}}$ produces sequences of tokens $\boldsymbol{x}_{1:n} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{c})$ conditioned on an input $\boldsymbol{c}$. The joint distribution over sequences is exponentially large. Autoregressive language models (Guo et al., 2025; Team et al., 2025; Ouyang et al., 2022a; Jaech et al., 2024), decompose the distribution into a product of per-token conditionals:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n} \mid \boldsymbol{c}) = \prod_{k=1}^{n} \pi_{\boldsymbol{\theta}}(x_k \mid \boldsymbol{x}_{<k}, \boldsymbol{c}). \tag{1}$$

This makes likelihood estimation tractable, and simplifies both training and generation, see Algorithm 1.

Masked Diffusion Language Models (MDLMs) (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024) formulate text generation as an iterative denoising process. The forward process corrupts a clean sequence $\boldsymbol{x}_{1:n}$ into $\boldsymbol{z}^t_{1:n}$ by independently replacing each token with a special `[MASK]` token with probability $\alpha_t \in [0, 1]$:

$$\boldsymbol{z}^t_k = \begin{cases} \boldsymbol{x}_k & \text{with probability } 1 - \alpha_t \\ \texttt{[MASK]} & \text{with probability } \alpha_t \end{cases} \tag{2}$$

Here $t = 1 \dots T$ is the diffusion step, and α_t is the corresponding masking probability. We write the forward process as $\boldsymbol{z}^t_{1:n} \sim q_t(\cdot \mid \boldsymbol{x}_{1:n})$. At the final step $t = T$, the sequence $\boldsymbol{z}^T = \boldsymbol{x}$ remains uncorrupted ($\alpha_T = 0$). At the first step $t = 0$, $\boldsymbol{z}^0$ is fully masked ($\alpha_0 = 1$).

A neural network $\pi_{\boldsymbol{\theta}}$ learns to reverse this process by predicting the original tokens at masked positions from a corrupted input $\boldsymbol{z}$, maximizing the Evidence Lower Bound (ELBO) of the data log-likelihood:

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{x}; \boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{t \sim \mathcal{U}\{1,\dots,T\} \\ \boldsymbol{z}^t \sim q_t(\cdot \mid \boldsymbol{x})}} \left[\sum_{k=1}^{n} \frac{1}{\alpha_t} \mathbb{1}(\boldsymbol{z}^t_k = \texttt{[MASK]}) \log \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_k \mid \boldsymbol{z}^t) \right], \tag{3}$$

In essence, the model learns to predict the original token at each masked position, with unmasked positions left unchanged.

Generation in MDLMs proceeds as described in Algorithm 2: We start from a completely masked sequence $z^0 = [\texttt{MASK}]^n$ of length n and iteratively denoise the sequence over T steps with masking schedule $1 = \alpha_0 > \alpha_1 > \ldots > \alpha_T = 0$. Each step t consists of a *prediction* followed by a *remasking*. First, the model predicts tokens at all masked positions: $\hat{z}^t \sim \pi_\theta(\cdot \mid z^{t-1})$, with $\hat{z}^t_k = z^{t-1}_k$ for already-unmasked positions ($z^{t-1}_k \neq \texttt{MASK}$). Then, a subset $\mathcal{U}_t \subseteq \{k : z^{t-1}_k = \texttt{MASK}\}$ of newly predicted positions is selected to remain unmasked, and the rest are remasked:

$$z^t_k = \begin{cases} \hat{z}^t_k & \text{if } k \in \mathcal{U}_t \text{ or } z^{t-1}_k \neq \texttt{MASK}, \\ \texttt{MASK} & \text{otherwise.} \end{cases} \tag{4}$$

This produces a trajectory $\mathbf{z} = (z^0, z^1, \ldots, z^T)$ where $z^T = x_{1:n}$ is the final output. We write $\mathbf{z} \sim \pi_\theta(\cdot \mid c)$ for the distribution over trajectories induced by the model π_θ when generating from prompt c.

In theory, remasking should be random to mimic the training process. In practice however, a deterministic strategy that selects positions to unmask based on confidence scores derived from the model's logits (Chang et al., 2022; Kim et al., 2025) often performs much better. For example, the model may select the top-k positions with the highest maximum logit values to unmask at each step. This allows the model to focus on positions it is most confident about, potentially improving generation quality and convergence speed.

2.1 Reinforcement learning for language models

RL optimizes the expected return

$$J(\theta) = \mathop{\mathbb{E}}_{x \sim \pi_\theta(\cdot \mid c)}[R(c, x)].$$

In the context of language models, x is the output produced by the model and R is the reward obtained via a rule-based verifier or a learned reward model. Most RL algorithms for language models rely on a policy gradient estimate: $\nabla_\theta J = \mathbb{E}_{x \sim \pi_\theta}\big[R(c, x)\, \nabla_\theta \log \pi_\theta(x \mid c)\big]$.

In reinforcement learning, $x \sim \pi_\theta$ is referred to as the action. For autoregressive models, this action is a sequence of token predictions, and $\log \pi_\theta(x \mid c)$ is directly available from Equation (1): the chain rule decomposition gives a sum of per-token log-probabilities, computed in a single forward pass. This makes the policy gradient fully tractable.

For MDLMs, however, $\log \pi_\theta(x \mid c)$ is *intractable* to compute, as it is defined by the marginal over all trajectories that produce x:

$$\pi_\theta(x \mid c) = \sum_{\mathbf{z}: z^T = x} \pi_\theta(\mathbf{z} \mid c). \tag{5}$$

Existing approximations fall into two families:

Evidence bound methods (Zhao et al., 2025b; Rojas et al., 2026; Wang et al., 2026) estimate the log-likelihood by evaluating the model on randomly masked versions of the completed sequence, typically computing some variant of the ELBO objective (Eq. 3) used for training.

Trajectory-based methods (Wang et al., 2025a) rewrite the expected return as

$$J(\theta) = \mathop{\mathbb{E}}_{x \sim \pi_\theta(\cdot \mid c)}[R(c, x)] = \sum_x \pi_\theta(x \mid c) R(c, x) = \sum_{\mathbf{z}} \pi_\theta(\mathbf{z} \mid c) R(c, z^T) = \mathop{\mathbb{E}}_{\mathbf{z} \sim \pi_\theta(\cdot \mid c)}[R(c, z^T)]. \tag{6}$$

The policy gradient estimate of the rephrased return is given by

$$\nabla_\theta J(\theta) = \sum_{\mathbf{z}} R(c, z^T) \frac{\nabla_\theta \pi_\theta(\mathbf{z} \mid c)}{\pi_\theta(\mathbf{z} \mid c)} \pi_\theta(\mathbf{z} \mid c) = \mathop{\mathbb{E}}_{\mathbf{z} \sim \pi_\theta(\cdot \mid c)}[R(c, z^T) \nabla_\theta \log \pi_\theta(\mathbf{z} \mid c)] \tag{7}$$

Unlike ELBO methods, trajectory-based methods compute an exact policy gradient estimate given we can compute $\log \pi_\theta(\mathbf{z} \mid c)$ for any sampled trajectory $\mathbf{z}$:

$$\pi_\theta(\mathbf{z} \mid c) = \pi_\theta(z^1, \ldots, z^T \mid c) = \prod_{t=1}^{T} \pi_\theta(z^t \mid c, z^{\leq t-1}) = \prod_{t=1}^{T} \pi_\theta(z^t \mid c, z^{t-1}), \tag{8}$$

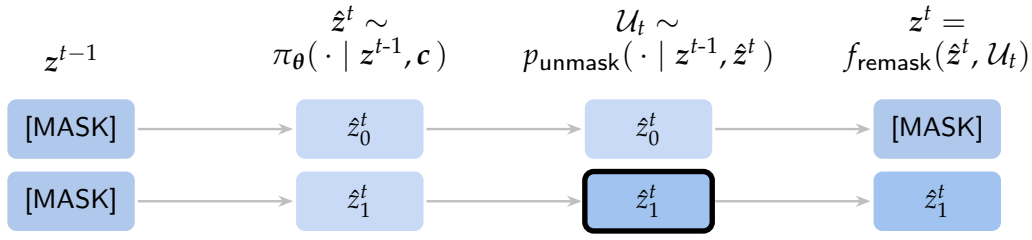

Figure 3: Illustration of a single denoising step with probabilistic remasking. (1) The model receives a partially masked sequence z^{t-1} and predicts tokens at all masked positions, producing $\hat{z}^t$. (2) Rather than deterministically unmasking the top-K most confident positions, we sample a subset $\mathcal{U}_t$ from a Plackett–Luce distribution proportional to the predicted token log-likelihoods (shown as the bar chart). (3) The remasking operator f_{remask} retains the selected positions and remasks all others, yielding z^t. In this example, position 1 is selected ($\mathcal{U}_t = \{1\}$), so only $\hat{z}_1^t$ is kept while position 0 is remasked.

where the last step follows since the diffusion process is Markovian. Wang et al. (2025a) approximate $\log \pi_\theta(\mathbf{z} \mid \boldsymbol{c}) \approx \sum_t \log \pi(z^t | z^{t-1}, c)$, but ignore the remasking step.

We show that a proper treatment of this full probability estimate leads to a position term in the policy gradient estimate, which in practice greatly improves performance.

3 Method

We derive a policy gradient estimator for MDLMs by modeling the probability of trajectories using both token prediction and remasking in the trajectory likelihood. Denoising from sequence z^{t-1} to z^t can be viewed as a two-stage process: predicting mask tokens $\hat{z}^t \sim \pi_\theta(\cdot \mid z^{t-1})$, and selecting positions to unmask $\mathcal{U}_t$. A masking function $z^t = f_{\text{remask}}(\hat{z}^t, \mathcal{U}_t \cup z^{t-1} \neq \texttt{[MASK]})$ remasks all remaining tokens. Figure 3 shows an overview.

