LaDi-RL: Latent Diffusion Reasoning Prevents Diversity Collapse in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning has become a central paradigm for improving LLM reasoning, but most existing methods optimize policies over discrete token sequences. This creates a mismatch between the optimization space and the structure of reasoning: many important decisions are semantic, global, and trajectory-level rather than local token choices. Continuous latent-space RL offers a promising alternative, but the latent policy must model a complex, multi-modal distribution over valid reasoning trajectories. We therefore propose **La**tent **Di**ffusion Reasoning with **R**einforcement **L**earning **(LaDi-RL)**, where a diffusion model generates latent reasoning trajectories through iterative denoising, enabling structured exploration and expressive distribution modeling. This comes with a credit-assignment challenge: the policy acts in latent space, but rewards are observed only after the latent is decoded into text, so a naive rollout cannot tell whether an incorrect answer comes from a poor latent trajectory or from an imperfect textual realization. We address this with hierarchical latent-text rollouts, which decode several answers per latent and score the latent by their mean reward, separating the quality of the reasoning from the luck of its decoding. Empirically, LaDi-RL outperforms token-level RL by 10.5% on code generation and 6.5% on math reasoning in pass@1, and raises pass@k above its pre-RL starting point, which token-level RL fails to do.

1 Introduction

Reinforcement learning (RL) has become a dominant paradigm for improving the reasoning ability of large language models (LLMs), particularly by optimizing the generation of reasoning trajectories (Guo et al., 2025a; Yu et al., 2025b; Shao et al., 2024). However, this process can suffer from *diversity collapse* (Song et al., 2024; Dang et al., 2025; Yue et al., 2025; Zhao et al., 2025b; He et al., 2025): during training, the model increasingly concentrates probability mass on a narrow set of reasoning patterns while suppressing alternative ones (Song et al., 2024; Dang et al., 2025; Yue et al., 2025; Zhao et al., 2025b; He et al., 2025). This collapse presents a critical bottleneck for further RL optimization: when the model's sample distribution becomes more concentrated, the explored trajectories become less diverse and the resulting RL performance gains become limited. This phenomenon is visible in pass@k metrics (He et al., 2025; Cobbe et al., 2021; Yue et al., 2025; Chung et al., 2025; Chen et al., 2025a): while RL-tuned models often improve pass@1, they may underperform their pre-RL starting point at large k, indicating a loss of diverse correct trajectories, as shown by vanilla GRPO in Figure 1.

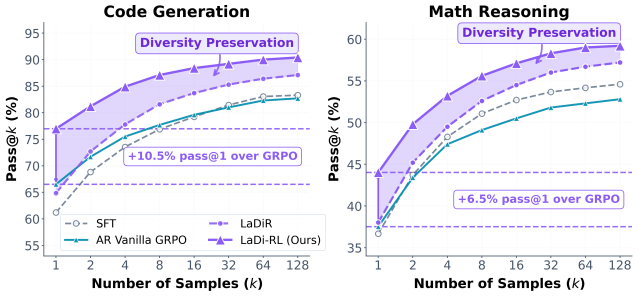

Figure 1: **Average pass@k over all tested benchmarks.** LaDi-RL preserves solution diversity and converts LaDiR's pass@k gains into strong pass@1 improvements, whereas vanilla GRPO exhibits diversity collapse at large k.

Recent work mitigates this problem by encouraging *semantic diversity* during training (Yao et al., 2025; Hu et al., 2025; Li et al., 2025d; Naik et al., 2023; Chen et al., 2025f; Walder & Karkhanis, 2025). These methods typically retain token-level exploration and add diversity-oriented rewards or regularizers. We argue that this creates a mismatch: *semantic diversity is a global, trajectory-level*

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

property, but the LLM policy explores through local token-level perturbations. As a result, the model can easily generate lexical variants of the same reasoning trajectory, such as replacing "therefore" with "so" or rephrasing an intermediate step, without discovering semantically different strategies such as switching from an algebraic derivation to a geometric argument. Token-level exploration is therefore an indirect and inefficient mechanism for optimizing semantic diversity.

To address this mismatch, we propose **La**tent **Di**ffusion Reasoning with **R**einforcement **L**earning **(LaDi-RL)**. LaDi-RL moves exploration from the token level to the trajectory level using latent reasoning, where an entire reasoning trajectory is encoded into continuous latent tokens (Kang et al., 2025). These latent tokens then condition the autoregressive generation of the final answer text, which are evaluated by downstream task rewards. In this latent space, perturbing the latent tokens changes the underlying reasoning trajectory as a whole, allowing the model to explore semantically different solution strategies rather than merely surface-form lexical variations.

Effective latent-space exploration, however, requires an expressive policy. Reasoning trajectories form a complex and multi-modal distribution: a problem may admit many valid solution strategies, each corresponding to a different region of latent space. Simple continuous policies may struggle to represent this structure or may average across modes, producing incoherent trajectories. We therefore parameterize the latent policy as a diffusion model over trajectory representations (Ho et al., 2020; Song & Ermon, 2019). Starting from noise, the policy can iteratively denoise toward a latent reasoning trajectory, injecting stochasticity throughout generation while converging towards the data distribution. This process supports structured exploration and naturally enables diversity-promoting guidance, such as repulsion during denoising (D'Angelo & Fortuin, 2021; Zilberstein et al., 2024).

Optimizing such a latent policy introduces a new credit-assignment challenge. The policy acts in latent space, but rewards are observed only after the latent trajectory is decoded into text and evaluated on the downstream task. A failed solution may therefore arise from a poor latent reasoning trajectory or from an imperfect textual realization of an otherwise useful latent. Naively assigning the reward of a single decoded completion to the latent trajectory entangles latent quality with decoder noise, yielding a noisy and potentially misleading learning signal.

