Efficient Token-Level Policy Optimization for Masked Diffusion Models via Velocity Alignment

Anonymous ACL submission

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) successfully aligns autoregressive models in rigorous domains like mathematical reasoning and code generation. However, applying RLVR to Diffusion Large Language Models (dLLMs) is hindered by intractable sequence-level likelihoods, forcing existing methods to rely on unstable, high-variance approximations. We propose *Sequence Likelihood-Free Policy Optimization* (LFPO), an efficient framework that circumvents this bottleneck by formulating policy alignment as *discrete velocity alignment*. Inspired by continuous flow matching, LFPO eschews trajectory-level estimation and directly optimizes denoising logits via contrastive token-level updates, providing a stable, low-variance surrogate. Furthermore, predicting terminal solutions from intermediate states implicitly straightens the generation trajectory, enabling high-quality sampling with significantly fewer denoising iterations. Extensive experiments on reasoning and coding benchmarks demonstrate that LFPO matches or exceeds the accuracy of likelihood-based baselines while substantially accelerating inference.

1 Introduction

While *autoregressive* (AR) models have long dominated mathematical reasoning and code generation (Hui et al., 2024; Google, 2025; Anthropic, 2025), *Diffusion Large Language Models* (dLLMs) (Ye et al., 2025b; Nie et al., 2025) have recently emerged as a compelling alternative (Gong et al., 2025). Fundamentally diverging from the strict left-to-right sequential generation (Tian et al., 2024), dLLMs operate via a holistic denoising mechanism (Li et al., 2025). This non-autoregressive paradigm endows dLLMs with inherent advantages in global planning (Ye et al., 2025a) and iterative refinement, enabling simultaneous updates across the entire sequence structure (Havasi et al., 2025; Zhang et al., 2023). Furthermore, dLLMs facilitate significantly faster inference through parallel decoding strategies (Wu et al., 2025; Wang et al., 2025b). Despite these capabilities, effectively aligning dLLMs with verifiable correctness feedback remains a formidable open challenge (Zhao et al., 2025; Zhan, 2025).

Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024) is the gold standard for aligning AR models in reasoning tasks (Wei et al., 2025; Zhang et al., 2026). Consequently, recent methods attempt to transpose this paradigm to dLLMs (Zhao et al., 2025; Zhan, 2025) by casting the non-Markovian diffusion process as a *Markov Decision Process* (MDP) (Yang et al., 2025; Wang et al., 2025c) to apply *Policy Gradient* (PG) algorithms (Schulman et al., 2017b). However, standard PG estimation relies on exact model likelihoods to compute importance sampling weights (Schulman et al., 2017a; Zheng et al., 2025a), posing a critical bottleneck due to the computationally intractable sequence-level log-likelihoods of dLLMs (Wang et al., 2025a; Zhao et al., 2025). Lacking exact trajectory likelihoods, current approaches resort to step-wise ODE/SDE approximations (Chen et al., 2023; Song et al., 2021; Gong et al., 2025). In the high-dimensional discrete token space, these approximations inevitably introduce severe error accumulation (Zhao et al., 2025) and high computational overhead (Wang et al., 2025c), resulting in training instability and sub-optimal efficiency (Ni et al., 2025) (see Appendix A for extended related work).

Unlike autoregressive models, whose sequential inference naturally fits the MDP paradigm for straightforward likelihood computation (Xiong et al., 2025), imposing trajectory-level likelihood estimation on dLLMs is inherently inefficient, as their holistic denoising mechanism renders exact evaluation intractable (Li et al., 2024; Zhu et al., 2025). To circumvent this bottleneck, we analyze

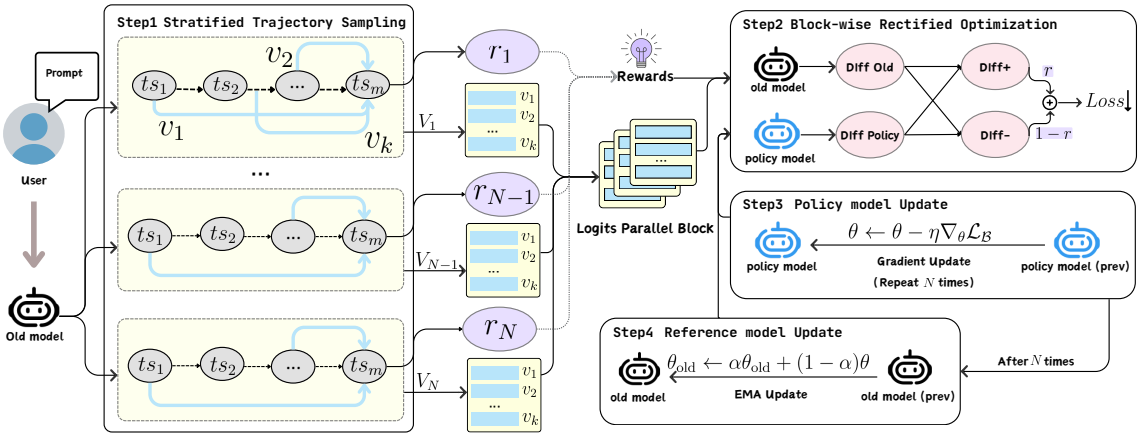

Figure 1: **Overview of the LFPO framework. Step 1** *Generate & Estimate*: π_{old} generates trajectories, with timesteps selected via *Stratified Trajectory Sampling* to minimize variance. **Step 2** *Block-Wise Optimization*: Data is partitioned into memory-efficient blocks for parallel logit computation. **Step 3** *Policy Update*: π_θ minimizes deviation from reward-induced implicit policies (π^+, π^-) via discrete vector field rectification. **Step 4** *Reference Update*: Stable EMA update of π_{old}. See Appendix C for method and Appendix E for implementation details.

dLLMs through the geometric lens of *Flow Matching* (FM) (Lipman et al., 2023), establishing that predicted logits for masked tokens effectively serve as a discrete proxy for the continuous velocity field v (Lou et al., 2024). Consequently, rather than approximating the intractable sequence integral $P_\theta(x)$, we propose aligning policies by directly rectifying these discrete logits toward high-reward trajectories via token-level *velocity alignment*. By performing localized updates in the logit space, we bypass the reliance on trajectory-level probability estimation, a strategy that aligns with recent breakthroughs in visual diffusion reinforcement learning (Tuo et al., 2025; Zheng et al., 2025b).

To actualize this vision, we propose **Sequence Likelihood-Free Policy Optimization (LFPO)**, an efficient token-level framework for aligning masked diffusion models. As illustrated in Figure 1, LFPO circumvents the sequence-likelihood bottleneck by operating directly on denoising logits. Employing a contrastive objective, it rectifies the denoising vector field by pulling predictions toward high-reward targets while repelling them from suboptimal ones. By eschewing noisy trajectory likelihood approximations, LFPO establishes a localized, low-variance empirical surrogate that significantly stabilizes training. Furthermore, to mitigate the inherent instability of step-wise denoising, LFPO incorporates a terminal anchoring mechanism that maps intermediate states directly to the final solution. This consistency objective encourages straighter probability flows, suppressing trajectory fluctuations and enabling high-quality sampling with drastically fewer iterations. Empirically, LFPO yields substantial accuracy improvements across reasoning and coding tasks while reducing inference latency by approximately 20%. Our core contributions are threefold:

- **Geometric Perspective:** Inspired by FM, we cast denoising logits as a discrete velocity proxy, deriving a token-level surrogate objective that entirely avoids intractable trajectory likelihood integration (Section 2).
- **LFPO Framework:** We propose a highly efficient RL framework that optimizes dLLMs via contrastive velocity alignment. This localized approach bypasses complex ODE backtracking, enabling robust and scalable training (Section 3; see Appendix C for supplementary conceptual discussions).
- **Empirical Superiority:** LFPO achieves a superior accuracy-efficiency trade-off on reasoning and coding benchmarks, matching or exceeding likelihood-based baselines while substantially accelerating inference via fewer denoising steps (Section 4).

2 Motivation: A Flow Matching Perspective

Standard reinforcement learning approaches for diffusion models are hindered by the intractability of likelihood computation. To overcome this, we reframe policy alignment through the geometric lens of FM. This section first reviews continuous

FM (Section 2.1), lifts discrete tokens to a probability simplex (Section 2.2), and derives a pointwise logit-gradient identity to motivate our token-level surrogate (Section 2.3).

2.1 Flow Matching

FM provides a simulation-free paradigm for modeling a probability path from a simple prior p_0 to a complex data distribution p_1 via an Ordinary Differential Equation (ODE): $\frac{d\boldsymbol{x}}{dt} = v_t(\boldsymbol{x})$ with $\boldsymbol{x}(0) \sim p_0$. The goal is to regress a neural vector field $v_\theta(\boldsymbol{x}_t, t)$ to match a target conditional vector field u_t. Under the standard *Conditional Optimal Transport* path (Onken et al., 2021), which interpolates linearly as $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$, the target velocity becomes a constant displacement:

$$u_t(\boldsymbol{x}_t|\boldsymbol{x}_1) = \frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{x}_1 - \boldsymbol{x}_0. \tag{1}$$

We minimize the expected *Mean Squared Error* between these velocities:

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t,\boldsymbol{x}_1,\boldsymbol{x}_0}\left[\|v_\theta(\boldsymbol{x}_t, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2\right]. \tag{2}$$

2.2 Lifting Discrete Tokens to the Probability Simplex

To bridge the gap between continuous FM and discrete dLLMs, we embed the vocabulary $\mathcal{V}$ as vertices on a probability simplex $\Delta^{V-1} \subset \mathbb{R}^V$ (Figure 2). In this geometric formulation, data tokens are represented as deterministic one-hot vectors $\boldsymbol{x}_1 \in \{0,1\}^V$ at the vertices, while the `[MASK]` token is defined as a fixed prior $\boldsymbol{m} \in \Delta^{V-1}$ at the simplex center (typically a uniform distribution). The forward diffusion process—transitioning from a masked to a revealed state—is modeled as a linear interpolation trajectory: $\boldsymbol{x}_t = (1-\alpha_t)\boldsymbol{m} + \alpha_t \boldsymbol{x}_1$, where $\alpha_t \in [0,1]$ dictates the noise schedule.