The main issue with the above formulation is that the standard greedy top-K mask $\mathcal{U}_t$ depends on π_θ, but is not differentiable, and thus a policy gradient estimate is not possible. We instead replace the greedy remasking with a probabilistic version both during generation, and reinforcement learning. This leads to a proper gradient estimate, and better empirical performance.

3.1 Probabilistic remasking

Let $v_k^t = \log \pi(\hat{z}_k^t | z^{t-1})$ be the log-likelihood of the sampled token $\hat{z}_k^t$ at position k and step t. In standard top-K remasking, the K positions with the highest scores v_k^t are deterministically selected for unmasking. This hard selection is not differentiable with respect to the model parameters $\boldsymbol{\theta}$, preventing gradient-based optimization of the remasking. We replace this with a *probabilistic* variant: instead of selecting the top-K positions deterministically, we sample K positions *without replacement* from a categorical distribution whose probabilities are proportional to $\exp(v_k^t/\tau)$ over all masked positions, where $\tau > 0$ is a temperature parameter. As $\tau \to 0$, this recovers the greedy top-K selection; for $\tau > 0$, it defines a proper distribution over position subsets whose log-probability can be included in the trajectory likelihood and optimized via policy gradients. Specifically, we sample the unmasking set $\mathcal{U}_t = \{u_1, u_2, \ldots, u_K\}$ sequentially without replacement, following the Plackett–Luce model (Plackett, 1975; Luce, 1959): $p_{\text{unmask}}(\mathcal{U}_t \mid z^{t-1}, \hat{z}^t) = \prod_{i=1}^K p_{\text{unmask}}(u_i = x | u_{<i}, z^{t-1}, \hat{z}^t)$ with

$$p_{\text{unmask}}(u_i = x | u_{<i}, z^{t-1}, \hat{z}^t) = \frac{1_{[z_x^{t-1} = \texttt{[MASK]} \wedge x \notin u_{<i}]} \exp(v_x^t/\tau)}{\sum_{j=1}^n 1_{[z_j^{t-1} = \texttt{[MASK]} \wedge j \notin u_{<i}]} \exp(v_j^t/\tau)}, \tag{9}$$

where each distribution models a softmax over currently masked tokens. Masked tokens with a higher log-likelihood v_k^t have a higher chance of being unmasked. Tokens with a low log-likelihood will likely be remasked.

The remasking probability p_{remask} is then

$$p_{\text{remask}}(z^t|\mathcal{U}_t, z^{t-1}, \hat{z}^t) = 1_{[z^t = f_{\text{remask}}(\hat{z}^t, \mathcal{U}_t \cup z^{t-1}=\texttt{[MASK]})]}$$

where f_{remask} is the remasking operator that applies the mask $\mathcal{U}_t$ to the sequence z^t:

$$f_{\text{remask}}(\hat{z}^t, S)_k = \begin{cases} \hat{z}_k^t & \text{if } k \in S \\ \texttt{[MASK]} & \text{otherwise} \end{cases}$$

Unlike greedy top-K selection, this probabilistic formulation defines a proper distribution $p(\mathcal{U}_t)$ over position subsets whose log-probability is differentiable with respect to the model parameters. The remasking operator f_{remask} and p_{remask} are deterministic and do not directly depend on the model parameters.

3.2 Reinforcement learning with probabilistic remasking

The diffusion process can be viewed as producing an extended sequence $\hat{z} = \{z^0, \hat{z}^1, \mathcal{U}^1, z^1, \ldots, z^T\}$ over a distribution:

$$\pi_{\boldsymbol{\theta}}(\hat{z}|\boldsymbol{c}) = \prod_{t=1}^{T} \pi(\hat{z}^t|z^{t-1}, \boldsymbol{c}) p_{\text{unmask}}(\mathcal{U}_t|\hat{z}^t, z^{t-1}, \boldsymbol{c}) p_{\text{remask}}(z^t|\mathcal{U}_t, \hat{z}^t, z^{t-1}, \boldsymbol{c}). \tag{10}$$

This interpretation extends the trajectory-based view of diffusion in Wang et al. (2025a), and further considers the remasking process as part of the generation. It leads to a similarly simple and direct policy gradient estimate over extended trajectories $\hat{z}$:

$$\nabla_\theta J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\hat{z}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{c})}\Big[R(\boldsymbol{c}, z^T)\sum_{t=1}^{T}\Big(\underbrace{\nabla_\theta \log p_{\text{unmask}}(\mathcal{U}_t|\hat{z}^t, z^{t-1}, \boldsymbol{c})}_{\text{unmasking gradient}} + \underbrace{\nabla_\theta \log \pi_\theta(\hat{z}^t \,|\, \boldsymbol{c}, z^{t-1})}_{\text{token gradient}}\Big)\Big]. \tag{11}$$

Here, the remasking is deterministic and does not depend on $\boldsymbol{\theta}$ and thus falls out of the gradient estimate. Both terms in the decomposition (Eq. 11) are functions of the same policy $\pi_{\boldsymbol{\theta}}$. Therefore, we can optimize them jointly with any standard policy gradient algorithm. We use GSPO (Zheng et al., 2025) and find it most effective.

Finally, we note that the unmasking gradient is necessary for a correct gradient estimate: optimizing only the token component, as in Wang et al. (2025a), can miss directions that improve the objective $J(\boldsymbol{\theta})$. Intuitively, changes to the unmasking order can affect the final reward even when they leave the token probabilities unchanged, so a token-only gradient provides no signal along such directions. We make this precise in Appendix G with a minimal counterexample: a finite-horizon MDLM problem with a linear policy in which the token-only gradient is identically zero along a direction $\boldsymbol{v}$ that strictly increases the true objective, while the full position-aware gradient is not.

4 Experiments

Following prior work (Zhao et al., 2025b; Rojas et al., 2026; Wang et al., 2026; Tang et al., 2026), we use LLaDA-8B-Instruct (Nie et al., 2025) as the base model for RL fine-tuning. We employ Low-Rank Adaptation (LoRA) with rank $r = 128$, scaling factor $\alpha = 64$, and dropout 0.05, targeting all attention and MLP projection layers. We use 4-bit quantization (NF4) and Flash Attention 2 for memory efficiency, with all training in bfloat16.

Position log-prob computation. We estimate both token and position log-probabilities using the StepMerge approximation (Wang et al., 2025a), which groups T denoising steps into N contiguous segments and treats all tokens unmasked within a segment as if they were unmasked at the same step. See Appendix F for more details on the method and

	GSM8K			MATH500			HumanEval			MBPP		
Model / Seq Len	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**
LLaDA-8B-Inst.	69.5	77.2	79.8	28.2	32.4	34.6	27.4	35.5	37.8	36.2	41.2	40.4
LLaDA-1.5	70.4	80.5	81.9	26.8	32.2	35.8	—	—	—	—	—	—
d1	72.2	80.6	81.3	31.4	36.0	39.4	29.3	39.0	34.8	42.0	45.5	41.6
wd1	74.6	81.5	83.0	31.0	37.4	39.0	—	—	—	—	—	—
UniGRPO	74.9	82.5	82.7	32.4	37.4	39.4	—	—	—	—	—	—
GDPO	78.4	82.8	<u>85.0</u>	33.2	39.6	41.4	26.2	39.6	39.0	43.6	<u>50.6</u>	47.1
SPG w/ Mixture	<u>78.5</u>	<u>83.9</u>†	84.5	<u>33.4</u>	<u>40.0</u>	<u>41.8</u>	<u>31.0</u>	<u>40.6</u>	<u>41.1</u>	44.3	<u>50.6</u>	50.8
StepMerge	78.0	83.3	84.8	33.1	39.1	41.2	30.5	39.9	40.7	<u>44.9</u>	50.1	<u>50.9</u>
Ours	**81.0**	**85.9**	**87.1**	**37.4**	**42.2**	**44.2**	**33.2**	**43.1**	**44.0**	**47.1**	**52.8**	**53.4**
	+2.5	+2.0	+2.1	+4.0	+2.2	+2.4	+2.2	+2.5	+2.9	+2.2	+2.2	+2.5

Table 1: Comparison of RL methods for diffusion language models on mathematical reasoning and code generation benchmarks. All methods use LLaDA-8B-Instruct as the base model. Results reported at generation lengths 128, 256, and 512. Best results are in **bold**, second best are <u>underlined</u>. Dashes indicate results not reported in the original work. †Obtained after running publicly available code; the originally reported number was 86.1.

the theoretical analysis of approximation error induced by using StepMerge to evaluate the trajectory log-probability. We subsample K boundaries for evaluation. The unmasking score at each masked position is the maximum logit from the model's output. Since we compute both log-probability terms from the same forward passes, the position term adds no overhead. For our main results, we set $N = 32$ and $K = 12$ (see Appendix E). We optimize with GSPO (Zheng et al., 2025), clipping the token and position importance ratios separately.

Datasets and evaluation. We evaluate on GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2023) for mathematical reasoning, and HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code generation. For math, we follow the train-test splitting, reward functions, and evaluation protocol of d1 (Zhao et al., 2025b), wd1 (Tang et al., 2026), and SPG (Wang et al., 2026). For code, we follow GDPO (Rojas et al., 2026), using KodCode-Light-RL-10K (Xu et al., 2025) as the training dataset. All evaluations are zero-shot. For RL rollouts, we use a sequence length of 256 tokens, 128 diffusion steps, and block size 32. We report accuracy at generation lengths 128, 256, and 512.