We address this issue with *hierarchical latent-text rollouts*. Under a fixed rollout budget, LaDi-RL first samples a smaller set of latent reasoning trajectories and then decodes multiple answers from each latent. Their rewards are averaged to estimate the latent's decoder-marginalized utility, i.e., how useful the latent is across possible textual realizations. A good latent is thus not penalized for a single failed decoding, nor a weak latent rewarded for a lucky one, so the latent policy is credited for the quality of its reasoning rather than for the outcome of one decoding.

We evaluate LaDi-RL on code generation and mathematical reasoning benchmarks, where it consistently improves both accuracy and diversity compared to token-level RL baselines. It achieves absolute pass@1 improvements of +10.5% and +6.5% respectively while its pass@k stays above that of its pre-RL starting point. These results demonstrate that diffusion-based latent trajectory optimization, combined with decoder-marginalized reward estimation, provides an effective alternative to token-level RL for improving LLM reasoning without sacrificing solution diversity.

2 Preliminary

2.1 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a reinforcement learning algorithm that has been widely adopted for optimizing large generative models, including large language models (Yu et al., 2025b) and diffusion-based generative models (Liu et al., 2025b; Xue et al., 2025). Formally, given a group of rollout trajectories $\{\mathbf{o}_g\}_{g=1}^{G}$ sampled from the behavior policy $\pi_{\theta_{\text{old}}}$, where the trajectories share the same conditioning input (e.g., prompt or context), GRPO computes a normalized advantage for each trajectory directly from scalar rewards: $\hat{A}_g = \frac{R_g - \text{mean}(\{R_g\}_{g=1}^{G})}{\text{std}(\{R_g\}_{g=1}^{G}) + \varepsilon_{\text{adv}}}$ where $\varepsilon_{\text{adv}} > 0$ stabilizes normalization and std denotes the population standard deviation (zero for a singleton group). The same convention is used for all group advantages below. Here R_g denotes the outcome reward of trajectory $\mathbf{o}_g$. The GRPO objective is defined as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\{\mathbf{o}_g\}_{g=1}^{G} \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{g=1}^{G} \min\Big(r_g(\theta)\hat{A}_g, \text{clip}\big(r_g(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_g \Big) \right] - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\theta_{\text{ref}}})\,. \quad (1)$$

where $r_g(\theta) = \frac{\pi_\theta(\mathbf{o}_g)}{\pi_{\theta_{\text{old}}}(\mathbf{o}_g)}$ (for text, applied per token and averaged) and $\pi_{\theta_{\text{ref}}}$ is a fixed reference policy, ϵ is the clip threshold, and β is the KL weight.

GRPO for Flow Matching Models. Flow-GRPO (Liu et al., 2025b; Li et al., 2025b) applies GRPO to flow matching models by turning deterministic denoising steps into stochastic ones. We use its Coefficients-Preserving Sampling (CPS) variant (Wang & Yu, 2025). Here $t \in [0, 1]$ is the diffusion timestep, with t=0 the clean sample and t=1 pure noise. With step size $\Delta t > 0$, each step is the Gaussian transition

$$\begin{aligned} \pi_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t) &= \mathcal{N}\big(\mathbf{x}_{t-\Delta t};\ \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\ \sigma_t^2 \mathbf{I}\big)\,, \\ \boldsymbol{\mu}_\theta &= (1 - t')\,\hat{\mathbf{x}}_0 + t' \cos\big(\tfrac{\eta\pi}{2}\big)\,\hat{\mathbf{x}}_1, \qquad \sigma_t = t' \sin\big(\tfrac{\eta\pi}{2}\big), \end{aligned} \tag{2}$$

where $t' = t - \Delta t$, $\hat{\mathbf{x}}_0 = \mathbf{x}_t - t\, v_\theta(\mathbf{x}_t, t)$ and $\hat{\mathbf{x}}_1 = \mathbf{x}_t + (1 - t)\, v_\theta(\mathbf{x}_t, t)$ are the clean-sample and noise predictions of the learned flow field v_θ, and $\eta \in [0, 1]$ sets the stochasticity (η=0 recovers the deterministic ODE step). The Flow-GRPO-Fast variant (Liu et al., 2025a) we use applies this stochastic step only within a short window of denoising steps and samples deterministically elsewhere. At stochastic steps with $\sigma_t > 0$, this formulation gives a Gaussian density; deterministic steps are point masses and do not enter the likelihood-ratio loss. This enables integration with the GRPO objective in Eq. (1). Further details are provided in Appendix C.

2.2 Latent Diffusion Reasoner (LaDiR)

LaDiR (Kang et al., 2025) is a latent diffusion–based reasoning framework that compresses text CoTs into a compact continuous latent block, enabling efficient semantic-level reasoning. In this work, we adopt LaDiR as a *cold-start* initialization for subsequent reinforcement learning. LaDiR thus plays the role of the standard SFT stage (same CoT data, matched cost; Appendix E.2).

Architecture. LaDiR represents a reasoning process using a single continuous latent CoT block. Given a question Q, the LLM operates on the sequence $[\,Q,\ \texttt{<BOT>},\ z_1, \ldots, z_B,\ \texttt{<EOT>},\ y_1, \ldots, y_L\,]$, where <BOT> and <EOT> mark the beginning and end of the latent block $Z = \{z_1, \ldots, z_B\}$, and the answer $Y = (y_1, \ldots, y_L)$ is generated autoregressively conditioned on Q and Z. The latent tokens are produced by compressing a short reference CoT (typically fewer than 1k tokens) using a variational autoencoder (VAE), yielding a fixed-size latent representation (e.g., $B = 64$). Unlike the original LaDiR formulation, which applies blockwise diffusion over multiple latent segments, we employ a single sufficiently large latent block, which we find adequate for reconstructing short CoTs while simplifying the model.