Because our token-level objective relies solely on the residual between the model and target directions, we define both displacements relative to an arbitrary shared origin $\boldsymbol{o}_t$ on the simplex (e.g., the mask prior $\boldsymbol{m}$ or the current state $\boldsymbol{x}_t$). This ensures the residual remains origin-invariant. The target displacement is thus defined as:

$$u_t^{(o)} := \boldsymbol{x}_1 - \boldsymbol{o}_t. \tag{3}$$

Crucially, a dLLM outputs a probability distribution $p_\theta(\cdot|\boldsymbol{x}_t) = \mathrm{Softmax}(\mathrm{Logits}_t)$ over the vocabulary (depicted as the point p_θ in the simplex).

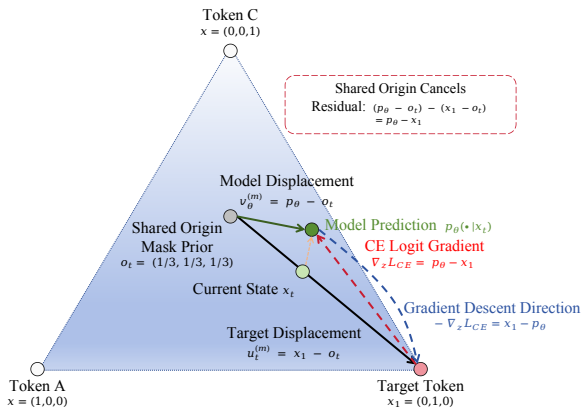

Figure 2: **Geometric Perspective of Discrete Lifting.** Visualization on a 2-simplex (Δ^2) where vertices represent tokens. Using a shared origin $\boldsymbol{o}_t$, the *Target Velocity* $u_t^{(o)}$ (black) points toward the ground truth $\boldsymbol{x}_1$, while the *Model Velocity* $v_\theta^{(o)}$ (green) points toward the model prediction p_θ. The dashed line indicates the origin-invariant residual direction, $p_\theta - \boldsymbol{x}_1$, which corresponds exactly to the CE logit gradient $\nabla\mathcal{L}_{CE}$.

Analogous to the continuous case, we define the **model velocity field** $v_\theta^{(o)}$ as the vector displacement from the shared origin to the prediction:

$$v_\theta^{(o)}(\boldsymbol{x}_t, t) := p_\theta(\cdot|\boldsymbol{x}_t) - \boldsymbol{o}_t. \tag{4}$$

This formulation interprets dLLM logits as local displacement parameters, rendering the velocity residual inherently origin-independent.

2.3 Pointwise Logit-Gradient Identity

We establish a identity linking standard token-level *Cross-Entropy* (CE) training to this geometric perspective. By evaluating the expected squared error between the model and target velocities, the shared origin $\boldsymbol{o}_t$ naturally cancels out:

$$\begin{aligned}\mathcal{L}_{\text{res}} &= \mathbb{E}\left[\|(p_\theta - \boldsymbol{o}_t) - (\boldsymbol{x}_1 - \boldsymbol{o}_t)\|^2\right] \\ &= \mathbb{E}\left[\|p_\theta - \boldsymbol{x}_1\|^2\right].\end{aligned} \tag{5}$$

Standard dLLM training minimizes the token-level CE loss:

$$\begin{aligned}\mathcal{L}_{CE} &= \mathbb{E}\left[-\sum_i (\boldsymbol{x}_1)_i \log(p_\theta)_i\right] \\ &= \mathbb{E}\left[-\log(p_\theta)_k\right],\end{aligned} \tag{6}$$

where k is the index of the ground-truth token. While these objectives reside in distinct functional spaces (log-likelihood versus L_2 distance) and are not globally equivalent, their pointwise logit gradients are mathematically identical:

Theorem 1 (Pointwise Logit-Gradient Identity)**.** *Let $\boldsymbol{z} \in \mathbb{R}^V$ be the unconstrained pre-softmax logits such that $p_\theta = \mathit{Softmax}(\boldsymbol{z})$. For any discrete time step t and any shared origin $\boldsymbol{o}_t$ fixed for the current input state, the gradient of $\mathcal{L}_{CE}$ with respect to the logits $\boldsymbol{z}$ exactly matches the residual error vector between the model velocity $v_\theta^{(o)}$ and the target velocity $u_t^{(o)}$:*

$$\nabla_{\boldsymbol{z}} \mathcal{L}_{CE} = p_\theta - \boldsymbol{x}_1 = v_\theta^{(o)} - u_t^{(o)}. \quad (7)$$

Remark. The detailed proof is provided in Appendix B and Appendix D details its theoretical scope and bounds the logit gradient. Theorem 1 establishes a pointwise logit-gradient identity rather than global equivalence between the CE and FM objectives. Practically, we leverage this shared-origin residual identity as a foundational design principle for a token-level surrogate, drastically reducing the reliance on sequence-level likelihood estimation. To bridge the distributional gap between the discrete masking process and the continuous FM interpolation path, we sample intermediate states directly from the model's empirical generation trajectories. Motivated by this empirical approximation, LFPO (Section 3) employs a contrastive objective to rectify the velocity field by pulling the model velocity $v_\theta^{(o)}$ toward high-reward Monte Carlo samples while repelling it from low-reward ones.

3 Policy Alignment via Velocity Alignment

We present LFPO, an RL framework for dLLMs. While inspired by continuous FM, its direct application is bottlenecked by the intractability of continuous ODEs in discrete token spaces. The pointwise logit-gradient identity established in Theorem 1 motivates a efficient implementation of velocity alignment via contrastive token-level updates.

3.1 Contrastive Velocity Alignment

In the supervised setting (Section 2), the optimization target is explicit: the ground-truth data defines a clear target displacement under the shared-origin convention (e.g., $\boldsymbol{u}_t^{(o)} = \boldsymbol{x}_1 - \boldsymbol{o}_t$). However, in the RLVR setting, such ground truth is absent. The model interacts with the environment to sample a trajectory τ, receiving only a sparse scalar reward $r(\tau)$. The fundamental challenge thus lies in defining a valid supervision target to effectively identify the correct velocity direction based solely on this scalar feedback. To address this, we draw inspiration from (Zheng et al., 2025b), which constructs implicit target velocity fields to guide continuous flow matching. We adapt this formulation into the discrete logit space. A critical design choice in LFPO is performing policy perturbations in the logit space rather than the probability space.

Justification for Log-Space Perturbation. As summarized in Table 6, applying linear perturbations directly in the probability space (e.g., $\pi' = \text{clip}(\pi_{\text{ref}} + \beta\Delta, 0, 1)$) frequently violates the $[0, 1]$ boundary constraints. This necessitates heuristic truncation, which fragments gradient continuity and yields an unstable optimization landscape. Conversely, performing perturbations in the logit space naturally bounds the policy within the simplex via the Softmax operation, maintaining a smooth and differentiable optimization landscape (see the expanded discussion on simplex constraints in Appendix C).

Let π_{ref} denote a frozen reference policy and π_θ the active policy. We define the velocity deviation as $\Delta(\boldsymbol{x}_t) = \log \pi_\theta(\cdot|\boldsymbol{x}_t) - \log \pi_{\text{ref}}(\cdot|\boldsymbol{x}_t)$. Based on this, we explicitly construct two implicit target policies in the log-space:

$$\log \pi^+(\cdot|\boldsymbol{x}_t) := \log \pi_{\text{ref}}(\cdot|\boldsymbol{x}_t) + \beta\Delta(\boldsymbol{x}_t), \quad (8)$$

$$\log \pi^-(\cdot|\boldsymbol{x}_t) := \log \pi_{\text{ref}}(\cdot|\boldsymbol{x}_t) - \beta\Delta(\boldsymbol{x}_t), \quad (9)$$

where $\beta > 0$ is a scalar coefficient. Geometrically, π^+ (*Implicit Positive Policy*) amplifies the current model's deviation from the reference, whereas π^- (*Implicit Negative Policy*) reverses it. These detached definitions provide the necessary gradient targets for the RLVR setting. For a detailed conceptual positioning against related diffusion RL and FM frameworks (Table 4), as well as a geometric comparison with naive reward-weighted cross-entropy (Table 5), please refer to Appendix C.

Leveraging the scalar reward $r(\tau) \in [0, 1]$, we formulate the LFPO objective as a dynamic interpolation between these implicit targets. We minimize the reward-weighted CE loss between the model's velocity field and these targets:

$$\mathcal{L}_{\text{LFPO}}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\mathbb{E}_t\Big[r(\tau) \cdot \text{CE}(\text{sg}(\pi^+), \pi_\theta) + (1 - r(\tau)) \cdot \text{CE}(\text{sg}(\pi^-), \pi_\theta)\Big]\Big], \quad (10)$$

where $\text{CE}(P, Q) = -\sum P(x) \log Q(x)$ and $\text{sg}(\cdot)$ denotes the stop-gradient operation. Crucially, because π^+ and π^- depend on the active policy π_θ,

these implicit target policies are computed without backpropagating through the target construction, ensuring unambiguous gradients and stable velocity rectification. Intuitively, high rewards ($r \to 1$) drive the policy to align with π^+ (pull), while low rewards ($r \to 0$) drive it toward π^- (push).