Baselines. We compare against LLaDA-8B-Instruct (Nie et al., 2025) (base model), LLaDA-1.5 (Zhu et al., 2025), d1 (Zhao et al., 2025b), wd1 (Tang et al., 2026), UniGRPO (Yang et al., 2025b), GDPO (Rojas et al., 2026), SPG w/ Mixture (Wang et al., 2026), and StepMerge (Wang et al., 2025a). Math results for prior methods are taken from their original papers. SPG does not report code generation results, so we evaluate their official code on HumanEval and MBPP ourselves. For StepMerge, we reimplement the algorithm with the same K subsampling as our method, making it a direct ablation that differs only in the absence of the masking term.

4.1 Experimental Results

Main results. Table 1 presents our main results across four benchmarks. Our method achieves the best results in every configuration. The comparison against StepMerge is particularly informative: both methods share the same trajectory-based likelihood framework and K subsampling, differing only in the masking log-probability term. The consistent 2-4 point improvement across all configurations demonstrates that the masking gradient provides a meaningful training signal beyond what token-level optimization alone captures. Against evidence bound methods such as GDPO and SPG, our method also shows consistent gains, particularly on MATH500 (+4.0 at length 128, +2.4 at length 512) and code generation benchmarks. We further show that our method generalizes to a second base model, Dream-7B (Appendix C), and to two planning tasks, Sudoku and Countdown (Appendix B).

	SPG	Ours
Throughput (steps/h)	**360**	290
Convergence (steps)	6500	**6000**
Final accuracy (%)	78.5	**81.0**
Match SPG-final (h)	18	**15**

(a) Efficiency summary.

Target	SPG (h)	Ours (h)	Speedup
74%	8.6	4.5	1.9×
75%	9.9	5.8	1.7×
76%	13.4	10.0	1.3×
77%	15.9	11.8	1.3×
78%	17.2	14.0	1.2×
79%	—	16.5	—
80%	—	18.3	—

(b) Wall-clock hours to reach a target accuracy.

Table 2: Training efficiency on GSM8K (8 H100 GPUs, generation length 128). (a) Throughput, convergence, and final accuracy, with the better value per row in **bold**. (b) Wall-clock hours to reach fixed accuracy targets, with speedup over SPG; "—" denotes a target SPG never reaches.

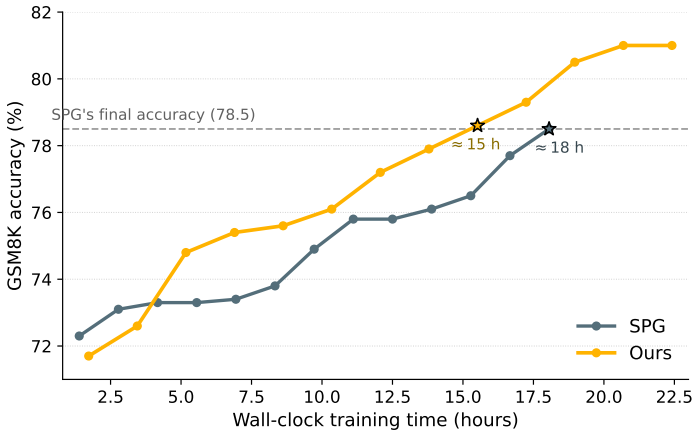

Figure 4: Wall-clock training efficiency on GSM8K (8 H100 GPUs, generation length 128). Test accuracy versus wall-clock training time for our method and SPG (Wang et al., 2026). Although our method runs fewer steps per hour, it overtakes SPG within the first few hours and converges to a higher accuracy (81.0 vs. 78.5). Stars mark where each method first reaches SPG's final accuracy: ≈15 hours for ours versus ≈18 hours for SPG.

Training efficiency. We compare the training efficiency of our method against SPG (Wang et al., 2026), the strongest ELBO-based baseline in our setting, on GSM8K. Both methods run on the same hardware (8 H100 GPUs) from the same base model, with identical generation, optimization, and evaluation configurations; the only difference is the log-probability estimator.

We compare the wall-clock time to reach fixed accuracy targets both for our method and SPG in Table 2b. Our method reaches 74% accuracy 1.9× faster than SPG and reaches SPG's final accuracy in roughly 15 hours, compared with 18 hours for SPG. It also reaches 79% and 80% accuracy, which SPG does not reach. Table 2a gives the efficiency summary: our per-step throughput is lower because the trajectory-based estimator requires more forward passes per optimizer step, but the position-aware gradient converges in fewer steps and reaches a higher final accuracy (81.0 vs. 78.5). Figure 4 plots accuracy against wall-clock time: our method starts marginally behind SPG, overtakes it within the first few hours, and the gap widens as training proceeds. Across three random seeds our final accuracy varies by at most 0.2 points, so these differences are well outside run-to-run variation.

Inference block size. Figure 5 compares inference block sizes (32, 64, and full sequence) at generation length 256 with confidence-based unmasking. We compare against SPG (Wang et al., 2026), a strong evidence-bound baseline, and StepMerge (Wang et al., 2025a), which

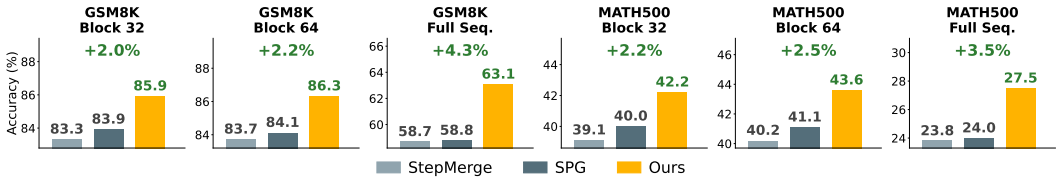

Figure 5: Ablation over inference block size. All methods use confidence-based unmasking at generation length 256. Results for baselines are from SPG (Wang et al., 2026).

shares the same trajectory-based framework but without the position term. All methods are trained with block size 32. Our method outperforms both baselines at every block size, and the gap widens as block size increases. Against SPG, the margin grows from +2.0 to +4.3 on GSM8K and from +2.2 to +3.5 on MATH500 (block 32 to full sequence). Against StepMerge, the margin grows from +2.6 to +4.4 on GSM8K and from +3.1 to +3.7 on MATH500. We attribute this to the growing importance of position selection as the diffusion window expands: larger blocks present more masked positions to choose from at each step, increasing the variability in unmasking order and amplifying the benefit of mask-aware optimization. Full numerical results are provided in Table 6 in the appendix. We provide further ablations on the RL algorithm, the StepMerge configuration, the sampling temperatures, and the unmasking-set size $|\mathcal{U}_t|$ in Appendix E.

5 Related work

Diffusion Language Models. Discrete diffusion models for text generation have developed along several lines. Early work applied diffusion in continuous embedding space (Li et al., 2022; Gong et al., 2022), while more recent approaches operate directly in discrete token space. Score Entropy Discrete Diffusion (SEDD) (Lou et al., 2023) and Masked Diffusion Language Models (MDLM) (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024) established masked diffusion as a competitive paradigm for text generation, with simplified training objectives based on the ELBO. Scaling these models to larger sizes, LLaDA (Nie et al., 2025) and Dream (Gong et al., 2024) demonstrated that masked diffusion LLMs can approach autoregressive performance on standard benchmarks. More recently, diffusion language models have been extended to multimodal settings (Yang et al., 2025b; Li et al., 2025; Wang et al., 2025b; Swerdlow et al., 2025), and several works have improved inference efficiency through caching (Ma et al., 2025; Liu et al., 2025a) and parallel decoding strategies (Wu et al., 2025; Arriola et al., 2025).

Reinforcement Learning for Language Models. Reinforcement learning has become a central technique for post-training large language models, building on classical policy gradient methods (Williams, 1992; Sutton et al., 1999). Early work used RL from human feedback (RLHF) with PPO (Schulman et al., 2017) and TRPO (Schulman et al., 2015) to align models with human preferences (Christiano et al., 2017; Ouyang et al., 2022b). More recently, RL has been applied to improve reasoning capabilities. DeepSeekMath (Shao et al., 2024) introduced Group Relative Policy Optimization (GRPO), a critic-free policy gradient method that estimates baselines from group rollouts, removing the need for a value network. DeepSeek-R1 (Guo et al., 2025) and Kimi K1.5 (Team et al., 2025) scaled RL-based reasoning training, demonstrating that policy gradient methods with verifiable rewards can elicit strong mathematical and coding performance. Subsequent work has refined the GRPO objective, reducing bias (Liu et al., 2025b), improving sample efficiency (Cohen et al., 2025), and introducing sequence-level clipping (Zheng et al., 2025).

Reinforcement Learning for Diffusion Language Models. Reinforcement learning for MDLMs has been used for reward optimization (Wang et al., 2024), preference alignment (Zhu et al., 2025), faster parallel decoding (Yang et al., 2025a; Jazbec et al., 2025; Chen et al., 2025; Zhao et al., 2025a), and reasoning (Zhao et al., 2025b; Rojas et al., 2026; Wang et al., 2026; 2025a). For reasoning, the central challenge of RL is estimating the

intractable sequence log-likelihood. ELBO-based methods estimate log-probabilities via variants of the training objective: d1 (Zhao et al., 2025b) uses a mean-field approximation, UniGRPO (Yang et al., 2025b) and ESPO (Ou et al., 2025) use sequence-level Monte Carlo estimators, GDPO (Rojas et al., 2026) tightens the bound with Gaussian quadrature, wd1 (Tang et al., 2025) uses weighted optimization, and SPG (Wang et al., 2026) combines ELBO and EUBO bounds with block-wise masking. Trajectory-based methods instead decompose the log-likelihood over denoising steps, attempting to follow the generation trajectory: d2 (Wang et al., 2025a) and Wang et al. (2025c) partition the trajectory into segments via the StepMerge estimator. However, these trajectory-based methods model the trajectory as a sequence of token prediction steps only, and in doing so ignore the position selection decision that MDLMs make at generation time. Our work addresses this gap.