Training Objective. LaDiR is trained using a joint objective that aligns latent reasoning with downstream text generation. Let Z^{enc} denote the latent tokens encoded from a reference CoT. We train a conditional latent generator using a flow-matching objective: $\mathcal{L}_{\text{FM}} = \mathbb{E}\left[\|v_\theta(Z_t, Q, t) - (\epsilon - Z^{\text{enc}})\|_2^2\right]$, where $Z_t = (1 - t)Z^{\text{enc}} + t\epsilon$, $t \sim \mathcal{U}[0, 1]$, and $\epsilon \sim \mathcal{N}(0, I)$ is independent of the encoded latent. Conditioned on the question Q and latent block Z, the LLM is simultaneously trained to generate the target answer Y using a standard autoregressive cross-entropy loss: $\mathcal{L}_{\text{CE}} = -\mathbb{E}_{(Q,Z,Y)} \sum_t \log p_\theta(y_t \mid y_{<t}, Q, Z)$. The final training objective is $\mathcal{L} = \lambda \mathcal{L}_{\text{FM}} + \mathcal{L}_{\text{CE}}$, where λ is loss weight for the latent part.

Inference. During inference, latent tokens are generated by iterative denoising conditioned on the input question and then the model generates the answer text given the denoised latent block.

3 Methodology

Building upon LaDiR, we propose *Latent Diffusion Reasoner with Reinforcement Learning (LaDi-RL)*, a framework that applies GRPO to latent diffusion policies, enabling structured and flexible exploration in continuous latent space with higher diversity. A complementary text policy then conditions on the denoised latent reasoning trajectories to generate the final answer text.

3.1 Hierarchical Rollout

LaDi-RL involves two heterogeneous policies operating at different levels of abstraction: a latent diffusion policy first samples a continuous latent reasoning trajectory, and a text decoder then realizes that latent trajectory into an answer. This factorization enables trajectory-level exploration, but it also introduces a sharper credit-assignment challenge than standard token-level LLM reasoning. The downstream reward is observed only after the latent trajectory has been decoded into text, so the reward of a single latent-text rollout conflates two distinct factors: whether the latent trajectory

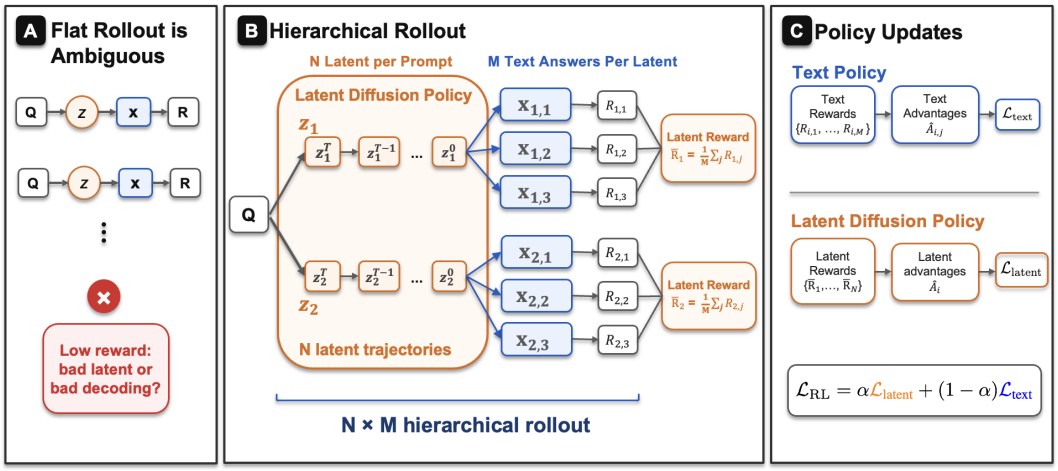

Figure 2: **Overview of LaDi-RL. (a)** Problem of standard flat rollout. **(b)** *Hierarchical $N\times M$ rollout:* the latent diffusion policy samples N latent blocks $\{z_n^0\}$, and the text policy generates M candidate answers $\{x_{n,m}\}$ per latent. **(c)** *Policy updates:* the latent policy is trained on the *average* reward of the M texts under each latent, and the text policy on *relative* differences among those M texts; both are optimized with GRPO.

encodes a useful reasoning mode, and whether the text decoder faithfully realizes that mode into a correct answer. As shown in Figure 2 (a), this makes credit assignment ambiguous: a low reward cannot tell whether the latent reasoning is flawed or whether a useful latent was merely decoded into a poor answer, making it difficult to optimize the latent and text policies separately.

To address this, we propose a *hierarchical $N\times M$ rollout* (Figure 2 (b)). Given a query Q, a single trajectory is represented as $\mathbf{o}=\{\mathbf{z}^K,\dots,\mathbf{z}^0,x^1,\dots,x^L\}$, where $\mathbf{z}^K,\dots,\mathbf{z}^0$ are the states of a single latent block along K denoising steps ($\mathbf{z}^K$ is noise and $\mathbf{z}^0$ is the denoised latent CoT), and $x^1,\dots,x^L$ are L text tokens generated autoregressively conditioned on $\mathbf{z}^0$. For each query we first sample N latent diffusion trajectories $\{\mathbf{z}_n\}_{n=1}^N$, and then sample M text responses $\{\mathbf{x}_{(n,m)}\}_{m=1}^M$ conditioned on each latent. Rewards are computed at the final answer text. This hierarchy disentangles the two sources of stochasticity: averaging the M rewards under each latent isolates the *quality of the latent reasoning*, while their relative differences isolate *which generated text works best for that reasoning*—giving the two policies clean, complementary signals from the same rollouts.

For efficiency, we use a small number of denoising steps K (e.g., K=10) and disable classifier-free guidance (Ho & Salimans, 2022) during rollout. The $N\times M$ hierarchy is used only for RL training; at test time LaDi-RL samples one latent and one answer (N=M=1).