Directly optimizing Eq. (10) presents practical challenges. Naively sampling a single timestep t yields high gradient variance and optimization instability. To mitigate this, we employ *Stratified Trajectory Sampling* to ensure dense temporal coverage. However, this multi-sample approach imposes a severe GPU memory burden. Consequently, we introduce *Block-wise Gradient Accumulation* to resolve this memory bottleneck, as detailed in the following subsections.

3.2 Scalable and Stable Gradient Estimation

Stratified Trajectory Sampling. The masked diffusion process entails a discrete transition sequence $\boldsymbol{x}_T \to \cdots \to \boldsymbol{x}_0$. Naively sampling a single random timestep t yields high-variance gradients because the denoising difficulty varies non-uniformly across the generation stages. To address this, we propose *Stratified Trajectory Sampling* to guarantee dense temporal coverage (see Appendix F.2 for empirical validation of its gradient variance reduction). Specifically, for a trajectory of length L, we partition the valid timestep range into K uniform segments. During each training step, we sample exactly one timestep t_k from each segment:

$$t_k \sim \mathcal{U}\left[\lfloor k \cdot \frac{L}{K} \rfloor, \lfloor (k+1) \cdot \frac{L}{K} \rfloor - 1\right], \quad k = 0, \ldots, K-1. \tag{11}$$

Block-wise Gradient Accumulation. Estimating an accurate velocity direction to rectify the policy requires aggregating statistics from multiple trajectories. For a batch size B, we sample N trajectories per prompt and calculate gradients at K stratified timesteps per trajectory. This produces an intractably large computational graph of size $B \times N \times K$. To resolve this memory bottleneck, we implement *Block-wise Gradient Accumulation* (see Appendix F.1 for an analysis of its critical role in memory feasibility). We partition the total $B \times N \times K$ samples into smaller, memory-efficient blocks. The optimization follows a hybrid parallel-serial execution: we compute gradients in parallel within each block to leverage GPU parallelism, then serially accumulate these updates across blocks. This technique allows us to substantially scale the effective batch size under fixed hardware constraints, thereby minimizing the variance of the policy gradient.

Algorithm 1 Likelihood-Free Policy Optimization (LFPO)

1: **Input:** Dataset $\mathcal{D}$, Initial Policy π_θ, Params $\beta, N, K, \eta, \alpha$.
2: **Initialize:** Reference model parameters $\theta_{\text{old}} \leftarrow \theta$.
3: **repeat**
4: // **Phase 1: Generate & Estimate Rewards**
5: For batch queries $Q \subset \mathcal{D}$, sample N trajectories $\mathcal{Y}_q \sim \pi_{\text{old}}(\cdot|q)$.
6: Compute rewards $r_{\boldsymbol{y}} = \text{reward}(\boldsymbol{y})$ for all $\boldsymbol{y}$.
7: $\mathcal{D}_{\text{batch}} \leftarrow \bigcup_{q \in Q} \{(\boldsymbol{y}, r_{\boldsymbol{y}}) \mid \boldsymbol{y} \in \mathcal{Y}_q\}$.
8: // **Phase 2: Block-wise Rectified Optimization**
9: Partition $\mathcal{D}_{\text{batch}}$ into blocks $\{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$.
10: **for** each block $\mathcal{B}_m$ **do**
11: Sample stratified timesteps $\mathcal{T}_{\boldsymbol{y}} = \{t_1, \ldots, t_K\}$ for all $\boldsymbol{y} \in \mathcal{B}_m$.
12: Compute block loss (in parallel):
13: $$\mathcal{L}_{\mathcal{B}} = \sum_{(\boldsymbol{y},r) \in \mathcal{B}_m} \sum_{t \in \mathcal{T}_{\boldsymbol{y}}} \Big[r \cdot \text{CE}(\text{sg}(\pi^+), \pi_\theta) + (1-r) \cdot \text{CE}(\text{sg}(\pi^-), \pi_\theta) \Big]$$
14: Update policy parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\mathcal{B}}$.
15: **Delete** computation graph to free VRAM.
16: **end for**
17: // **Phase 3: Reference Update**
18: Update reference model via EMA: $\theta_{\text{old}} \leftarrow \alpha \theta_{\text{old}} + (1-\alpha)\theta$.
19: **until** convergence

3.3 LFPO

The complete training procedure of LFPO is summarized in Algorithm 1, and the unified hyperparameter configurations applied across all evaluated tasks are detailed in Appendix E.1. During the initial data collection phase, we sample N diverse trajectories per prompt using the reference policy π_{old} (line 4) and compute task-specific scalar rewards r (line 5). To optimize the policy under strict memory constraints, we apply a block-wise strategy (lines 8-15). Specifically, gradients are computed in par-

allel across K stratified timesteps within each partitioned block (lines 11-12), immediately followed by a parameter update (line 13) and explicit deletion of the computation graph to prevent VRAM overflow (line 14). Finally, to maintain training stability, the reference model π_{old} is updated via an *Exponential Moving Average* (EMA) with decay rate α at the end of each iteration (line 17).

4 Empirical Results

In this section, we provide a comprehensive evaluation of LFPO across code generation and mathematical reasoning domains to validate its empirical performance and the efficiency gains inherent to its likelihood-free design. Section 4.1 details the experimental setup. Section 4.2 analyzes the method's impact on performance, inference latency, and stability, followed by an in-depth ablation study of the contrastive velocity alignment in Section 4.3.

4.1 Experimental Setup

We evaluate LFPO using DiffuCoder (Gong et al., 2025) for code and LLaDA 8B (Nie et al., 2025) for reasoning. We benchmark against state-of-the-art RL baselines, including Diffu/Coupled-GRPO (Zhao et al., 2025; Gong et al., 2025), UniGRPO (Yang et al., 2025), SPG (Wang et al., 2025a), and AGRPO (Zhan, 2025). To ensure statistical rigor, all main results are reported as the mean and standard deviation across five independent random seeds.

Benchmarks and Data Strategy. We employ distinct evaluation protocols for each domain. For code generation, we target *Out-of-Domain Generalization*: models are trained on AceCode-87K (Zeng et al., 2025) and evaluated zero-shot on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b), EvalPlus (Liu et al., 2023a), and BigCodeBench (Zhuo et al., 2025). For mathematical reasoning, we follow standard *In-Domain Evaluation* using the training and test splits of GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), alongside general evaluations on GPQA (Rein et al., 2024) and PIQA (Bisk et al., 2019).

Implementation and Inference Protocol. Our framework utilizes sparse binary rewards, specifically functional correctness for code and symbolic accuracy for reasoning. We optimize the policy using AdamW combined with our memory-efficient block-wise strategy. To guarantee fair comparisons, we strictly standardize the inference protocol across all diffusion-based baselines. This unified protocol features a Cosine noise schedule with a maximum of $T = 2048$ steps, a top-k parallel update rule to maximize hardware utilization, and a confidence-based early stopping mechanism with a threshold of $\tau = 0.99$ and a patience of 3 steps. Comprehensive training hyperparameter configurations and the detailed inference protocol are provided in Appendix E.1 and Appendix E.2, respectively.

4.2 Main Results

Performance and Stability. We evaluate generation quality across downstream tasks to directly validate the core advantage of our likelihood-free design (Tables 1 and 2). LFPO consistently outperforms the strongest likelihood-based baseline, AGRPO, achieving the highest average scores in both code generation (60.8) and reasoning (67.1). Notably, it yields substantial absolute improvements over the base models on foundational benchmarks, including HumanEval (+3.6%), GSM8K (+9.9%), and MATH (+7.0%). We attribute this superior accuracy to our token-level velocity alignment, which provides a direct learning signal and avoids the high-variance importance sampling inherent to traditional RL methods. Crucially, bypassing sequence likelihood estimation yields unprecedented optimization stability. Across five independent random seeds, LFPO maintains an exceptionally low standard deviation (± 0.2 to ± 0.3), starkly contrasting with the high volatility of AGRPO (± 0.8 to ± 1.3). This confirms that our localized empirical surrogate provides a fundamentally more reliable training paradigm (see Appendix E.4 for formal statistical analysis).

Inference Efficiency via Straighter Trajectory Learning. A critical bottleneck for diffusion language models is the high computational overhead of iterative denoising. Table 3 highlights a definitive advantage of our approach: LFPO significantly accelerates inference while concurrently elevating accuracy. Compared to the base models, our method reduces the average inference steps by 41.8 steps for code generation and 159.0 steps for reasoning tasks. In stark contrast, likelihood-based baselines like AGRPO often degrade efficiency (e.g., increasing steps by 73.6 on MATH) to achieve marginal performance gains. This divergence stems from a fundamental difference in

Table 1: **Main Results on Code Generation Benchmarks.** We compare **LFPO** against the base model DiffuCoder and various reinforcement learning baselines. Values in red parentheses denote absolute improvements over the base model. Standard deviations ($\pm$) across 5 random seeds are reported for the strongest baseline and our method to demonstrate statistical stability. **LFPO (Pos. Only)** and **LFPO (Neg. Only)** denote ablation variants optimized solely with the positive or negative loss, respectively, while **LFPO (All Loss)** utilizes the full contrastive velocity rectification objective. Best results are highlighted in **bold**, and second-best are underlined.