Learning the unmasking order. A line of work aims to learn or optimize the unmasking order of MDLMs, frequently to accelerate parallel decoding by reducing the number of forward passes. Jazbec et al. (Jazbec et al., 2025) learn an unmasking policy over the model's confidences, dUltra (Chen et al., 2025) trains an unmasking planner head with reinforcement learning, and P2 (Peng et al., 2025) frames sampling as path planning to choose which positions to update; Where-to-Unmask (Asano et al., 2026) instead trains a separate planner in a supervised fashion to imitate an oracle order derived from ground-truth tokens. Unlike these methods, our goal is not to introduce an explicit model of the unmasking order; rather, we show that a position-selection term arises directly from the complete policy-gradient estimator, and we optimize it using the model's own logits without any additional module. The most closely related method in this respect is LLaDOU/DCoLT (Huang et al., 2025), which also incorporates position selection into the RL policy through a Plackett–Luce distribution; unlike DCoLT, which trains a separate module to model the position logits, we reuse the base model's own logits. We provide a detailed comparison with DCoLT in Appendix D.

6 Conclusion

We propose a policy gradient method for masked diffusion language models that optimizes both token prediction and position selection. By replacing greedy top-K remasking with a probabilistic variant, we obtain a differentiable distribution over unmasking positions, yielding a gradient decomposition into a token term and a masking term. Extensive experiments across mathematical reasoning and code generation benchmarks show that jointly optimizing these terms in the policy gradient achieves significant improvements over existing methods and state-of-the-art results.

Acknowledgments

The experiments in this work were run on the Vista GPU Cluster through the Center for Generative AI (CGAI) and the Texas Advanced Computing Center (TACC) at The University of Texas at Austin.

References

Arel. Arel's sudoku generator. https://www.ocf.berkeley.edu/ arel/sudoku/main.html, 2025.

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

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

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

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

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

Shirui Chen, Jiantao Jiao, Lillian J Ratliff, and Banghua Zhu. dultra: Ultra-fast diffusion language models via reinforcement learning. *arXiv preprint arXiv:2512.21446*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

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

Taco Cohen, David W Zhang, Kunhao Zheng, Yunhao Tang, Remi Munos, and Gabriel Synnaeve. Soft policy optimization: Online off-policy rl for sequence models. *arXiv preprint arXiv:2503.05453*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*, 2024.

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

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Lm8T39vLDTE.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models. *arXiv preprint arXiv:2505.10446*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Metod Jazbec, Theo X Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. *arXiv preprint arXiv:2512.09106*, 2025.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DjJmre5IkP.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *DeepRLStructPred@ICLR*, 2019. URL https://api.semanticscholar.org/CorpusID:198489118.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*, 2025.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *ICML*, 2024.

R Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley & Sons, 1959.

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

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

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022a.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022b.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

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

Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 1975.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JaqvespRBP.

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

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Alexander Swerdlow, Mihir Prabhudesai, Siddharth Gandhi, Deepak Pathak, and Katerina Fragkiadaki. Unified multimodal discrete diffusion. *arXiv preprint arXiv:2503.20853*, 2025.

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

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=L2rfd2Czbj.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=18j5Q49GwN.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improved techniques for training reasoning diffusion language models. *arXiv preprint arXiv:2509.21474*, 2025a.

Jin Wang, Yao Lai, Aoxue Li, Shifeng Zhang, Jiacheng Sun, Ning Kang, Chengyue Wu, Zhenguo Li, and Ping Luo. Fudoki: Discrete flow-based unified understanding and generation via kinetic-optimal velocities. *arXiv preprint arXiv:2505.20147*, 2025b.

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

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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

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

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*, 2025a.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025b.

Hanyang Zhao, Dawen Liang, Wenpin Tang, David Yao, and Nathan Kallus. Diffpo: Training diffusion llms to reason fast and furious via reinforcement learning. *arXiv preprint arXiv:2510.02212*, 2025a.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

A Training details

Base model. We use LLaDA-8B-Instruct (Nie et al., 2025) as the base model for all experiments. LLaDA is a masked diffusion language model that uses absorbing-state diffusion with a special `[MASK]` token (ID 126336). We do not apply supervised fine-tuning before RL training; all results use the pretrained instruction-tuned checkpoint directly.

Parameter-efficient fine-tuning. We use Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning. We set the LoRA rank to $r = 128$, scaling factor $\alpha = 64$, and dropout 0.05. LoRA adapters are applied to all attention projection layers (query, key, value, and output) and all MLP projection layers (gate, up, and down). The base model weights are frozen in 4-bit quantization (NF4 format) to reduce memory usage, with all LoRA parameters and computations in bfloat16. We use Flash Attention 2 (Dao, 2023) for memory-efficient attention computation.

Optimizer. We use AdamW (Loshchilov & Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay 0.1. The learning rate is set to 3×10^{-6} with a constant schedule and a brief warmup (warmup ratio 0.0001). We clip gradients to a maximum norm of 0.2.

Batch size and gradient accumulation. We use a per-device training batch size of 3 with gradient accumulation over 2 steps. Training is distributed across 8 GPUs, yielding an effective batch size of $3 \times 2 \times 8 = 48$ prompt-completion pairs per optimization step. For each prompt, we sample $G = 6$ completions for group-relative advantage estimation.

Policy optimization. We optimize using GSPO (Zheng et al., 2025) with separate clipping for the token and position importance ratios. GSPO clips the importance ratio with a lower bound of 3×10^{-4} and an upper bound of 4×10^{-4}. No KL regularization term is used. We perform $\mu = 12$ inner gradient updates per batch of generated completions. We synchronize the reference model every 64 optimization steps.

Generation parameters. During RL rollouts, we generate completions with a maximum length of 256 tokens. Generation uses semi-autoregressive block-wise decoding with a block size of 32 tokens and 128 diffusion steps per block. At each denoising step, we use probabilistic remasking. The sampling temperature is set to 0.9 for token selection and 0.5 for remasking.

Datasets. For mathematical reasoning, we train on the training split of GSM8K (Cobbe et al., 2021) for grade-school math and on the training split of the MATH dataset for competition-level math, evaluating on the MATH500 (Lightman et al., 2023) test set. We follow the same train-test splitting as d1 (Zhao et al., 2025b) and SPG (Wang et al., 2026). For code generation, we train on KodCode-Light-RL-10K (Xu et al., 2025), a dataset of coding problems at varying difficulty levels with synthetic unit tests for reward computation. We evaluate on HumanEval (Chen et al., 2021) (164 hand-crafted Python problems) and sanitized MBPP (Austin et al., 2021) (257 crowd-sourced Python tasks). We follow the same dataset setup as GDPO (Rojas et al., 2026).

Reward functions. We follow the reward functions of d1 (Zhao et al., 2025b) and SPG (Wang et al., 2026) for mathematical reasoning, and GDPO (Rojas et al., 2026) for code generation.

GSM8K. We use a composite reward with five additive components:

- XML Structure Reward: $+0.125$ per correctly placed opening or closing tag.
- Soft Format Reward: $+0.5$ if the output matches the pattern `<reasoning>...</reasoning><answer>...</answer>`.
- Strict Format Reward: $+0.5$ for exact formatting with correct line breaks.
- Integer Answer Reward: $+0.5$ if the extracted answer is a valid integer.
- Correctness Reward: $+2.0$ if the extracted answer matches the ground truth.

MATH500. We use a two-component reward:

- Format Reward: $+1.0$ if answer tags are present with `\boxed{}` inside; $+0.75$ if answer tags are present without `\boxed{}`; $+0.5$ if `\boxed{}` is present without answer tags; $+0.25$ otherwise.
- Correctness Reward: $+2.0$ if the answer in `\boxed{}` matches the ground truth.

Code generation (HumanEval, MBPP). We use a two-component reward:

- Format Reward: $+1.0$ if the output contains a valid Python code block with no syntax errors; $+0.5$ if well-formatted but with syntax errors; $+0.0$ otherwise.

- Code Execution Reward: the fraction of unit tests passed. We block unsafe modules (os, sys, shutil, subprocess, socket, psutil, ctypes, pathlib, builtins, __import__) and assign a reward of 0 if any are used.

Training schedule. We train each dataset independently. For GSM8K, we train for approximately 8,000 optimization steps. For MATH500, we train for approximately 8,000 steps. For code generation (KodCode), we train for approximately 6,000 steps. We save checkpoints every 100 steps and evaluate at each checkpoint.

All experiments are conducted on 8 NVIDIA H100-80GB GPUs with DeepSpeed ZeRO-2.

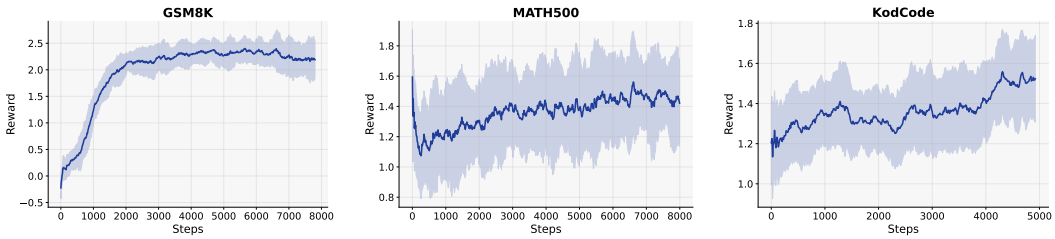

Figure 6: Training reward dynamics for our method on GSM8K, MATH500, and KodCode. Curves show EMA-smoothed reward with one standard deviation band.