3.2 Latent Diffusion Policy

Following Flow-GRPO-Fast (Liu et al., 2025b;a), we treat each stochastic CPS step (Eq. (2)) as a policy action. Each guided transition ratio is weighted by the advantage of that particle's decoded answers:

$$\mathcal{L}_{\text{latent}}^{\text{clip}(\epsilon_z)}=-\frac{1}{N}\sum_{n=1}^{N}\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\min\Big(r_{n,t}(\phi)\hat{A}_n,\text{clip}(r_{n,t}(\phi),1-\epsilon_z,1+\epsilon_z)\hat{A}_n\Big),\tag{3}$$

where, as in Flow-GRPO, $r_{n,t}(\phi)=\exp(\ell_\phi-\ell_{\phi_{\text{old}}})$ is the dimension-averaged likelihood ratio of the guided training policy $q_\phi(\cdot\mid\mathbf{Z}^t,Q)=\mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{z}_n^t,\tau_t,Q)+\Delta_{n,t},\sigma_t^2 I)$ with $\Delta_{n,t}=\gamma_t F_n(\mathbf{Z}^t)$, $\ell_\phi=-\frac{1}{D}\|\mathbf{z}_n^{t-1}-\boldsymbol{\mu}_\phi(\mathbf{z}_n^t,\tau_t,Q)-\Delta_{n,t}\|^2$, and stored group state $\mathbf{Z}^t=(\mathbf{z}_1^t,\dots,\mathbf{z}_N^t)$ (Appendix C.7). Following DDPO (Black et al., 2023), the same guidance shift enters rollout sampling and both likelihoods, so the importance ratio is computed under the policy that generated the samples. The group state, sampled transition, shift, and old log-probability are held fixed during the update. Denoising step t corresponds to flow time τ_t, with $\tau_K=1>\cdots>\tau_0=0$ and $\Delta t=\tau_t-\tau_{t-1}$. $\mathcal{T}$ is the window of stochastic steps that are optimized; following Flow-GRPO-Fast, $\mathcal{T}=\{K-s,\,K-s-1\}$ with s drawn uniformly from $\{0,1,2,3\}$, i.e., two consecutive steps among the first five sampling steps, and all other steps are deterministic (Table 6); and the corresponding group-relative advantage $\hat{A}_n$ is computed by standardizing the N mean rewards $\{\bar{R}_n\}_{n=1}^N$ across the latent blocks for the same query Q (with ε_{adv} as in Section 2), where each $\bar{R}_n$ is calculated by averaging over the

M text answers $\mathbf{x}_{(n,m)}$ conditioned on the same n-th latent block: $\bar{R}_n = \frac{1}{M}\sum_{m=1}^{M} R(\mathbf{x}_{(n,m)})$, with outcome reward R.

Diversity Guidance To further improve diversity during the rollout process, inspired by kernel-density estimation, repulsive forces, and particle guidance (D'Angelo & Fortuin, 2021; Zilberstein et al., 2024; Corso et al., 2024), we introduce a repulsion-based guidance mechanism (see Figure 3). At each denoising step t, given the group of N latent trajectories $\{\mathbf{z}_n^t\}_{n=1}^N$ sampled for a query Q, we first compute a bandwidth parameter: $h = \text{median}_{n<n'}\|\mathbf{z}_n^t - \mathbf{z}_{n'}^t\|_2$ (we write h to avoid a clash with the noise scale σ_t). For $N = 1$, the interaction sum is empty and the force is zero; for $N \geq 2$, the formula below assumes $h > 0$. The repulsion force on a latent block $\mathbf{z}_n^t$ is defined by the interaction force $\mathbf{F}(\mathbf{z}_n^t)$:

$$\mathbf{F}(\mathbf{z}_n^t) = \sum_{n' \neq n} 2\Big(1 - \tfrac{d_{nn'}^2}{h^2}\Big)\exp\Big(-\tfrac{d_{nn'}^2}{h^2}\Big)(\mathbf{z}_n^t - \mathbf{z}_{n'}^t) \tag{4}$$

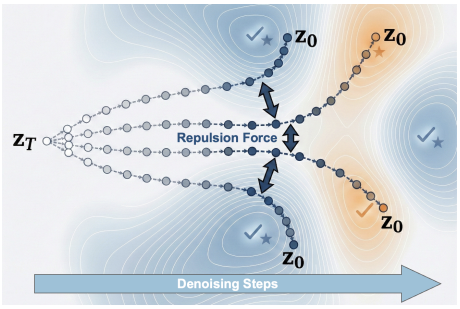

Figure 3: **Latent diffusion-based exploration with diversity guidance.** Multi-step denoising maps a noise to diverse solutions, while a repulsive force pushes trajectories apart latent space to encourage exploration.

where $d_{nn'}^2 = \|\mathbf{z}_n^t - \mathbf{z}_{n'}^t\|_2^2$. Intuitively, since the bandwidth h is the median pairwise distance, pairs closer than h are pushed apart and farther pairs are pulled slightly closer, which spreads the group evenly and keeps outliers from drifting away. The repulsion is applied with higher magnitude at early denoising steps and progressively reduced as $t \to 0$. Specifically, we define a time-dependent scale $\gamma_t = \gamma_{\max}\frac{t}{K}$, where K is the number of denoising steps. The guided update is:

$$\hat{\mathbf{z}}_n^{t-1} = f_\phi(\mathbf{z}_n^t, t, Q) + \gamma_t \mathbf{F}(\mathbf{z}_n^t), \tag{5}$$

where $f_\phi(\mathbf{z}_n^t, t, Q)$ denotes one denoising step of the base latent policy, i.e., a sample from $p_\phi(\cdot \mid \mathbf{z}_n^t, Q)$ for $t \in \mathcal{T}$ and the deterministic (η=0) step otherwise. The guided sample is stored as $\mathbf{z}_n^{t-1} := \hat{\mathbf{z}}_n^{t-1}$ together with its guidance shift and old score $\ell_{\phi_{\text{old}}}$, and used in Eq. (3). This repulsion-based guidance serves as a form of local geometric regularization, explicitly separating nearby latent trajectories, while the latent diffusion process provides global multi-modal support. Together, they reshape the geometry of exploration, preserving diversity during RL training without relying on policy entropy. Guidance is applied only when collecting training rollouts; for evaluation (including pass@k) and deployment we set γ_t=0 and sample latents independently. See Appendix C.8 and C.7 for analysis.