Model	HumanEval	HumanEval+	MBPP	MBPP+	EvalPlus	BigCodeBench Full	BigCodeBench Hard	Avg.
DiffuCoder	72.0	65.2	75.1	61.9	63.6	35.7	12.2	55.1
+ diffu-GRPO	73.1 (+1.1)	66.1 (+0.9)	77.3 (+2.2)	67.3 (+5.4)	67.9 (+4.3)	39.2 (+3.5)	12.5 (+0.3)	57.6 (+2.5)
+ UniGRPO	74.5 (+2.5)	67.2 (+2.0)	78.9 (+3.8)	68.8 (+6.9)	**70.3** (+6.7)	42.6 (+6.9)	12.9 (+0.7)	59.3 (+4.2)
+ SPG	74.6 (+2.6)	69.1 (+3.9)	80.2 (+5.1)	69.2 (+7.3)	68.9 (+5.3)	42.1 (+6.4)	12.9 (+0.7)	59.6 (+4.5)
+ coupled-GRPO	73.2 (+1.2)	68.3 (+3.1)	78.6 (+3.5)	67.5 (+5.6)	67.9 (+4.3)	40.4 (+4.7)	10.8 (-1.4)	58.1 (+3.0)
+ AGRPO	75.3±1.1 (+3.3)	**70.2**±0.9 (+5.0)	80.3±1.0 (+5.2)	**71.7**±1.2 (+9.8)	69.3±0.8 (+5.7)	44.6±1.1 (+8.9)	**13.1**±0.9 (+0.9)	60.6±1.0 (+5.5)
+ LFPO (Pos. Only)	73.2 (+1.2)	67.3 (+2.1)	80.1 (+5.0)	68.2 (+6.3)	68.3 (+4.7)	43.1 (+7.4)	12.4 (+0.2)	58.9 (+3.8)
+ LFPO (Neg. Only)	73.1 (+1.1)	67.2 (+2.0)	79.5 (+4.4)	68.5 (+6.6)	68.9 (+5.3)	43.4 (+7.7)	12.5 (+0.3)	59.0 (+3.9)
+ LFPO (All Loss)	**75.6**±0.2 (+3.6)	70.1±0.3 (+4.9)	**81.6**±0.2 (+6.5)	71.3±0.3 (+9.4)	69.5±0.2 (+5.9)	**44.8**±0.3 (+9.1)	12.9±0.2 (+0.7)	**60.8**±0.2 (+5.7)

Table 2: **Main Results on Reasoning Benchmarks.** We compare **LFPO** against the base model LLaDA 8B and various reinforcement learning baselines. Formatting conventions, standard deviation reporting across 5 random seeds, and the nomenclature for ablation variants strictly follow those detailed in Table 1.

Model	GSM8K	MATH	GPQA	Hellaswag	WinoGrande	PIQA	Avg.
LLaDA 8B	69.7	30.6	25.3	70.3	75.8	71.6	57.2
+ diffu-GRPO	74.3 (+4.6)	33.7 (+3.1)	26.1 (+0.8)	80.1 (+9.8)	83.2 (+7.4)	79.3 (+7.7)	62.8 (+5.6)
+ UniGRPO	77.2 (+7.5)	34.5 (+3.9)	26.2 (+0.9)	82.3 (+12.0)	84.7 (+8.9)	81.2 (+9.6)	64.4 (+7.2)
+ SPG	76.2 (+6.5)	34.1 (+3.5)	25.9 (+0.6)	83.1 (+12.8)	84.3 (+8.5)	80.9 (+9.3)	64.1 (+6.9)
+ coupled-GRPO	75.1 (+5.4)	33.9 (+3.3)	25.7 (+0.4)	82.5 (+12.2)	85.6 (+9.8)	83.2 (+11.6)	64.3 (+7.1)
+ AGRPO	78.3±1.3 (+8.6)	36.1±0.9 (+5.5)	**27.3**±1.1 (+2.0)	85.1±1.0 (+14.8)	**87.3**±1.2 (+11.5)	85.6±0.8 (+14.0)	66.6±1.0 (+9.4)
+ LFPO (Pos. Only)	78.3 (+8.6)	35.2 (+4.6)	26.1 (+0.8)	84.2 (+13.9)	83.1 (+7.3)	80.6 (+9.0)	64.6 (+7.4)
+ LFPO (Neg. Only)	78.1 (+8.4)	34.9 (+4.3)	26.3 (+1.0)	84.3 (+14.0)	83.7 (+7.9)	79.6 (+8.0)	64.5 (+7.3)
+ LFPO (All Loss)	**79.6**±0.3 (+9.9)	**37.6**±0.2 (+7.0)	27.1±0.3 (+1.8)	**85.7**±0.2 (+15.4)	86.9±0.2 (+11.1)	**85.9**±0.2 (+14.3)	**67.1**±0.2 (+9.9)

optimization objectives. Standard likelihood maximization implicitly forces models to mimic specific, often meandering denoising trajectories observed in the training data. Conversely, by casting generation as a geometric velocity alignment problem, LFPO encourages the model to learn a straightened probability flow from the mask prior to the target data distribution. This empirical straightening actively shortens the generation trajectory, enabling the model to converge to high-quality solutions with drastically fewer intermediate steps (see Appendix G for a visual case study).

Training Convergence and Deconstructing Acceleration. Beyond inference efficiency, Figure 3 highlights the rapid training convergence of our framework. Compared to the AGRPO baseline, LFPO matches peak performance significantly faster in terms of GPU hours (8.0× on HumanEval and MATH, 6.7× on MBPP-Diamond, and 4.4× on HellaSwag) (detailed hardware configurations and wall-clock costs are provided in Appendix E.3). We attribute this acceleration to two synergistic factors. First, our contrastive dual-policy design yields high sample efficiency. By anchoring intermediate states to high-reward terminal predictions, this push-and-pull mechanism encourages straightened probability flows, which enables the model to reach correct solutions in fewer iterative steps and accelerates data collection. Second, our tailored gradient estimation techniques (Section 3.2) optimize both update stability and computational throughput. Specifically, Stratified Trajectory Sampling guarantees dense temporal coverage to reduce gradient variance, providing a highly stable optimization signal per step. Concurrently, block-wise parallelization minimizes computational overhead by operating directly in the discrete logit space to update multiple tokens simultaneously, avoiding the poorly scaling sequential backtracking of traditional methods. Ultimately, the combination of high sample efficiency, low-variance updates, and parallelized computation drives the superior wall-clock convergence observed across our evaluations.

4.3 Ablation Study and Sensitivity Analysis

Ablation on Contrastive Components. To dissect the geometric mechanisms driving LFPO, we analyze variants optimized with partial objectives:

Table 3: **Unified Efficiency Analysis.** We report the average inference steps per problem across code generation and reasoning benchmarks. **Avg.** represents the mean steps within each task category. The values in red parentheses denote the reduction in inference steps (efficiency improvement) compared to the Base Model. *Base Model* refers to DiffuCoder for code tasks and LLaDA 8B for reasoning tasks. Lower values (↓) indicate better efficiency.

Model	Code Generation				Reasoning			
	HumanEval	MBPP	BCB-Full	Avg.	GSM8K	MATH	Hellaswag	Avg.
Base Model†	253.7	317.2	426.3	332.4	1034.2	1536.3	1036.7	1202.4
+ diffu-GRPO	254.3 (+0.6)	305.4 (-11.8)	430.7 (+4.4)	330.1 (-2.3)	1048.4 (+14.2)	1547.4 (+11.1)	1027.8 (-8.9)	1207.9 (+5.5)
+ UniGRPO	248.0 (-5.7)	319.3 (+2.1)	443.3 (+17.0)	336.9 (+4.5)	1020.3 (-13.9)	1539.3 (+3.0)	985.8 (-50.9)	1181.8 (-20.6)
+ SPG	243.7 (-10.0)	305.8 (-11.4)	416.7 (-9.6)	322.1 (-10.3)	1071.0 (+36.8)	1576.7 (+40.4)	1009.1 (-27.6)	1218.9 (+16.5)
+ coupled-GRPO	262.2 (+8.5)	326.5 (+9.3)	431.2 (+4.9)	340.0 (+7.6)	1015.9 (-18.3)	1601.6 (+65.3)	1025.8 (-10.9)	1214.4 (+12.0)
+ AGRPO	254.2 (+0.5)	313.9 (-3.3)	438.5 (+12.2)	335.5 (+3.1)	1004.8 (-29.4)	1609.9 (+73.6)	1063.6 (+26.9)	1226.1 (+23.7)
+ LFPO (Pos. Only)	223.6 (-30.1)	292.3 (-24.9)	373.4 (-52.9)	296.4 (-36.0)	911.3 (-122.9)	1355.3 (-181.0)	893.1 (-143.6)	1053.2 (-149.2)
+ LFPO (Neg. Only)	234.7 (-19.0)	286.7 (-30.5)	377.3 (-49.0)	299.6 (-32.8)	909.7 (-124.5)	1367.9 (-168.4)	891.2 (-145.5)	1056.3 (-146.1)
+ LFPO (All Loss)	**218.3** (-35.4)	**282.3** (-34.9)	**371.2** (-55.1)	**290.6** (-41.8)	**891.2** (-143.0)	**1348.9** (-187.4)	**890.2** (-146.5)	**1043.4** (-159.0)