B Results on planning tasks

We additionally evaluate on two planning benchmarks, Sudoku (Arel, 2025) and Countdown (Pan et al., 2025), following the evaluation protocol of SPG (Wang et al., 2026) and d1 (Zhao et al., 2025b). Following SPG, Sudoku uses 3 few-shot examples during rollouts and evaluation. Table 3 reports accuracy at generation lengths 128, 256, and 512; results for prior methods are taken from SPG (Wang et al., 2026). Our method achieves the best results on both tasks at every generation length, outperforming the strongest baseline, SPG, with the largest gain on Sudoku (+3.1 at length 512). This is consistent with our finding that the benefit of position-aware optimization grows as the number of masked positions increases.

	Sudoku			Countdown		
Model / Seq Len	**128**	**256**	**512**	**128**	**256**	**512**
LLaDA-8B-Inst.	5.7	27.7	26.2	18.8	16.8	16.8
LLaDA-1.5	7.4	26.9	29.0	21.9	21.1	21.5
d1	7.2	32.5	29.3	30.9	30.9	34.4
wd1	33.1	32.1	22.5	48.8	52.3	50.8
UniGRPO	59.0	67.0	62.9	44.5	43.0	57.0
SPG w/ Mixture	<u>82.9</u>	<u>94.0</u>	<u>93.1</u>	<u>68.8</u>	<u>70.7</u>	<u>70.3</u>
Ours	**84.5**	**94.9**	**96.2**	**69.3**	**71.4**	**71.0**

Table 3: Results on two planning benchmarks, Sudoku and Countdown, using LLaDA-8B-Instruct as the base model. Results for prior methods are taken from SPG (Wang et al., 2026); following SPG, Sudoku uses 3 few-shot examples during rollouts. Best results are in **bold**, second best are <u>underlined</u>.

C Generalization to Dream-7B

We additionally evaluate our method on a different base model, Dream-7B (Gong et al., 2024). Table 4 reports GSM8K, MATH500, and Sudoku accuracy at generation lengths 128, 256, and 512; SPG results are obtained by running their publicly available code. Our

method outperforms all baselines across all three benchmarks and generation lengths, with improvements of up to 2.9 points over the strongest baseline.

	GSM8K			MATH500			Sudoku		
Model / Seq Len	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**
Dream-7B	75.8	81.3	80.7	38.2	45.7	48.0	9.3	2.1	14.0
d1	77.0	81.9	81.7	39.4	46.9	48.9	64.4	69.7	51.1
wd1	76.3	82.4	82.9	39.5	47.4	50.5	29.5	39.2	30.3
ESPO	79.6	82.3	82.0	40.3	47.4	50.3	71.7	72.3	71.3
SPG w/ Mixture	<u>80.1</u>	<u>82.6</u>	<u>83.5</u>	<u>41.1</u>	<u>48.5</u>	<u>50.7</u>	<u>72.1</u>	<u>72.4</u>	<u>71.9</u>
Ours	**82.5**	**84.1**	**86.4**	**43.8**	**50.3**	**52.9**	**73.4**	**74.1**	**74.3**

Table 4: Generalization to Dream-7B (Gong et al., 2024) on GSM8K, MATH500, and Sudoku, at generation lengths 128, 256, and 512. SPG results are obtained by running their publicly available code. Best results in **bold**, second best <u>underlined</u>.

D Comparison to LLaDOU

LLaDOU (Huang et al., 2025), trained with the Diffusion Chain of Lateral Thought (DCoLT) algorithm, shares our high-level insight that the unmasking order should be part of the policy, and arrives at a similar probabilistic framework based on Plackett–Luce position sampling. The key difference is architectural: LLaDOU introduces a separately trained head for position selection, whereas our method derives position scores directly from the model's own token log-likelihoods, requiring no additional parameters or architectural changes. This has the practical advantage that our approach integrates directly with existing MDLM training infrastructure such as StepMerge (Wang et al., 2025a), and applies to any off-the-shelf MDLM.

We compare the two approaches on GSM8K in Table 5, reporting accuracy, training throughput, and total training cost. When DCoLT uses full trajectory back-propagation (full replay), it reaches accuracy comparable to ours but at roughly 6× lower throughput and 5× higher total GPU cost. In a matched comparison where both methods use the StepMerge approximation, our method outperforms DCoLT by approximately 5% on average across generation lengths. This efficiency gain does not come solely from StepMerge: even with StepMerge, DCoLT still requires about 5× more GPU hours to train (∼800 vs ∼160), because it must train a separate position-selection head from scratch. Our method instead derives position scores from the model's existing token log-likelihoods, providing a useful signal from the start of training and converging much faster. Overall, our method matches or exceeds DCoLT's accuracy while being substantially cheaper to train and applicable to any MDLM without architectural changes.

Method	**128**	**256**	**512**	**Throughput (steps/h)**	**GPU hours**
DCoLT (full replay)	80.4	84.3	86.2	48	∼800
DCoLT (w/ StepMerge)	77.1	80.1	82.3	280	∼800
Ours	**81.0**	**85.9**	**87.1**	**290**	∼**160**

Table 5: Comparison with LLaDOU (trained with DCoLT) on GSM8K. We report accuracy at generation lengths 128, 256, and 512, training throughput, and total training cost (number of GPUs × hours). Our method matches or exceeds LLaDOU's accuracy at a fraction of the training cost. Best accuracy in **bold**.

E Additional ablations

E.1 Inference block size

Table 6 reports the full numerical results for the inference block size ablation discussed in Section 4.1 and summarized in Figure 5.

	Block 32		Block 64		Full Sequence	
Model	**GSM8K**	**MATH500**	**GSM8K**	**MATH500**	**GSM8K**	**MATH500**
LLaDA-8B-Inst.	77.2	32.4	78.6	33.2	23.9	17.8
LLaDA-1.5	80.5	32.2	81.0	35.4	41.4	20.4
d1	80.6	36.0	80.9	37.6	57.5	22.6
wd1	81.5	37.4	82.5	37.4	56.7	25.0
UniGRPO	82.5	37.4	82.3	37.4	50.0	24.2
SPG w/ Mixture	83.9	40.0	84.1	41.1	58.8	24.0
StepMerge	83.3	39.1	83.7	40.2	58.7	23.8
Ours	**85.9**	**42.2**	**86.3**	**43.6**	**63.1**	**27.5**

Table 6: Ablation over inference block size. All methods use confidence-based unmasking at generation length 256. Results for baselines are from SPG (Wang et al., 2026).

E.2 RL algorithm

Our decomposition is compatible with any policy gradient algorithm. We compare three widely used algorithms, RLOO (Kool et al., 2019), GRPO, and GSPO, in Table 7a with $N = 32$ and $K = 12$, evaluating on GSM8K and MATH500 at generation length 128. GSPO performs the strongest on both benchmarks, which we attribute to its sequence-level clipping being better suited to handle the variance introduced by subsampling K boundaries.

E.3 StepMerge configuration

Table 7b ablates the number of segments N and subsampled boundaries K, evaluating on GSM8K and MATH500 at generation length 128. Performance improves with larger N, as finer segmentation reduces the approximation error. Performance degrades only modestly as K decreases (at $N = 32$, from 81.0 at $K = 12$ to 78.0 at $K = 1$ on GSM8K), indicating the estimator remains effective even with few sampled boundaries. However, increasing K beyond 12 provides only marginal gains: at $N = 32$, $K = 12$ matches $K = 16$ on MATH500 while requiring fewer forward passes. We use $N = 32$, $K = 12$ for our main results.

E.4 Sampling temperature

Our method uses two temperatures: a token sampling temperature τ_{tok} and a position selection temperature τ_{pos} from our probabilistic position selection (Eq. 9), set to $\tau_{\text{tok}} = 0.9$ (matching prior work) and $\tau_{\text{pos}} = 0.5$ by default. We ablate both in Table 8 on GSM8K and MATH500 at generation length 128. Performance degrades as τ_{tok} is lowered, since less diverse rollouts weaken the group-relative advantage signal, and as τ_{pos} is raised, since more random selection unmasks low-confidence positions. The defaults strike a good balance, with results remaining reasonably stable across the tested range.

Algorithm	GSM8K	MATH500
RLOO	75.0	30.7
GRPO	79.5	36.4
GSPO	**81.0**	**37.4**

(a) RL algorithm.

N	K	GSM8K	MATH500
16	8	79.9	36.8
	16	80.6	37.2
32	1	78.0	35.4
	4	79.4	36.3
	8	80.3	36.9
	12	81.0	**37.4**
	16	**81.3**	**37.4**
64	8	79.8	36.8
	16	80.2	37.0

(b) StepMerge configuration: N segments, K sampled.

Table 7: Ablations on (a) RL algorithm and (b) StepMerge configuration. All use position-aware trajectory likelihood, evaluated on GSM8K and MATH500 at generation length 128. Underlined entries denote the configuration used in our main results. Best results in **bold**.

τ_{tok}	GSM8K	MATH500
0.5	79.1	35.9
0.9	**81.0**	**37.4**
1.5	80.5	37.0

(a) Token sampling temperature τ_{tok} (with $\tau_{\text{pos}} = 0.5$).