3.3 Text Policy and Joint Optimization

The text policy p_θ acts as a complementary module, grounding the reasoning generated by the latent diffusion policy into specific text answers. We employ a local standard GRPO loss for it:

$$\mathcal{L}_{\text{text}}^{\text{clip}(\epsilon_x^l,\epsilon_x^h)} = -\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\frac{1}{L_{n,m}}\sum_{j=1}^{L_{n,m}}\min\Big(r_{n,m,j}(\theta)\hat{A}_{n,m}, \text{clip}(r_{n,m,j}(\theta), 1-\epsilon_x^l, 1+\epsilon_x^h)\hat{A}_{n,m}\Big), \tag{6}$$

where $r_{n,m,j}(\theta) = \frac{p_\theta(x_{n,m,j}|x_{n,m,<j},Q,\mathbf{z}_n^0)}{p_{\theta_{\text{old}}}(x_{n,m,j}|x_{n,m,<j},Q,\mathbf{z}_n^0)}$ is the importance sampling ratio for the j-th token of an answer with $L_{n,m}$ tokens (as in GRPO for LLMs, token terms are averaged within each answer), and $\hat{A}_{n,m} = \frac{R(\mathbf{x}_{(n,m)}) - \text{mean}(\{R(\mathbf{x}_{(n,i)})\}_{i=1}^M)}{\text{std}(\{R(\mathbf{x}_{(n,i)})\}_{i=1}^M) + \varepsilon_{\text{adv}}}$ represents the local group advantage calculated strictly within the M text samples generated conditioned on the same latent block $\mathbf{z}_n$ (again with a small constant added to the standard deviation). This design decouples text generation from latent-space exploration: because advantages are compared only among answers to the same latent, the text policy is not penalized for poor latent trajectories and instead learns to realize each given reasoning trajectory as well as possible. The two policies are optimized jointly with the objective

$$\mathcal{L}_{\text{RL}} = \alpha\mathcal{L}_{\text{latent}}^{\text{clip}(\epsilon_z)} + (1-\alpha)\mathcal{L}_{\text{text}}^{\text{clip}(\epsilon_x^l,\epsilon_x^h)}, \tag{7}$$

where $\alpha \in [0, 1]$ balances the latent and text objectives, so that the text policy p_θ adapts to the reasoning explored by the latent diffusion policy p_ϕ (Figure 2). Both policies share one backbone (ϕ and θ denote the same parameters, used for latent denoising and for text decoding), so LaDi-RL has the same number of parameters as token-level GRPO. All losses in this section are minimized.

4 EXPERIMENT

We evaluate our method on two representative reasoning domains: code generation and math reasoning. We compare against strong baselines under controlled settings and report standard pass@1 and pass@k metrics across benchmarks. See more experimental details in Appendix D.

4.1 EXPERIMENTAL SETUP

Dataset For code generation, we initialize the model with SFT on the Ling-Coder dataset (Codefuse et al., 2025), which contains 1.4M Python-only samples. RL training is then performed on 24k filtered problems drawn from AceCoder (Zeng et al., 2025a) and KodCode (Xu et al., 2025). We evaluate model performance on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), their extended variants HumanEval+ and MBPP+ (Liu et al., 2023), as well as LiveCodeBench v6 (LCB V6) (Jain et al., 2024). For math reasoning, we use the R1-distill dataset (Madhusudhan et al., 2025) for SFT and the DeepScaleR-Preview-Dataset (Luo et al., 2025) for RL training, which contains approximately 40K unique problem–answer pairs. We evaluate on AIME 2024 (Veeraboina, 2023), AIME 2025 (Zhang & Math-AI Team, 2025), AMC 2023, MATH-500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024).

Baselines We compare against strong autoregressive, diffusion-based, and reinforcement learning baselines. For code generation, we report open-source AR coding models (Qwen2.5-Coder (Hui et al., 2024), OpenCoder (Huang et al., 2025b), rStar-Coder (Liu et al., 2025c), OlympicCoder (Hugging Face, 2025), Seed-Coder (Seed et al., 2025)) and reasoning-enhanced models (OpenThinker (Guha et al., 2025), DeepSeek-R1-Distill (Guo et al., 2025a)), discrete diffusion language models (Dream (Ye et al., 2025), Dream-Coder (Xie et al., 2025b), LLaDA (Nie et al., 2025), Diffu-Coder (Gong et al., 2025)), and the looped latent reasoning model Ouro (Zhu et al., 2025c). To validate our training method, we compare with reasoning methods (Soft Thinking (Zhang et al., 2025d), Standard SFT, TaH+ (Fu et al., 2025)); RL baselines (AR vanilla GRPO (Shao et al., 2024), GRPO with entropy advantages (Cheng et al., 2025), HybridGRPO (Sane, 2025), Soft Token (Butt et al., 2025), SofT-GRPO (Zheng et al., 2025b), Multiplex Thinking (Tang et al., 2026)); and methods that directly target diversity collapse: diversity-promoting RL (DSDR (Wan et al., 2026), DIVER (Hu et al., 2025), DARLING (Li et al., 2025d), Rewarding the Unlikely (He et al., 2025)), diversity-preserving SFT (GEM (Li et al., 2025f)), pass@k optimization (PKPO (Walder & Karkhanis, 2025)), and latent-exploration RL (ReLaX (Zhang et al., 2025c)). For math reasoning, we compare with vanilla GRPO, GRPO with entropy advantages, Soft Thinking (Zhang et al., 2025d), and Multiplex Thinking (Tang et al., 2026). We also include LaVAE, which replaces the flow-matching loss with an ℓ_2 loss on the VAE latents, and its RL variant LaVAE-RL. For fairness, all RL baselines, including vanilla GRPO, start from the SFT checkpoint trained on the same data; LaDi-RL starts from LaDiR, its counterpart of this SFT stage; each RL method is compared with its own starting point.