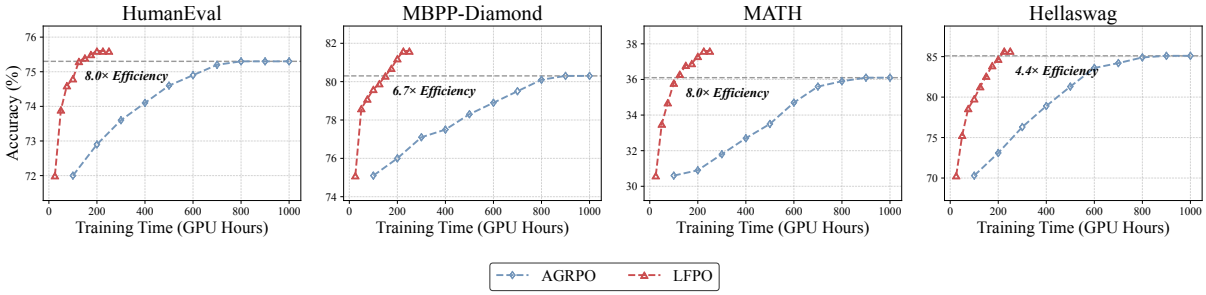

Figure 3: **Convergence Analysis on Code and Reasoning Tasks.** We plot accuracy progression against training time (GPU Hours). The red and blue curves denote our proposed LFPO and the AGRPO baseline, respectively. The horizontal dashed line marks the baseline's final converged accuracy. Notably, LFPO requires less time to match or surpass this peak performance, underscoring LFPO sample efficiency and rapid convergence.

Pos. Only (Attraction) and *Neg. Only* (Repulsion). As shown in Tables 1 and 2, both variants yield improvements over the base models, yet neither matches the overall performance of the complete *All Loss* objective. From a geometric perspective on the probability simplex (Figure 1), the *Pos. Only* term acts as an attraction force, pulling the model's velocity $v_\theta^{(o)}$ toward the vertex of the correct token $\boldsymbol{x}_1$. Conversely, the *Neg. Only* term acts as a repulsion force, pushing the velocity away from incorrect vertices. The superior performance of the combined objective suggests that shaping the vector field benefits from a contrastive approach: simultaneously encouraging correct directions while suppressing deviations into low-reward regions yields the most reliable generative flow.

Sensitivity Analysis and Case Studies. Extensive sensitivity analyses (Appendix F) dissect the empirical gains of LFPO across various configurations. First, ablations on core hyperparameters and negative controls (Appendices F.2 and F.1) demonstrate that our block-wise accumulation strictly bounds peak VRAM, while validating the absolute necessity of logit-space perturbation for optimization stability. Second, evaluations of inference decoding parameters (Appendix F.3) reveal that a Cosine masking schedule combined with a high confidence threshold ($\tau = 0.99$) yields an optimal accuracy-efficiency balance, slashing denoising steps without degrading quality. Third, generalization tests on the Dream architecture (Appendix F.4) confirm that our contrastive alignment framework is architecture-agnostic and universally applicable. Finally, qualitative case studies (Appendix G) visually corroborate our core theoretical claim, illustrating exactly how LFPO anchors to correct terminal states early to straighten probability flows and bypass meandering denoising steps.

5 Conclusion

We introduced LFPO, a likelihood-free alignment framework for discrete diffusion language models that bypasses high-variance RL likelihood estimation via contrastive velocity alignment. Evaluations on reasoning and coding tasks demonstrate that LFPO concurrently delivers superior accuracy, optimization stability, and inference acceleration.

Limitations

While LFPO presents a highly effective empirical framework, we acknowledge several limitations that warrant future investigation.

First, the method is constrained by a cold-start bottleneck. Because our contrastive optimization relies on anchoring to high-reward terminal states, the base model must possess a non-zero zero-shot success rate to discover initial valid trajectories. Second, although block-wise parallelization mitigates VRAM constraints, the overall training process maintains a substantial computational cost due to the necessity of generating N rollouts and evaluating K stratified timesteps per trajectory. Third, the current framework is exclusively tailored for verifiable sparse reward scenarios (e.g., exact mathematical matching and code execution). It does not trivially extend to subjective alignment tasks governed by dense human preference models. Fourth, unlike classical reinforcement learning algorithms, LFPO operates as an empirical velocity surrogate and does not currently provide strict theoretical guarantees for monotonic policy improvement or global convergence. Finally, despite our reference-anchored design mitigating certain reward over-optimization risks, the potential for reward hacking and benchmark overfitting remains a persistent challenge necessitating careful monitoring in practical deployments.

References

Anthropic. 2025. Claude code: Best practices for agentic coding. https://www.anthropic.com/engineering/claude-code-best-practices. Accessed: 2025.

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

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Sitan Chen, Sinho Chewi, Holden Lee, Yuanzhi Li, Jianfeng Lu, and Adil Salim. 2023. The probability flow ODE is provably fast. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Chenghao Fan, Wen Heng, Bo Li, Sichen Liu, Yuxuan Song, Jing Su, Xiaoye Qu, Kai Shen, and Wei Wei. 2026. Stable-diffcoder: Pushing the frontier of code diffusion large language model. *Preprint*, arXiv:2601.15892.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. Diffucoder: Understanding and improving masked diffusion models for code generation. *Preprint*, arXiv:2506.20639.

Google. 2025. Gemini 3. https://aistudio.google.com/models/gemini-3. Accessed: 2025.

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

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Henry Li, Ronen Basri, and Yuval Kluger. 2024. Likelihood training of cascaded diffusion models via hierarchical volume-preserving maps. In *The Twelfth International Conference on Learning Representations*.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. 2023. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. 2025. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xingchao Liu, Chengyue Gong, and qiang liu. 2023b. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Jinjie Ni, Qian Liu, Chao Du, Longxu Dou, Hang Yan, Zili Wang, Tianyu Pang, and Michael Qizhe Shieh. 2025. Training optimal large diffusion language models. *Preprint*, arXiv:2510.03280.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. 2021. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. *Preprint*, arXiv:2006.00104.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization. *Preprint*, arXiv:1502.05477.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. *Preprint*, arXiv:2011.13456.

Keyu Tian, Yi Jiang, Zehuan Yuan, BINGYUE PENG, and Liwei Wang. 2024. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ping Tuo, Zezhu Zeng, Jiale Chen, and Bingqing Cheng. 2025. Scalable multitemperature free energy sampling of classical ising spin states. *Preprint*, arXiv:2503.08063.

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. 2025a. Spg: Sandwiched policy gradient for masked diffusion language models. *Preprint*, arXiv:2510.09541.

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

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

Chenxing Wei, Jiarui Yu, Ying Tiffany He, Hande Dong, Yao Shu, and Fei Yu. 2025. Redit: Reward dithering for improved LLM policy optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and

Enze Xie. 2025. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *Preprint*, arXiv:2505.22618.

Luolin Xiong, Haofen Wang, Xi Chen, Lu Sheng, Yun Xiong, Jingping Liu, Yanghua Xiao, Huajun Chen, Qing-Long Han, and Yang Tang. 2025. Deepseek: Paradigm shifts and technical evolution in large ai models. *Preprint*, arXiv:2507.09955.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. 2025. MMaDA: Multimodal large diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025a. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *The Thirteenth International Conference on Learning Representations*.

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

Runpeng Yu, Qi Li, and Xinchao Wang. 2025. Discrete diffusion in large language and multimodal models: A survey. *Preprint*, arXiv:2506.13759.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. 2025. ACECODER: Acing coder RL via automated test-case synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12023–12040, Vienna, Austria. Association for Computational Linguistics.

Anthony Zhan. 2025. Principled and tractable rl for reasoning with diffusion language models. *Preprint*, arXiv:2510.04019.

Jianqing Zhang, Zhezheng Hao, Wei Xia, Hande Dong, Hong Wang, Chenxing Wei, Yuyan Zhou, Yubin Qi, Qiang Lin, and Jian Cao. 2026. Gapo: Robust advantage estimation for real-world code llms. *Preprint*, arXiv:2510.21830.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua M. Susskind, and Navdeep Jaitly. 2023. PLANNER: Generating diversified paragraph via latent language diffusion model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025a. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. 2025b. Diffusionnft: Online diffusion reinforcement with forward process. *Preprint*, arXiv:2509.16117.

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

Terry Yue Zhuo, Vu Minh Chien, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen GONG, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, and 14 others. 2025. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*.

A Related Work

Diffusion Large Language Models. Discrete denoising diffusion probabilistic models (D3PMs) (Austin et al., 2021a) have recently emerged as a compelling non-autoregressive paradigm for text generation (Yu et al., 2025). Unlike traditional autoregressive models that generate tokens strictly left-to-right, dLLMs generate text via parallel iterative unmasking, offering flexible bidirectional context modeling and potential improvements in decoding efficiency (Ye et al., 2025b). Recent large-scale implementations, such as LLaDA (Nie et al., 2025) and DiffuCoder (Gong et al., 2025), have demonstrated that dLLMs can achieve language modeling performance competitive with their autoregressive counterparts (Fan et al., 2026). However, while pre-training endows these models with strong general capabilities, effective alignment techniques to enhance their performance remain underexplored.