τ_{pos}	GSM8K	MATH500
0.1	80.7	37.3
0.5	**81.0**	**37.4**
1.0	79.0	35.7

(b) Position selection temperature τ_{pos} (with $\tau_{\text{tok}} = 0.9$).

Table 8: Ablation over the two sampling temperatures, evaluated on GSM8K and MATH500 at generation length 128. Underlined entries denote the configuration used in our main results. Best results in **bold**.

E.5 Unmasking-set size

At each denoising step, our method selects a set $\mathcal{U}_t$ of positions to unmask; for all main results we use $|\mathcal{U}_t| = 2$ per step, matching d1 (Zhao et al., 2025b), wd1 (Tang et al., 2026), and SPG (Wang et al., 2026). The selection distribution $p_{\text{unmask}}(\mathcal{U}_t)$ in Eq. 9 is defined for any $|\mathcal{U}_t|$, and Table 9 varies it from 2 to 8 on GSM8K with all other settings fixed. Our method remains robust across this range, degrading by at most 1.0 point from $|\mathcal{U}_t| = 2$ to 8. This modest drop is expected, since larger sets commit more positions per step and yield lower-quality rollouts during training.

$\lvert\mathcal{U}_t\rvert$	Gen Len 128	Gen Len 256	Gen Len 512
2	**81.0**	**85.9**	**87.1**
4	80.4	85.1	86.8
8	80.1	84.9	86.5

Table 9: Ablation over the unmasking-set size $|\mathcal{U}_t|$ on GSM8K. We unmask $|\mathcal{U}_t| = 2$ positions per step in all main results, matching the configuration of d1, wd1, and SPG. Underlined entries denote the configuration used in our main results. Best results in **bold**.

F StepMerge approximation

In our experiments, we approximate the token component of the policy gradient objective using the conditional distribution over each unmasked token, $\pi_\theta(z_i^t \mid \boldsymbol{c}, \boldsymbol{z}^{t-1})$. Specifically, following Wang et al. (2025a), we approximate the log-likelihood term in the token gradient by $\sum_{i \in \mathcal{U}_t} \log \pi_\theta(z_i^t \mid \boldsymbol{c}, \boldsymbol{z}^{t-1})$.

Computing trajectory log-probability in the policy gradient using above approximation requires evaluating all T denoising transitions, which is computationally expensive. Following Wang et al. (2025a), we adopt the StepMerge approximation, which partitions the trajectory into N segments with $N \ll T$. Let $t_0 = 0 < t_1 < \cdots < t_N = T$ denote segment boundaries. Instead of evaluating every transition, we approximate the trajectory likelihood using only the segment endpoints $\log \pi_\theta(\boldsymbol{z}^{t_i} \mid \boldsymbol{z}^{t_{i-1}}, \boldsymbol{c})$. Intuitively, this approximation treats multiple denoising steps as a single macro-transition, reducing the number of required forward passes from T to N while preserving the overall trajectory structure.

F.1 StepMerge approximation error

We analyze the approximation error induced by using StepMerge to evaluate the trajectory log-probability. The error has three sources: within-segment unmasking timing information lost by the macro-stage approximation, token-probability error from using the segment boundary context, and unmasking-probability error from using the same boundary context.

Proposition 1 (Macro-StepMerge approximation error)**.** *Let the T denoising steps be partitioned into N equal-length StepMerge segments with boundaries $0 = t_0 < t_1 < \cdots < t_N = T$. Let $I_m = \{t_{m-1} + 1, \ldots, t_m\}$ be segment m.*

Suppose that, for every segment m, every $t \in I_m$, and every retained position $i \in \mathcal{U}_t$,

$$\log \frac{\pi_{\boldsymbol{\theta}}(z_i^t \mid \boldsymbol{z}^{t-1}, \boldsymbol{c})}{\pi_{\boldsymbol{\theta}}(z_i^t \mid \boldsymbol{z}^{t_{m-1}}, \boldsymbol{c})} \le \epsilon_{\text{tok}}. \tag{12}$$

Suppose also that the unmasking distribution is stable within each segment:

$$\log \frac{p_{\text{unmask}}(\mathcal{U}_t \mid \boldsymbol{z}^{t-1}, \boldsymbol{c})}{p_{\text{unmask}}(\mathcal{U}_t \mid \boldsymbol{z}^{t_{m-1}}, \boldsymbol{c})} \le |\mathcal{U}_t| \epsilon_{\text{unmask}}. \tag{13}$$

Then the macro-StepMerge approximation error satisfies

$$D_N \le n \log\left(\frac{T}{N} + 1\right) + n\epsilon_{\text{tok}} + n\epsilon_{\text{unmask}}. \tag{14}$$

Proof. Let the full length-T trajectory be $\xi = (\boldsymbol{z}^0, \hat{\boldsymbol{z}}^1, \mathcal{U}_1, \boldsymbol{z}^1, \ldots, \boldsymbol{z}^{T-1}, \hat{\boldsymbol{z}}^T, \mathcal{U}_T, \boldsymbol{z}^T)$. For segment m, define the segment-level unmasking set $\bar{\mathcal{U}}^m = \bigcup_{t \in I_m} \mathcal{U}_t$. The macro-StepMerge block trajectory for the segment keeps only $\xi_{\text{SM}}^m = \left(\boldsymbol{z}^{t_{m-1}}, \bar{\mathcal{U}}^m, \boldsymbol{z}^{t_m}\right)$. It records the start state, the end state, and the positions unmasked somewhere in the segment, but not the exact within-segment step at which those positions were unmasked.

Let the skipped intermediate information be $H^m = \left(\boldsymbol{z}^{t_{m-1}+1}, \hat{\boldsymbol{z}}^{t_{m-1}+1}, \ldots, \boldsymbol{z}^{t_m-1}, \hat{\boldsymbol{z}}^{t_m}\right)$, and let the within-segment timing variable be $\Lambda^m = \left(\mathcal{U}_{t_{m-1}+1}, \mathcal{U}_{t_{m-1}+2}, \ldots, \mathcal{U}_{t_m}\right)$. Thus the full block can be represented as $(\xi_{\text{SM}}^m, H^m, \Lambda^m)$.

Let $\mathcal{F}_{m-1}$ denote the history up to the start of block m. By the chain rule of probability, the exact block conditional distribution is

$$\begin{aligned} p_m(\xi_{\text{SM}}^m, H^m, \Lambda^m \mid \mathcal{F}_{m-1}) = {} & p_m(\xi_{\text{SM}}^m \mid \mathcal{F}_{m-1}) p_m(H^m \mid \xi_{\text{SM}}^m, \mathcal{F}_{m-1}) \\ & \cdot p_m(\Lambda^m \mid H^m, \xi_{\text{SM}}^m, \mathcal{F}_{m-1}). \end{aligned} \tag{15}$$

The StepMerge block distribution on the same variables is given by:

$$\begin{aligned} q_m(\xi_{\text{SM}}^m, H^m, \Lambda^m \mid \mathcal{F}_{m-1}) = {} & q_m(\xi_{\text{SM}}^m \mid \mathcal{F}_{m-1}) p_m(H^m \mid \xi_{\text{SM}}^m, \mathcal{F}_{m-1}) \\ & \cdot p_m(\Lambda^m \mid \xi_{\text{SM}}^m, \mathcal{F}_{m-1}). \end{aligned} \tag{16}$$

The timing distribution in Eq. 16 does not condition on H^m, because StepMerge only knows the coarsened block trajectory ξ_{SM}^m when assigning within-segment timing and does not depend on H^m.

The full and StepMerge approximation trajectory distributions factor over blocks:

$$p(\xi) = \prod_{m=1}^{N} p_m(\xi_{\mathrm{SM}}^m, H^m, \Lambda^m \mid \mathcal{F}_{m-1}), \qquad q(\xi) = \prod_{m=1}^{N} q_m(\xi_{\mathrm{SM}}^m, H^m, \Lambda^m \mid \mathcal{F}_{m-1}). \tag{17}$$

Therefore, the KL divergence between full and StepMerge trajectories is

$$\begin{aligned} D_N &:= D_{\mathrm{KL}}(p(\xi) \,\|\, q(\xi)) \\ &= \sum_{m=1}^{N} \mathbb{E}_{\mathcal{F}_{m-1}\sim p} \left[D_{\mathrm{KL}}(p_m(\cdot \mid \mathcal{F}_{m-1}) \,\|\, q_m(\cdot \mid \mathcal{F}_{m-1})) \right]. \end{aligned} \tag{18}$$

Using Eqs. 15 and 16, the H^m terms cancel in the block log-ratio:

$$\log \frac{p_m}{q_m} = \log \frac{p_m(\Lambda^m \mid H^m, \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1})}{p_m(\Lambda^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1})} + \log \frac{p_m(\xi_{\mathrm{SM}}^m \mid \mathcal{F}_{m-1})}{q_m(\xi_{\mathrm{SM}}^m \mid \mathcal{F}_{m-1})}. \tag{19}$$

Taking expectations gives

$$D_N = D_{\mathrm{time}} + D_{\mathrm{coarse}}, \tag{20}$$

where

$$D_{\mathrm{time}} = \sum_{m=1}^{N} I(\Lambda^m; H^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}), \tag{21}$$

and D_{coarse} is the remaining KL between exact and StepMerge macro-block likelihoods. We first bound the timing term. By the conditional mutual information identity,

$$\begin{aligned} I(\Lambda^m; H^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}) &= H(\Lambda^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}) - H(\Lambda^m \mid H^m, \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}). \\ &\le H(\Lambda^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}), \end{aligned}$$

where the inequality follows from the nonnegativity of conditional entropy. For each $i \in \bar{\mathcal{U}}^m$, let R_i^m denote the exact within-segment unmasking time of position i. Since $|I_m| = T/N$, there are at most $T/N + 1$ possible timing choices under the macro-stage endpoint convention. Therefore,

$$H(R_i^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}) \le \log\left(\frac{T}{N} + 1\right). \tag{22}$$

The variable Λ^m is determined by the collection of per-position timing variables, so by subadditivity of entropy,

$$H(\Lambda^m \mid \xi_{\mathrm{SM}}^m, \mathcal{F}_{m-1}) \le |\bar{\mathcal{U}}^m| \log\left(\frac{T}{N} + 1\right). \tag{23}$$

Therefore,

$$D_{\mathrm{time}} \le \sum_{m=1}^{N} |\bar{\mathcal{U}}^m| \log\left(\frac{T}{N} + 1\right) = n \log\left(\frac{T}{N} + 1\right), \tag{24}$$

because every final token position is unmasked exactly once.