Implementation Details For code generation, we use *Qwen3-8B-Base* (Yang et al., 2025a) as the base model, while for math reasoning we adopt *DeepSeek-R1-Distill-Qwen-7B* (Guo et al., 2025a). To mitigate performance variance inherent in single-run evaluations, we sample 16 solutions per problem and report the average pass@1 accuracy for all benchmarks. For pass@k, we draw 128 independent samples per problem. For both code and math, every RL method uses the same budget of 16 rollouts per prompt; LaDi-RL arranges them as $N \times M = 4 \times 4$, i.e., 4 latent trajectories with 4 sampled answers each. See Appendix D for more details.

5 RESULTS

In this section, we present the main results, analysis, and ablation studies on the two domains. See efficiency analysis and more ablation studies in Appendix E, and qualitative results in Appendix F.

5.1 MAIN RESULTS

pass@1 improvements over baselines. Tables 1 and 2 show that **LaDi-RL** consistently outperforms prior *latent reasoning* and *reinforcement learning* methods across both code generation and math reasoning tasks. Compared to latent reasoning approaches such as Soft Thinking, LaDi-RL

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Average
Base (DS-R1-Distill-Qwen-7B)	15.2	16.7	41.3	72.4	32.6	34.8	35.50
Soft Thinking‡	20.3	19.1	47.9	76.5	37.2	40.6	40.27
Multiplex Thinking‡	20.6	19.7	50.7	78.0	38.6	41.7	41.55
Standard SFT	16.9	17.1	43.0	73.2	34.2	35.7	36.68
↪ AR Vanilla GRPO	17.9$_{+1.0}$	17.3$_{+0.2}$	44.8$_{+1.8}$	73.6$_{+0.4}$	35.4$_{+1.2}$	36.2$_{+0.5}$	37.53$_{+0.85}$
↪ GRPO+Entropy Adv.	17.9$_{+1.0}$	18.8$_{+1.7}$	43.1$_{+0.1}$	76.8$_{+3.6}$	37.8$_{+3.6}$	38.4$_{+2.7}$	38.80$_{+2.12}$
LaVAE	6.5	7.7	20.9	47.5	17.6	18.2	19.73
↪ LaVAE-RL	7.3$_{+0.8}$	8.5$_{+0.8}$	26.6$_{+5.7}$	53.4$_{+5.9}$	20.9$_{+3.3}$	26.8$_{+8.6}$	23.92$_{+4.19}$
LaDiR	17.9	18.3	46.4	75.8	35.6	34.1	38.02
↪ **LaDi-RL**	**22.5**$_{+4.6}$	**21.0**$_{+2.7}$	**53.1**$_{+6.7}$	**82.0**$_{+6.2}$	**41.2**$_{+5.6}$	**44.3**$_{+10.2}$	**44.02**$_{+6.00}$

Table 1: Math reasoning results (pass@1). ‡Reported by Tang et al. (2026). ↪ marks RL applied to the checkpoint above; subscripts give the gain from RL. Blue : latent reasoning methods.