Reinforcement Learning for Diffusion Alignment. To bridge this gap, recent research has focused on adapting PG methods to the discrete diffusion setting. A prominent line of work utilizes the *Group Relative Policy Optimization* (GRPO) (Shao et al., 2024) framework to enable critic-free optimization. Diffu-GRPO (Zhao et al., 2025) pioneered this direction, introducing the first PG algorithm tailored for masked dLLMs to improve mathematical reasoning. UniGRPO (Yang et al., 2025) extended this framework to multimodal domains, unifying reasoning and generation tasks with diversified reward modeling. To address the gradient bias caused by the intractable likelihood of diffusion trajectories, *Sandwiched Policy Gradient* (SPG) (Wang et al., 2025a) proposed leveraging both upper and lower likelihood bounds. More recently, Coupled-GRPO (Gong et al., 2025) and AGRPO (Zhan, 2025) have been proposed to further improve stability and sample efficiency. Specifically, AGRPO achieves strong results by employing Monte Carlo sampling for unbiased policy gradient estimation. The distinction between AGRPO and LFPO is therefore not that AGRPO lacks Monte Carlo estimation, but that the two methods estimate different objects. AGRPO remains within a trajectory-level policy-gradient framework and uses Monte Carlo samples to estimate policy-gradient quantities for diffusion trajectories. LFPO instead constructs detached token-level implicit targets and optimizes a local CE/logit surrogate motivated by the shared-origin velocity residual. This changes the optimization signal from trajectory likelihood estimation to token-level vector-field rectification, which is where our observed variance and efficiency gains arise.

Despite these advancements, the aforementioned methods predominantly remain within the paradigm of maximizing the likelihood (or its surrogates) of high-reward trajectories. We argue that strictly adhering to this likelihood-based objective creates a bottleneck for dLLMs. Rather than persisting with likelihood maximization, we posit that dLLMs require a native RL formulation grounded in their geometric nature. In the continuous domain, FM (Lipman et al., 2023; Liu et al., 2023b) has successfully replaced likelihood objectives with stable vector field regression. While translating this to the discrete domain is non-trivial, recent theoretical works (Zheng et al., 2025b; Liu et al., 2025) have begun to uncover the geometric structures of discrete diffusion. Building on these insights, our work identifies a pointwise logit-gradient connection between token-level CE updates and discrete velocity residuals. This perspective motivates us to reframe alignment not as trajectory probability maximization, but as velocity rectification, leading to the stable and memory-efficient optimization strategy that we introduce in the following section.

B Proof of Theorem 1

Proof. We analyze the gradient of the Cross-Entropy loss $\mathcal{L}_{CE}$ with respect to the i-th component of the pre-softmax logits, z_i. Let $p_k = \frac{\exp(z_k)}{\sum_{j=1}^{V} \exp(z_j)}$ and the loss be $\mathcal{L}_{CE} = -\sum_{j=1}^{V} x_{1,j} \log p_j$. Applying the

chain rule, we expand the partial derivative:

$$\frac{\partial \mathcal{L}_{CE}}{\partial z_i} = \sum_{j=1}^{V} \frac{\partial \mathcal{L}_{CE}}{\partial p_j} \frac{\partial p_j}{\partial z_i} \tag{12}$$

$$= \sum_{j=1}^{V} \left[-\frac{x_{1,j}}{p_j} \right] [p_j(\delta_{ij} - p_i)] \tag{13}$$

$$= -\sum_{j=1}^{V} x_{1,j}(\delta_{ij} - p_i) \tag{14}$$

$$= -\left[\sum_{j=1}^{V} x_{1,j}\delta_{ij} - \sum_{j=1}^{V} x_{1,j} p_i \right] \tag{15}$$

$$= -\left[x_{1,i} - p_i \sum_{j=1}^{V} x_{1,j} \right] \tag{16}$$

Since $\boldsymbol{x}_1$ is a valid one-hot probability distribution ($\sum_{j=1}^{V} x_{1,j} = 1$), the scalar derivative simplifies to $\frac{\partial \mathcal{L}_{CE}}{\partial z_i} = p_i - x_{1,i}$. Expressing this in vector notation over the vocabulary V, let $\boldsymbol{o}_t \in \Delta^{V-1}$ denote any shared origin fixed for the current input state (e.g., the mask prior $\boldsymbol{m}$ or the current denoising state $\boldsymbol{x}_t$). Define the model velocity field $v_\theta^{(o)} = \boldsymbol{p} - \boldsymbol{o}_t$ and the target velocity field $u_t^{(o)} = \boldsymbol{x}_1 - \boldsymbol{o}_t$. Their residual cancels the shared origin:

$$v_\theta^{(o)} - u_t^{(o)} = (\boldsymbol{p} - \boldsymbol{o}_t) - (\boldsymbol{x}_1 - \boldsymbol{o}_t) \tag{17}$$

$$= \boldsymbol{p} - \boldsymbol{x}_1 \tag{18}$$

$$= \nabla_{\boldsymbol{z}} \mathcal{L}_{CE} \tag{19}$$

This yields the pointwise logit-gradient identity. □

C Additional Method Details

In this section, we provide supplementary discussions to deeply contextualize the algorithmic design choices of LFPO within the broader landscape of diffusion reinforcement learning and discrete policy optimization.

Methodological Positioning and Contributions. Table 4 delineates the conceptual boundaries between LFPO and closely related paradigms. While continuous diffusion RL and flow-matching frameworks (e.g., DiffusionNFT, Flow-GRPO) inspire our positive and negative target formulation, they inherently operate in continuous spaces and do not address discrete sequence likelihoods. Conversely, standard dLLM reinforcement learning methods like AGRPO rely on policy-gradient estimators that require approximating high-variance trajectory likelihoods. LFPO bridges this gap by adapting the contrastive target into a discrete logit-space objective. This unique instantiation provides a token-level surrogate loss that completely circumvents trajectory likelihood estimation. Furthermore, we support this theoretical adaptation with a highly scalable block-wise parallelization strategy and a standardized evaluation protocol for reproducible benchmarking.

Geometric Interpretation vs. Naive Cross-Entropy. Table 5 details why the geometric flow-matching perspective is fundamentally superior to a naive reward-weighted cross-entropy (CE) approach. Standard reward-weighted CE essentially acts as an attraction-only heuristic, pulling the model exclusively toward sampled high-reward trajectories. By framing the generation process as a shared-origin velocity alignment problem, LFPO formally constructs explicit positive and negative implicit policies (π^+ and π^-). This contrastive formulation actively pulls the generation toward high-reward directions while simultaneously repelling it from low-reward deviations. As demonstrated by the performance gap between the *Pos. Only*

Table 4: **Positioning of LFPO relative to closely related RL/FM methods.** The positive–negative implicit target idea is inspired by prior diffusion RL/FM work. The main contribution of LFPO is the discrete dLLM instantiation: token-level CE/logit supervision that avoids sequence-likelihood estimation, together with scalable sampling/optimization and standardized dLLM RLVR evaluation.

Component	DiffusionNFT / Flow-GRPO	AGRPO	LFPO
Positive–negative or reward-induced target direction	Inspired by this line of diffusion RL/FM work (Zheng et al., 2025b; Liu et al., 2025)	Not the central design	Adopted as a detached contrastive target in discrete logit space
Discrete masked-dLLM token-level formulation	Continuous/image or flow-matching setting	Likelihood-based dLLM policy-gradient estimator	Token-level CE/logit surrogate for masked dLLMs
Sequence/trajectory likelihood handling	Not designed for masked dLLM sequence likelihood	Estimates or approximates policy-gradient quantities	Avoids sequence/trajectory likelihood estimation; retains token-level CE/logit supervision
Perturbation space	Model/flow-space target construction	Policy-gradient style update	Logit-space perturbation to keep simplex constraints smooth
Temporal credit assignment	Method-specific continuous-time sampling	Trajectory-level PG sampling	Stratified timestep sampling over denoising trajectories
Scalable implementation	Not focused on dLLM token-logit memory	Monte Carlo estimator for dLLMs	Block-wise logit parallelization for $B \times N \times K$ training samples
Evaluation scope	Diffusion RL/FM benchmarks	dLLM RL baselines	Standardized dLLM RLVR evaluation on code and reasoning with shared decoding protocol

variant and the complete *All Loss* variant in our ablation studies, this dual push-and-pull mechanism is vital for maintaining a reliable generative flow.

Table 5: **Why the FM view changes the implementation.** The algebraic CE gradient gives a tractable token-level update, while the shared-origin velocity view motivates the contrastive target construction used by LFPO.

Aspect	Naive reward-weighted CE	LFPO
Target construction	Pull toward sampled high-reward outputs	Detached positive/negative implicit policies (π^+, π^-)
Geometric view	Supervised attraction heuristic	Shared-origin residual rectification
Update effect	Attraction-only signal	Pull high-reward directions and push away low-reward directions
Empirical check	Pos. Only variant	All Loss variant in Tables 1–2

Simplex Constraints and Logit-Space Perturbation. A critical technical detail of LFPO is the space in which the policy perturbation is applied, as compared in Table 6. Applying linear perturbations directly in the probability space violates fundamental simplex constraints (i.e., $p_i \in [0, 1]$ and $\sum p_i = 1$). Enforcing these constraints post hoc requires harsh clipping or truncation operations, which severely fragment gradient continuity and degrade optimization stability. By constructing targets and applying perturbations strictly in the unconstrained logit space ($z_i \in \mathbb{R}$), LFPO preserves smooth relative updates. The valid probability simplex is then naturally recovered via the Softmax operation without discarding crucial gradient information. This structural choice directly translates to empirical gains, elevating the HumanEval score from 73.9 (using probability clipping) to 75.6 (using our full logit-space formulation).