We now bound the coarse term. Marginalization cannot increase KL divergence so we have: $D_{\mathrm{KL}}(p(X,Y)\|q(X,Y)) \ge D_{\mathrm{KL}}(p(X)\|q(X))$ for random variables X and Y. Applying the KL inequality between ξ_{SM}^m and $(\mathcal{U}_{t_{m-1}+1}, \boldsymbol{z}^{t_{m-1}+1}, \ldots, \mathcal{U}_{t_m})$, we have ξ_{SM}^m and Y^m gives

$$\begin{aligned} D_{\mathrm{coarse}} &= \sum_{m=1}^{N} \mathbb{E}_{\mathcal{F}_{m-1}\sim p} \left[D_{\mathrm{KL}}(p_m(\xi_{\mathrm{SM}}^m \mid \mathcal{F}_{m-1}) \,\|\, q_m(\xi_{\mathrm{SM}}^m \mid \mathcal{F}_{m-1})) \right] \\ &\le \sum_{m=1}^{N} \mathbb{E}_{\mathcal{F}_{m-1}\sim p} \Big[D_{\mathrm{KL}}\Big(p_m((\mathcal{U}_{t_{m-1}+1}, \boldsymbol{z}^{t_{m-1}+1}, \ldots, \mathcal{U}_{t_m}, \boldsymbol{z}^{t_m}) \mid \mathcal{F}_{m-1}) \,\Big\| \\ &\qquad\qquad q_m((\mathcal{U}_{t_{m-1}+1}, \boldsymbol{z}^{t_{m-1}+1}, \ldots, \mathcal{U}_{t_m}, \boldsymbol{z}^{t_m}) \mid \mathcal{F}_{m-1}) \Big) \Big]. \end{aligned}$$

For a realized segment, the deterministic remasking factors cancel. The exact-to-StepMerge density ratio therefore decomposes into token and unmasking likelihood ratios:

$$\begin{aligned}\log \frac{p_m(\cdot \mid \mathcal{F}_{m-1})}{q_m(\cdot \mid \mathcal{F}_{m-1})} &= \sum_{t\in I_m}\sum_{i\in\mathcal{U}_t} \log \frac{\pi_{\boldsymbol{\theta}}(z_i^t \mid \boldsymbol{z}^{t-1}, \boldsymbol{c})}{\pi_{\boldsymbol{\theta}}(z_i^t \mid \boldsymbol{z}^{t_{m-1}}, \boldsymbol{c})} \\ &\quad + \sum_{t\in I_m} \log \frac{p_{\text{unmask}}(\mathcal{U}_t \mid \boldsymbol{z}^{t-1}, \boldsymbol{c})}{p_{\text{unmask}}(\mathcal{U}_t \mid \boldsymbol{z}^{t_{m-1}}, \boldsymbol{c})} \\ &\le \sum_{t\in I_m} |\mathcal{U}_t| \left(\epsilon_{\text{tok}} + \epsilon_{\text{unmask}}\right),\end{aligned}$$

where the last inequality follows from the two stability assumptions. Since each generated token is unmasked exactly once, $\sum_{m=1}^{N}\sum_{t\in I_m}|\mathcal{U}_t| = n$. Hence

$$D_{\text{coarse}} \le n(\epsilon_{\text{tok}} + \epsilon_{\text{unmask}}). \tag{25}$$

Combining this bound with Eq. 24 gives Eq. 14. □

G Incompleteness of the token-only policy gradient

In this section, we show that optimizing only the token gradient can miss a direction that improves the reward, in the following setting. Recall that at each denoising step t, the model produces probability $\pi_\theta(\cdot|z^t)$ over the full vocabulary at every position. Let $\mathcal{M}_t$ denote the set of positions that are still masked in state z^t. For each masked position $k \in \mathcal{M}_t$, we define an unmasking score v_k^t as the highest token confidence at position k. The generation process then selects the top positions with the highest unmasking scores to unmask, while the rest of the tokens get remasked. We define $\nabla_{\boldsymbol{\theta}} J_{|\text{pos}}$ and $\nabla_{\boldsymbol{\theta}} J_{|\text{tok}}$ to be the position part and token part of the policy gradient as follows.

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_{|\text{pos}} &= \mathop{\mathbb{E}}_{\hat{\boldsymbol{z}}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{c})}\left[R(\boldsymbol{c}, \boldsymbol{z}^T)\sum_{t=1}^{T}\Big(\nabla_\theta \log p_{\text{unmask}}(\mathcal{U}_t|\hat{\boldsymbol{z}}^t, \boldsymbol{z}^{t-1}, \boldsymbol{c})\Big)\right] \\ \nabla_{\boldsymbol{\theta}} J_{|\text{tok}} &= \mathop{\mathbb{E}}_{\hat{\boldsymbol{z}}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{c})}\left[R(\boldsymbol{c}, \boldsymbol{z}^T)\sum_{t=1}^{T}\Big(\nabla_\theta \log \pi_\theta(\boldsymbol{z}^t \mid \boldsymbol{c}, \boldsymbol{z}^{t-1})\Big)\right]\end{aligned}$$

Proposition 2. *There exists a finite-horizon MDLM generation problem and a linear policy class $\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{z}^t)$ such that, when the unmasking policy is defined by selecting positions with highest token confidence, the following holds:*

There exists a direction $\boldsymbol{v}$ such that the token-only policy gradient is orthogonal to this direction, and therefore, does not move parameters along the direction of $\boldsymbol{v}$:

$$\langle \nabla_{\boldsymbol{\theta}} J_{|\text{tok}}(\boldsymbol{\theta}), \boldsymbol{v}\rangle = 0 \quad \textit{for all } \boldsymbol{\theta}.$$

The true objective is not stationary along this direction, i.e.,

$$\exists \boldsymbol{\theta} \textit{ such that } \langle \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \boldsymbol{v}\rangle \neq 0.$$

In particular, there exists $\boldsymbol{\theta}$ and $\boldsymbol{\theta}' = \boldsymbol{\theta} + \epsilon\boldsymbol{v}$ for some $\epsilon > 0$ such that

$$J(\boldsymbol{\theta}') > J(\boldsymbol{\theta}),$$

while the token-only gradient provides no update in the $\boldsymbol{v}$ direction at $\boldsymbol{\theta}$.

Proof. We construct a two-position example with sequence length $n = 2$ and vocabulary $\{a, b\}$. At each step, exactly one masked position is selected and filled, producing $T = 3$ states and $T - 1 = 2$ transitions. The initial state is $z^1 = (\texttt{[MASK]}, \texttt{[MASK]})$, z^2 has one masked token, and z^3 is the completely unmasked sequence.

The reward is defined as

$$R(x_1, x_2) = \mathbb{I}\{x_1 = a\}.$$

We consider the linear policy class on the feature vector $\phi(z, k)$ for each masked position $k \in \{1, 2\}$. More specifically, the logits of the policy is given by

$$s_{\boldsymbol{\theta}}(z, k) = \boldsymbol{\theta}^\top \phi(z, k).$$

The probability of each symbol in vocabulary is given by the sigmoid of the logits.

$$p(a \mid z, k) = \sigma(s_{\boldsymbol{\theta}}(z, k)), \quad \text{and} \quad p(b \mid z, k) = 1 - \sigma(s_{\boldsymbol{\theta}}(z, k)).$$

We consider the following set of one-hot features in $\phi(z, k) \in \mathbb{R}^3$ to represent z and position k:

$$\phi(z, k) = \begin{cases} \boldsymbol{e}_1 & \text{if} \quad (z, k) = ((\texttt{[MASK]}, \texttt{[MASK]}),\ 1) \\ \boldsymbol{e}_2 & \text{if} \quad (z, k) = ((\texttt{[MASK]}, \texttt{[MASK]}),\ 2) \\ \boldsymbol{e}_3 & \text{if} \quad (z, k) = ((\texttt{[MASK]},\ a),\ 1) \quad \text{or} \quad (z, k) = ((\texttt{[MASK]},\ b),\ 1) \end{cases}$$

This gives us that

$$\begin{aligned} p(a \mid z^1,\ 1) = \sigma(\theta_1) \quad &\text{and} \quad p(a \mid z^1,\ 2) = \sigma(\theta_2) \\ p(a \mid z^2,\ 1) = \sigma(\theta_3) \quad &\text{and} \quad p(a \mid z^2,\ 2) = \sigma(0) = 0.5 \end{aligned}$$