Model / Method	Base Model	MBPP	MBPP+	HumanEval	HumanEval+	LCB V6	Avg.
Open-Sourced Models							
Autoregressive Coding Models							
Qwen 2.5 Coder*	Qwen2.5-7B	75.9	62.9	66.5	60.4	26.8	58.50
OpenCoder*	from scratch (8B)	79.9	70.4	66.5	63.4	29.6	61.96
rStar-Coder*	Qwen2.5-7B	87.9	74.0	95.9	90.8	53.5	80.42
OlympicCoder*	Qwen2.5-7B	80.0	66.4	82.1	76.9	37.3	68.54
DeepSeek-R1-Distill*	Qwen2.5-7B	78.4	66.7	89.6	83.7	34.2	70.52
OpenThinker2*	Qwen2.5-7B	86.9	73.9	92.7	87.8	29.2	74.10
Seed-Coder*	from scratch	82.0	69.0	77.4	68.3	28.4	65.02
Diffusion Language Models							
Dream*	Qwen2.5-7B	68.7	57.4	56.7	50.0	18.6	50.28
LLaDA*	from scratch (8B)	50.1	42.1	35.4	30.5	12.4	34.10
Diffu-Coder*	Qwen2.5-Coder-7B	75.1	61.9	72.0	65.2	24.5	59.74
Dream-Coder*	Qwen2.5-Coder-7B	75.9	61.6	66.5	60.4	21.4	57.16
d1*	LLaDA+RL	39.0	–	45.5	–	–	–
Looped Latent Reasoning Models							
Ouro*	from scratch (2.6B)	80.4	66.6	78.2	70.7	38.7	66.92
Method Comparison							
Non-RL Methods							
Base Model	Qwen3-8B-Base	60.6	53.7	78.0	68.3	33.0	58.72
Standard SFT	Qwen3-8B-Base	63.2	52.6	84.8	69.5	35.9	61.20
GEM	Qwen3-8B-Base	69.6	61.9	87.8	75.6	41.0	67.18
Soft Thinking	Qwen3-8B-Base	64.3	53.2	84.8	71.3	35.2	61.76
TaH+	Qwen3-8B-Base	65.6	56.6	86.0	74.4	37.0	63.92
LaVAE	Qwen3-8B-Base	42.1	30.2	47.6	32.9	9.9	32.54
LaDiR	Qwen3-8B-Base	66.7	59.5	87.2	73.2	37.7	64.86
RL Training Methods (from Standard SFT)							
AR Vanilla GRPO	Qwen3-8B-Base	72.2$_{+9.0}$	61.6$_{+9.0}$	82.3$_{-2.5}$	74.4$_{+4.9}$	42.1$_{+6.2}$	66.52$_{+5.32}$
GRPO w/ Entropy Adv.	Qwen3-8B-Base	73.8$_{+10.6}$	63.0$_{+10.4}$	84.1$_{-0.7}$	78.0$_{+8.5}$	43.8$_{+7.9}$	68.54$_{+7.34}$
DSDR	Qwen3-8B-Base	77.0$_{+13.8}$	68.5$_{+15.9}$	86.6$_{+1.8}$	78.7$_{+9.2}$	44.7$_{+8.8}$	71.10$_{+9.90}$
DIVER	Qwen3-8B-Base	77.8$_{+14.6}$	69.3$_{+16.7}$	87.2$_{+2.4}$	79.3$_{+9.8}$	45.4$_{+9.5}$	71.80$_{+10.60}$
DARLING	Qwen3-8B-Base	75.1$_{+11.9}$	66.9$_{+14.3}$	86.0$_{+1.2}$	77.4$_{+7.9}$	43.2$_{+7.3}$	69.72$_{+8.52}$
Rewarding the Unlikely	Qwen3-8B-Base	77.2$_{+14.0}$	68.8$_{+16.2}$	86.6$_{+1.8}$	78.7$_{+9.2}$	44.9$_{+9.0}$	71.24$_{+10.04}$
PKPO	Qwen3-8B-Base	75.7$_{+12.5}$	67.5$_{+14.9}$	86.0$_{+1.2}$	77.4$_{+7.9}$	43.8$_{+7.9}$	70.08$_{+8.88}$
HybridGRPO	Qwen3-8B-Base	73.0$_{+9.8}$	61.9$_{+9.3}$	82.9$_{-1.9}$	76.8$_{+7.3}$	42.7$_{+6.8}$	67.46$_{+6.26}$
Soft Token	Qwen3-8B-Base	70.1$_{+6.9}$	60.6$_{+8.0}$	81.7$_{-3.1}$	70.7$_{+1.2}$	36.6$_{+0.7}$	63.94$_{+2.74}$
SofT-GRPO	Qwen3-8B-Base	72.8$_{+9.6}$	60.3$_{+7.7}$	84.1$_{-0.7}$	72.0$_{+2.5}$	37.9$_{+2.0}$	65.42$_{+4.22}$
ReLaX	Qwen3-8B-Base	78.0$_{+14.8}$	69.6$_{+17.0}$	87.2$_{+2.4}$	79.3$_{+9.8}$	45.6$_{+9.7}$	71.94$_{+10.74}$
RL Training Methods (from latent SFT: LaVAE, LaDiR)							
LaVAE-RL	Qwen3-8B-Base	45.5$_{+3.4}$	31.0$_{+0.8}$	53.0$_{+5.4}$	34.1$_{+1.2}$	11.0$_{+1.1}$	34.92$_{+2.38}$
LaDi-RL	Qwen3-8B-Base	**84.1**$_{+17.4}$	**75.1**$_{+15.6}$	**90.2**$_{+3.0}$	**82.9**$_{+9.7}$	**52.6**$_{+14.9}$	**76.98**$_{+12.12}$

Table 2: pass@1 results on code generation benchmarks. Subscripts on RL rows give the gain from RL over the method's starting checkpoint (green: gains; red: drops). Blue-shaded rows are latent reasoning methods; unshaded rows are token-space methods. *Results are from the original papers.

achieves average improvements of $+3.8\%$ on math reasoning and up to $+5.5\%$ on individual datasets (e.g., MATH500). Comparing to RL baselines, LaDi-RL improves over the latent RL method Multiplex Thinking (Tang et al., 2026), by $+2.5\%$ on math reasoning and GRPO with entropy advantage by $+8.4\%$ on code generation, while yielding even larger margins over vanilla GRPO ($+6.5\%$ on math and $+10.5\%$ on code). Against methods that directly target diversity collapse, LaDi-RL improves average code pass@1 by $+5.2\%$ over the strongest diversity-oriented RL method (DIVER) and by $+5.0\%$ over the strongest latent RL method (ReLaX). Thus optimizing latent diffusion trajectories gives stronger gains than other latent or discrete RL.

We further compare LaDi-RL with well-trained open-source coding models, which use different backbones and training data. Compared with *diffusion language models*, LaDi-RL improves average pass@1 by about $+17\%$, with gains of around $+18\%$ on HumanEval and $+28\%$ on LiveCodeBench-V6. Compared to *looped latent reasoning models*, it improves average pass@1 by $+10.1\%$. These results suggest that LaDi-RL offers a general framework for LLM reasoning.

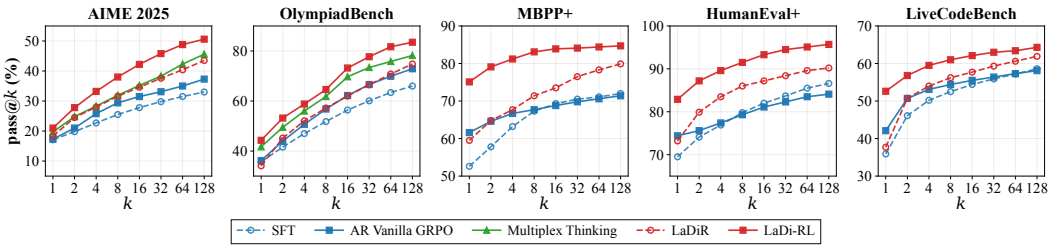

Figure 4: pass@k performance on code generation and math reasoning across different k.