D Scope of the Theoretical Claim

Theorem 1 is intentionally pointwise. It states an identity in unconstrained logit space for a fixed input state and a fixed shared origin; it does not establish global equivalence between the CE objective and a continuous FM objective, nor does it prove monotonic policy improvement. The identity is useful because it gives a tractable local residual direction that avoids sequence/trajectory likelihood estimation.

Table 6: **Properties of different spaces for policy perturbation.** Probability-space perturbation uses clipping/truncation after a linear update, while LFPO constructs targets in logit space and maps them back to the simplex.

Property	Probability Space	Log-Prob. Space	Logit Space (Ours)
Domain	$p_i \in [0, 1]$	$\log p_i \leq 0$	$z_i \in \mathbb{R}$
Linear perturbation	Requires clipping	Shift-normalized	Unconstrained before Softmax
Simplex constraint	Broken before truncation	Recovered by normalization	Recovered by Softmax
Gradient continuity	Fragmented by clipping	Mostly smooth	Smooth relative update
HumanEval check	73.9 with clipping	–	75.6 with full LFPO

One limited implication is that the token-level CE logit gradient is bounded on the probability simplex. Since p_θ and $\boldsymbol{x}_1$ are probability vectors and $\boldsymbol{x}_1$ is one-hot,

$$\|\nabla_{\boldsymbol{z}} \mathcal{L}_{CE}\|_2 = \|p_\theta - \boldsymbol{x}_1\|_2 \leq \sqrt{2}. \tag{20}$$

This bounded token-level signal helps explain why LFPO has lower observed cross-seed variance than trajectory-likelihood estimators in our experiments. However, this observation should be interpreted as motivation for an empirical low-variance surrogate, not as a formal convergence guarantee for sparse-reward RLVR.

E Experimental Details and Hyperparameters

In this section, we provide the comprehensive hyperparameter configurations and implementation details for LFPO to ensure reproducibility. All experiments were conducted using the PyTorch framework on a cluster equipped with NVIDIA H20 (96GB) GPUs.

E.1 Training Configurations and Unified Hyperparameters

To ensure full reproducibility and demonstrate the robust out-of-the-box generalization of LFPO, we utilize a unified hyperparameter configuration across completely different tasks and model architectures (DiffuCoder for code generation and LLaDA 8B for mathematical reasoning). This design choice highlights that our framework does not rely on heavy, task-specific tuning to achieve strong performance.

We employ the AdamW optimizer with a linear learning rate warm-up for the first 10% of training steps, followed by a cosine decay schedule. To manage the memory constraints of dLLMs during stratified trajectory sampling, we implement the Block-wise Gradient Accumulation as described in Section 3.

Crucially, contrary to prior likelihood-based methods that often require brittle, domain-specific adjustments, LFPO maintains consistent core parameters. The contrastive coefficient β is fixed at 1.0, and we sample $N = 10$ trajectories per prompt with $K = 8$ stratified timesteps. The dynamic EMA decay rate α is scaled according to the training step to prevent early policy collapse. The complete, unified hyperparameter settings applied to all benchmarks are summarized in Table 7.

Table 7: Unified Hyperparameters for LFPO. This single configuration applies uniformly to all tasks and model architectures evaluated in our study.

Hyperparameter	Value	Hyperparameter	Value
Contrastive Coefficient (β)	1.0	Block-wise Partition Size	8
Learning Rate (Policy)	1×10^{-6}	Trajectories per Prompt (N)	10
EMA Decay Rate (α)	$\min(10^{-3} \times \text{step}, 0.5)$	Max Generation Length	512
Stratified Timesteps (K)	8	Max Diffusion Steps	2048
Optimizer	AdamW	Warm-up Ratio	0.1
Weight Decay	0.01	Gradient Clipping	1.0

E.2 Inference Protocol and Efficiency Baselines

To ensure a fair comparison of denoising-step counts, we strictly standardize the inference protocol for all diffusion-based methods. We use a deterministic Cosine noise schedule for the forward and reverse

processes. The schedule defines a target unmasking ratio at each step; there is no separately tuned fixed top-k value. Instead, each step fixes the most confident eligible positions needed to match the scheduled number of revealed tokens. The confidence-based early stopping mechanism is a key component of our step-efficiency analysis. Algorithm 2 gives the complete decoding protocol used for the step-count comparison.

- **Noise Schedule**: Cosine schedule with $T = 2048$ maximum diffusion steps. At step t, it gives a deterministic reveal ratio γ_t.
- **Confidence Metric**: For each position i, confidence is $c_i = \max_v \pi_\theta(x_i = v|\boldsymbol{x}_t)$.
- **Dynamic Parallel Update**: The schedule sets the target number of fixed positions $n_t = \lfloor L\gamma_t \rfloor$ for sequence length L. We fix the highest-confidence eligible positions until n_t positions are fixed; no fixed top-k hyperparameter is tuned.
- **Confidence Threshold** (τ): We set $\tau = 0.99$. If all positions exceed this threshold, decoding stops early.
- **Patience Parameter**: We use a patience of 3 steps. If the full decoded sequence prediction remains identical for 3 consecutive steps, inference terminates early.

The specific inference hyperparameters used to generate Table 3 and the denoising-step analysis in Section 4.2 are listed in Table 8.

Table 8: Inference hyperparameters and efficiency protocol. The parallel update size is dynamically determined by the cosine schedule, rather than by a separately tuned fixed top-k.

Parameter	Value
Max Diffusion Steps (T)	2048
Noise Schedule	Cosine
Confidence Metric	Max softmax probability
Dynamic Parallel Update	$n_t = \lfloor L\gamma_t \rfloor$ revealed positions
Confidence Threshold (τ)	0.99
Patience (Early Stopping)	3 identical sequence predictions
Sampling Temperature	1.0 (Greedy)

E.3 Hardware and Computational Cost

All models were trained using a cluster node equipped with 8 $\times$ NVIDIA H20 (96GB) GPUs. For DiffuCoder RL training, the average wall-clock time was approximately 18 hours per random seed. For the larger LLaDA 8B model, the training time was approximately 42 hours per seed. As noted in Section 4.2, while wall-clock time provides a practical reference, it fluctuates based on implementation details and specific hardware utilization. Therefore, our primary efficiency comparisons rigorously focus on the standardized reduction of denoising steps, which is an implementation-agnostic metric.

E.4 Statistical Significance and Optimization Stability

To ensure statistical rigor and avoid overstating small numerical gaps in maximum performance, we report the five-seed mean and standard deviation for the strongest baseline and our method in the main tables.

Because the main tables report aggregate seed statistics rather than paired per-example predictions, we conservatively estimate the 95% confidence intervals for the aggregate accuracy differences between LFPO and AGRPO using the independent-seed formula:

$$\Delta \pm t_{0.975,4}\sqrt{\frac{s_{\text{LFPO}}^2}{5} + \frac{s_{\text{AGRPO}}^2}{5}}, \tag{21}$$

where Δ is the mean difference and s denotes the reported standard deviation across seeds. As summarized in Table 9, while the confidence intervals for the mean difference encompass zero (indicating comparable

Algorithm 2 Standardized Diffusion Decoding Protocol

Input: prompt q, policy π_θ, max steps T, threshold τ, patience p.
Initialize response state $\boldsymbol{x}_T$ with masked answer positions.
$\hat{\boldsymbol{x}}_{\text{prev}} \leftarrow \varnothing$, $n_{\text{stale}} \leftarrow 0$.
for $t = T, T-1, \ldots, 1$ **do**
  Compute token probabilities for all positions and let $\hat{\boldsymbol{x}}_t$ be the full greedy decoded sequence.
  Compute confidences $c_i = \max_v \pi_\theta(x_i = v | \boldsymbol{x}_t)$.
  Use the cosine schedule to compute reveal ratio γ_t and target count $n_t = \lfloor L\gamma_t \rfloor$.
  Fix the highest-confidence eligible positions in parallel until n_t positions are fixed.
  if $\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{\text{prev}}$ **then**
    $n_{\text{stale}} \leftarrow n_{\text{stale}} + 1$.
  else
    $\hat{\boldsymbol{x}}_{\text{prev}} \leftarrow \hat{\boldsymbol{x}}_t$, $n_{\text{stale}} \leftarrow 0$.
  end if
  if $\min_i c_i \geq \tau$ or $n_{\text{stale}} \geq p$ **then**
    Stop early and return the current decoded sequence.
  end if
end for
Return the final decoded sequence.

peak accuracy rather than a strictly significant mean separation under this conservative estimate), the crucial takeaway lies in the variance terms. The standard deviation s_{LFPO} is consistently and significantly smaller than s_{AGRPO} across all tasks. This statistical reporting solidifies our core claim: LFPO delivers highly competitive performance bounds while offering a fundamentally more stable and reliable optimization trajectory.

Table 9: Approximate 95% confidence intervals for aggregate accuracy differences between LFPO and the strongest baseline (AGRPO). While the mean performance is strictly comparable, LFPO achieves this with significantly lower variance across seeds.

Domain	Mean Difference	Approx. 95% CI	Performance Interpretation
Code Avg.	+0.2	$[-1.1,\ 1.5]$	Comparable Mean, Superior Stability
Reasoning Avg.	+0.5	$[-0.8,\ 1.8]$	Comparable Mean, Superior Stability

F Comprehensive Sensitivity Analysis and Ablations

In this section, we provide a detailed sensitivity analysis to quantify the impact of algorithmic configurations on both training costs and denoising step efficiency. We also include additional generalization tests to further validate the robustness of LFPO.