We use the confidence score as the maximum token probability, that gives $v_k^t = \max_x \pi_\theta(x \mid z^t, k) = \sigma(|s_{\boldsymbol{\theta}}(z^t, k)|)$. When $|\mathcal{U}_t| = 1$, at denoising step t, we choose k^{th} position to decode with the following probability:

$$p_{\text{unmask}}(u_1 = k \mid z^t) = \frac{\exp(v_k^t/\tau)}{\exp(v_k^t/\tau) + \sum_{j \in \mathcal{M}_t \setminus \{k\}} \exp(v_j^t/\tau)}.$$

We now compute the expected reward $J(\boldsymbol{\theta})$. As $R(x_1, x_2) = \mathbb{I}\{x_1 = a\}$, we compute the probability of $x_1 = a$ under policy $\pi_{\boldsymbol{\theta}}$. The probability of the first position getting selected is $p_{\text{unmask}}(1 \mid z^1)$ and then the probability of position 1 getting filled with a is $\sigma(\theta_1)$. The probability of the second position getting selected at initial stage is $p_{\text{unmask}}(2 \mid z^1)$ and irrespective of the value at second position, $x_1 = a$ with probability $\sigma(\theta_3)$. Therefore, the total expected reward is

$$\begin{aligned} J(\boldsymbol{\theta}) &= p_{\text{unmask}}(1 \mid z^1)\pi_\theta(a \mid z^1, 1) + p_{\text{unmask}}(2 \mid z^1)\pi_\theta(a \mid z^2, 1) \\ &= \pi_\theta(a \mid z^1, 1) + p_{\text{unmask}}(2 \mid z^1)(\pi_\theta(a \mid z^2, 1) - p_{\text{val}}(a \mid z^1, 1)) \\ &= \sigma(\theta_1) \ + \ p_{\text{unmask}}(2 \mid z^1)(\sigma(\theta_3) - \sigma(\theta_1)) \end{aligned}$$

Zero token-only gradient along direction v. We define the direction $\boldsymbol{v} = (0, 1, 0)$. The token-only gradient along direction v is given by

$$\partial_{\theta_2} J_{|\text{tok}} = \mathop{\mathbb{E}}_{\xi \sim \pi_{\boldsymbol{\theta}}} \Big[\sum_t R(\xi)\, \partial_{\theta_2} \log \pi_\theta(x_{u_1} \mid z^t, u_1) \Big].$$

Observe that $\pi_\theta(x_{u_1} \mid z^t, u_1)$ depends on θ_2 only for $t = 1$ and $u_1 = 2$. Therefore, we can write the above as

$$\begin{aligned} \partial_{\theta_2} J_{|\text{tok}} &= \mathop{\mathbb{E}}_{\xi \sim \pi_{\boldsymbol{\theta}}} \Big[R(\xi) \mathbf{1}\{u_1 = 2\}\, \partial_{\theta_2} \log \pi_\theta(x_2 \mid z^1, u_1 = 2) \Big] \\ &= p_{\text{unmask}}(2 \mid z^1) \mathop{\mathbb{E}}_{\xi \sim \pi_{\boldsymbol{\theta}}} \Big[R(\xi)\, \partial_{\theta_2} \log \pi_\theta(x_2 \mid z^1, u_1 = 2) \ \mid \ u_1 = 2 \Big] \\ &= p_{\text{unmask}}(2 \mid z^1) \mathop{\mathbb{E}}_{x_2 \sim \pi_\theta(\cdot \mid z^1, 2)} \Big[\partial_{\theta_2} \log \pi_\theta(x_2 \mid z^1, u_1 = 2) \mathop{\mathbb{E}}_{u_2, x_1} \Big[R(\xi) \ \mid \ u_1 = 2,\ x_2 \Big] \Big] \\ &= p_{\text{unmask}}(2 \mid z^1)\, \sigma(\theta_3) \mathop{\mathbb{E}}_{x_2 \sim \pi_\theta(\cdot \mid z^1, 2)} \Big[\partial_{\theta_2} \log \pi_\theta(x_2 \mid z^1, u_1 = 2) \Big] \end{aligned}$$

We now show that the expectation in the above equation becomes zero. To show that, we write the expectation as sum and then change the order of expectation and derivative. The change in the order is valid because both are with respect to different variables.

$$\mathop{\mathbb{E}}_{x_2\sim\pi_\theta(\cdot|z^1,2)}\Big[\,\partial_{\theta_2}\log\pi_\theta(x_2\mid z^1,u_1=2)\Big]=\sum_{x_2}\pi_\theta(x_2\mid z^1,2)\frac{\partial_{\theta_2}\pi_\theta(x_2\mid z^1,2)}{\pi_\theta(x_2\mid z^1,2)}$$
$$=\partial_{\theta_2}\sum_{x_2}\pi_\theta(x_2\mid z^1,2)=\partial_{\theta_2}1=0$$

Increasing the true objective $J(\boldsymbol{\theta})$ along v direction. We show that for any parameter $\boldsymbol{\theta}$ satisfying $\sigma(\theta_1)>\sigma(\theta_3)$, the gradient along θ_2 improves the performance and $\partial_{\theta_2}J(\boldsymbol{\theta})\neq 0$.

Observe that for such a parameter vector $\boldsymbol{\theta}$, $\sigma(\theta_3)-\sigma(\theta_1)<0$ and hence $J(\boldsymbol{\theta})$ is strictly decreasing as $p_{\text{unmask}}(2\mid z^1)$ increases. Now $p_{\text{unmask}}(2\mid z^1)$ is a strictly increasing function of $v_2^1=\sigma(|\theta_2|)$, and $\sigma(|\theta_2|)$ is minimized at $\theta_2=0$ and strictly larger for $\theta_2\neq 0$. Therefore, keeping θ_1 and θ_3 fixed, the objective $J(\boldsymbol{\theta})$ is maximized when $\theta_2=0$. Therefore, for any $\boldsymbol{\theta}$ such that $\sigma(\theta_1)>\sigma(\theta_3)$ and $\theta_2\neq 0$, consider $\boldsymbol{\theta}'=\boldsymbol{\theta}+\epsilon\boldsymbol{v}$ such that $|\theta_2'|<|\theta_2|$, the true objective improves:

$$J(\boldsymbol{\theta}')>J(\boldsymbol{\theta}).$$

Nonzero gradient along v. We now show that the true objective exhibits a nonzero directional derivative along $\boldsymbol{v}=(0,1,0)$. To this end, we explicitly compute the gradient with respect to θ_2 and verify that it is nonvanishing under the stated conditions.

Differentiating the true objective with respect to θ_2 yields

$$\partial_{\theta_2}J(\boldsymbol{\theta})=\partial_{\theta_2}p_{\text{unmask}}(2\mid z^1)\,\big(\sigma(\theta_3)-\sigma(\theta_1)\big).$$

Since $\sigma(\theta_3)-\sigma(\theta_1)<0$, to conclude that $\partial_{\theta_2}J(\boldsymbol{\theta})\neq 0$, it suffices to establish that

$$\partial_{\theta_2}p_{\text{unmask}}(2\mid z^1)\neq 0\qquad\text{for }\theta_2\neq 0.$$

We now analyze the dependence of $p_{\text{unmask}}(2\mid z^1)$ on θ_2. Recall that $v_1^1=\sigma(|\theta_1|), v_2^1=\sigma(|\theta_2|)$, and

$$p_{\text{unmask}}(2\mid z^1)=\frac{\exp(v_2^1/\tau)}{\exp(v_1^1/\tau)+\exp(v_2^1/\tau)}.$$

Differentiating $p_{\text{unmask}}(2\mid z^1)$ gives

$$\partial_{\theta_2}p_{\text{unmask}}(2\mid z^1)=\frac{1}{\tau}\,p_{\text{unmask}}(2\mid z^1)\big(1-p_{\text{unmask}}(2\mid z^1)\big)\,\partial_{\theta_2}v_2^1.$$

Observe that the multiplicative factor $p_{\text{unmask}}(2\mid z^1)\big(1-p_{\text{unmask}}(2\mid z^1)\big)$ is strictly positive because $0<p_{\text{unmask}}(2\mid z^1)<1$. Therefore, the only remaining question is whether $\partial_{\theta_2}v_2^1$ vanishes. To compute this term, note that $v_2^1=\sigma(|\theta_2|)$ is a composition of smooth functions away from $\theta_2=0$. Differentiating yields

$$\partial_{\theta_2}v_2^1=\sigma(|\theta_2|)\big(1-\sigma(|\theta_2|)\big)\,\operatorname{sign}(\theta_2),\qquad\text{for }\theta_2\neq 0.$$

Since $\sigma(|\theta_2|)\in(0,1)$ for all finite θ_2, the factor $\sigma(|\theta_2|)(1-\sigma(|\theta_2|))$ is strictly positive, and $\operatorname{sign}(\theta_2)\neq 0$ whenever $\theta_2\neq 0$. Hence, $\partial_{\theta_2}v_2^1\neq 0$ for all $\theta_2\neq 0$.

Combining the above observations, we conclude that

$$\partial_{\theta_2}p_{\text{unmask}}(2\mid z^1)\neq 0\qquad\text{for all }\theta_2\neq 0,$$

and therefore

$$\partial_{\theta_2}J(\boldsymbol{\theta})\neq 0\qquad\text{whenever }\sigma(\theta_1)>\sigma(\theta_3)\text{ and }\theta_2\neq 0.$$

Finally, since $\boldsymbol{v} = (0, 1, 0)$, the directional derivative along $\boldsymbol{v}$ is given by

$$\langle \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \boldsymbol{v} \rangle = \partial_{\theta_2} J(\boldsymbol{\theta}).$$

Thus,

$$\langle \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \boldsymbol{v} \rangle \neq 0,$$

which establishes that the true objective is not stationary along $\boldsymbol{v}$.

□