F.1 Negative Controls and Ablation Coverage

Table 10 summarizes the additional controls used to separate algorithmic contributions from implementation feasibility. While some controls produce comparable accuracy numbers, others are included strictly as negative controls because they introduce discontinuous target constructions or render the default training configuration computationally infeasible.

F.2 Training Costs and Hyperparameter Ablations

To quantify the computational and memory sensitivity of LFPO, we systematically ablate core hyperparameters including the number of sampled trajectories (N), diffusion timesteps (K), the policy perturbation scale (β), and the EMA decay rate (α). The results on the HumanEval benchmark are summarized in Table 11.

Table 10: Ablation and negative control coverage. Numeric ablations are detailed in subsequent tables; feasibility controls are reported as implementation outcomes rather than quantitative accuracy comparisons.

Control	Purpose	Observed Outcome
No stratified sampling	Tests whether dense temporal coverage matters	Uniform timestep sampling yields higher gradient variance and lower GSM8K accuracy (Table 12)
No block-wise accumulation	Tests implementation feasibility of the full $B \times N \times K$ training tensor	Default configuration is not memory-feasible in our implementation; block-wise accumulation bounds peak VRAM to approximately 92.7 GB (Table 11)
Probability space perturbation	Tests whether targets can be constructed directly in probability space	Requires clipping or truncation at simplex boundaries, which fragments gradients; we therefore enforce logit space perturbation (Table 6)
Fixed EMA	Tests whether dynamic reference updates matter	Fixed $\alpha = 0.9$ drops HumanEval accuracy to 70.8% (Table 11)
Low N, low K, high β	Tests sensitivity to trajectories, timesteps, and perturbation scale	Reduces computational cost but lowers accuracy, or destabilizes training due to overly aggressive targets (Table 11)

Trajectories and Timesteps. The default configuration ($N = 10, K = 8$) achieves the highest accuracy in this ablation (75.6%). Reducing the number of trajectories to $N = 5$ decreases the GPU hours from 96 to 49 but incurs a 1.5% accuracy drop. Similarly, halving the timesteps to $K = 4$ reduces the training time to 52 hours but decreases accuracy to 74.2%. Notably, the Peak VRAM remains stable around 92.7 GB across all configurations, demonstrating that our memory-efficient block-wise optimization successfully bounds memory consumption regardless of sequence length or trajectory count.

Perturbation Scale and EMA. Increasing the perturbation scale to $\beta = 1.5$ slightly degrades performance (73.9%), indicating that overly aggressive contrastive targets can destabilize the policy. Furthermore, replacing our dynamic EMA update (α^*) with a fixed rate ($\alpha = 0.9$) causes a severe accuracy drop to 70.8%, highlighting the necessity of dynamically adjusting the reference model to prevent early policy collapse.

Table 11: Core Hyperparameters vs. Training Costs. Evaluated on HumanEval. The dynamic EMA rate is defined as $\alpha^* = \min(10^{-3} \times \text{step}, 0.5)$.

Config	β	K	N	α	Peak VRAM (GB)	GPU Hours	HumanEval Acc. (%)
Default	1.0	8	10	α^*	92.7	96	**75.6**
Low N	1.0	8	**5**	α^*	92.6	49	74.1
Low K	1.0	**4**	10	α^*	92.5	52	74.2
High β	**1.5**	8	10	α^*	92.7	96	73.9
Fixed EMA	1.0	8	10	0.9	92.7	96	70.8

Sampling Strategy. As discussed in Section 3, naive uniform sampling introduces high gradient variance. Table 12 indicates that our Stratified Trajectory Sampling substantially stabilizes the gradient, yielding a lower variance and boosting the GSM8K accuracy from 72.3% to 79.6%.

Table 12: Impact of Sampling Strategy on GSM8K (reported across 5 random seeds).

Sampling Method	Gradient Variance	Accuracy (%)
Uniform	High	72.3 ± 5.3
Stratified (Ours)	**Low**	79.6 ± 1.2

F.3 Inference Decoding Parameters

To explicitly quantify the sensitivity of LFPO during generation, we evaluate inference accuracy and average decoding steps under varying confidence thresholds (τ) and masking schedules. Table 13 reveals

several key insights regarding the performance and efficiency trade-offs.

First, the confidence threshold τ acts as a controllable parameter for efficiency. We find that $\tau = 0.99$ serves as an optimal empirical balance. It sacrifices merely 0.4% accuracy compared to the full-step setting without early stopping ($\tau = 1.00$, 76.9% to 76.5%), while substantially reducing the required denoising steps from 2048 to 218.3 on HumanEval. Lowering the threshold further yields fewer steps (173.2 steps at $\tau = 0.90$) but results in noticeable performance degradation.

Second, the Cosine schedule provides a superior balance of accuracy and efficiency compared to the Linear schedule. At the default threshold ($\tau = 0.99$), the Cosine schedule achieves higher accuracy with fewer steps (76.5% at 218.3 steps) than the Linear baseline (76.3% at 229.6 steps). This indicates that a non-linear unmasking rate better aligns with the confidence trajectory of the velocity field.

Table 13: Sensitivity Analysis of Inference Decoding Parameters on HumanEval.

Masking Schedule	Threshold (τ)	Accuracy (%)	Avg. Steps	Step Trade-off
Linear	0.99	76.3	229.6	Suboptimal step allocation.
Cosine (Ours)	**1.00 (No Early Stop)**	76.9	2048 (Max)	Upper bound accuracy, max cost.
Cosine (Ours)	**0.99 (Default)**	**76.5**	**218.3**	**Favorable accuracy and step balance.**
Cosine (Ours)	0.95	74.8	197.5	Fewer steps, minor accuracy drop.
Cosine (Ours)	0.90	73.5	173.2	Fewest steps, noticeable degradation.

F.4 Generalization to Other Base Models

To verify that the advantages of LFPO are not strictly bound to specific model architectures, we conducted generalization tests using the LLaDA framework applied to Dream Models. As shown in Table 14, LFPO consistently improves the performance of strong base models, achieving 81.9% (+1.2) on HumanEval and 36.1% (+1.5) on MATH, further demonstrating its universal applicability as a tractable RL reasoning framework.

Table 14: Generalization capabilities of LFPO applied to Dream Models.

Method	HumanEval (%)	MATH (%)
Base Model (Dream)	79.3	32.7
+ AGRPO	80.7	34.9
+ LFPO	**81.9 (+1.2)**	**36.1 (+1.5)**

F.5 Expanded Discussion on Robustness and Failure Modes

As introduced in the Limitations section (Section 5), while LFPO uses reference-anchored implicit targets to reduce sensitivity to single high-reward samples, this does not grant complete immunity to reward hacking. Sparse verifiable rewards may still be over-optimized, especially under severe distribution shifts or adversarial prompts. In practical deployments, we strongly recommend monitoring held-out tasks, formatting failures, and reward-specific shortcuts when utilizing LFPO-style training.

Furthermore, regarding the cold-start bottleneck, the positive target π^+ is effective only when the reference policy can sample at least a minimal set of reward-bearing trajectories. If the base dLLM has a near-zero success rate on a task, the contrastive attraction signal becomes too weak, and the training process may stall. Utilizing warm-start data, supervised bootstrapping, or curriculum sampling represents natural mitigations for this issue, and we leave a systematic study of these remedies to future work.

G Case Study: Visualizing Trajectory Straightening

To intuitively demonstrate the "trajectory straightening" effect discussed in Appendix F, we visualize the intermediate denoising states of the base diffusion model versus our LFPO framework. As shown in the case study below, the base model exhibits severe fluctuation in intermediate steps, temporarily predicting incorrect logical branches (e.g., calculating $r = 9$ or predicting the point is on the x-axis). In contrast, LFPO anchors to the correct terminal state (radius 3, angle $\pi/2$) extremely early in the diffusion process, effectively bypassing meandering paths and allowing for confidence-based early stopping.

LLaDA 8B, MATH dataset

```
Question: Convert the point (0,3) in rectangular coordinates to polar coordinates.
Enter your answer in the form (r, \theta).

True answer: \left( 3, \frac{\pi}{2} \right)
--------------------------------------------------------------------------------
BASE model (Intermediate Denoising Trajectory):
Step 500 : r = \sqrt{0^2 + [MASK]} = [MASK]. \theta = \arctan([MASK]/0) = [MASK]
Step 1000: r = \sqrt{0^2 + 3^2} = 9. The point lies on the x-axis, so \theta = 0.
Step 1500: r = 3. The point is on the y-axis, \theta = \pi. Answer: (3, \pi).
Step 2048: r = 3. The point is on the positive y-axis. \theta = \pi/2.
           Final Answer: \left( 3, \frac{\pi}{2} \right)
[Analysis: Severe epistemic fluctuation. Logic oscillates before the final step.]

LFPO Model (Intermediate Denoising Trajectory):
Step 500 : r = \sqrt{0^2 + 3^2} = 3. \theta = [MASK]. Answer: (3, \pi/2).
Step 1000: r = 3. The point is on the positive y-axis. \theta = \pi/2.
Step 1500: r = 3. The point is on the positive y-axis. \theta = \pi/2.
Step 2048: r = 3. The point is on the positive y-axis. \theta = \pi/2.
           Final Answer: \left( 3, \frac{\pi}{2} \right)
[Analysis: Terminal anchoring. The correct skeleton forms early, saving compute.]
```