Latent diffusion exploration mitigates diversity collapse in pass@k. As shown in Figure 4, our results confirm the diversity collapse of AR vanilla GRPO (Yue et al., 2025) in discrete token space: while GRPO improves pass@1, its pass@128 falls below its SFT starting point on the three displayed code benchmarks, indicating reduced solution coverage there. While recent continuous latent reasoning methods such as *Multiplex Thinking* (Tang et al., 2026) partially alleviate this issue, our LaDi-RL exceeds Standard SFT at large k on all five displayed benchmarks, achieving absolute improvements over SFT of +17.6 (AIME 2025), +17.5 (OlympiadBench), +12.7 (MBPP+), +9.1 (HumanEval+), and +5.7 (LiveCodeBench) at k=128. These gains go beyond the initialization: LaDi-RL improves over its LaDiR starting point by $+12.1$ pass@1 on code, and its pass@k curve stays above LaDiR at every k on all five benchmarks, so RL raises accuracy without losing coverage.

5.2 Ablation Studies

Token-level vs. Trajectory-level Exploration. We compare token-level with latent trajectory exploration. Besides vanilla GRPO, we include the strongest token-level alternatives from our full comparison (Appendix E.4): the best diversity-oriented method, PKPO, which directly optimizes pass@k, and the best latent-exploration method, ReLaX. We report *semantic diversity* (Sem.) as the average fraction of sampled solutions judged by GPT-5 to represent distinct solution strategies (Appendix E.5). As shown in Table 3, vanilla GRPO has the lowest semantic diversity (0.18), and from their starting points GRPO lowers Sem. and pass@128, while LaDi-RL raises both. Lifting exploration into latent trajectory space finds more semantically distinct trajectories (Sem. 0.52) and improves over GRPO by $+10.5$ pass@1 and $+7.7$ pass@128. LaDi-RL also outperforms the strongest token-level alternatives, by $+5.1$ pass@1 over ReLaX and by $+2.7$ pass@128 and $+0.07$ Sem. over PKPO. Exploring whole reasoning trajectories thus finds distinct strategies more effectively than adding diversity objectives or latent regularization to token-level sampling, even when only correct solutions are counted (Appendix E.5).

Diffusion vs. VAE Policy in Latent Space. To test whether the diffusion policy itself matters, we compare LaDi-RL with LaVAE-RL, whose only difference is that the multi-step diffusion sampler is replaced by a single-step VAE (ℓ_2) predictor over the same latent tokens (blue rows in Table 3). Diffusion yields both higher semantic diversity (Sem. 0.52 vs. 0.40) and substantially stronger performance ($+42.1$ pass@1 and $+45.8$ pass@128), and most of LaVAE-RL's diversity comes from diverse failures: its Sem. drops to 0.06 on correct solutions only (LaDi-RL: 0.36). The gap appears before RL: one-step ℓ_2 prediction averages distinct reasoning modes, so LaVAE already falls below the base model after SFT (32.5 vs. 58.7), and RL cannot recover them. Multi-step denoising, not the latent space alone, makes latent exploration work.

Latent exploration is the primary driver; text policy is complementary. We decompose LaDi-RL's policy into latent-only, text-only, and joint variants in Table 4. First, latent-only optimization (uncoupled, 70.3 avg) substantially outperforms text-only optimization (uncoupled, 67.3 avg), confirming that trajectory-level exploration carries the bulk of the gain in LaDi-RL. With tied weights, the latent-only gradient also shifts text generation with no text-level signal to correct it, so training is unstable and collapses (46.2); uncoupling keeps text generation fixed in a frozen copy. Second, jointly training both policies (75.6 avg) further improves over the latent-only setting by $+5.3$ avg, showing that the text policy complements latent exploration.

Method	p@1	p@128	Sem.
Base (ref.)	58.7	83.3	0.29
Standard SFT	61.2	83.3	0.27
LaDiR	64.9	87.1	0.39
Token-level exploration			
Vanilla GRPO (discrete)	66.5	82.7	0.18
PKPO (best pass@k)	70.1	87.7	0.45
ReLaX (best latent RL)	71.9	86.3	0.38
Trajectory-level exploration			
LaVAE-RL (VAE)	34.9	44.6	0.40
LaDi-RL (Diffusion)	**77.0**	**90.4**	**0.52**

Table 3: **Exploration-space ablation** on code (avg). Sem.: semantic difference in 16 samples

Method	MBPP	HE	LCB	Avg
LaDiR	66.7	87.2	37.7	63.9
Text Only (tied)	68.2	85.9	43.1	65.7
Text Only (uncoupled)	70.3	86.8	44.8	67.3
Latent Only (tied)	54.3	62.6	21.6	46.2
Latent Only (uncoupled)	73.8	89.8	47.2	70.3
Latent + Text	**84.1**	**90.2**	**52.6**	**75.6**
w/o div. guidance	80.1	88.2	49.4	72.6

Table 4: **Policy composition and diversity guidance** ablation (pass@1) on code (Avg. of 3). HE: HumanEval; LCB: LCB V6.

	4 rollouts per prompt			*16 rollouts per prompt*					
$N\times M$	1×4	2×2	4×1	1×16	2×8	4×4	4×4†	8×2	16×1
LaDi-RL	59.9	**65.5**	41.7	66.5	70.1	**77.0**	42.9	72.4	46.2

Table 5: Hierarchical-rollout ablation on code (five-benchmark avg pass@1). †: random uses one randomly selected answer reward per latent instead of the mean of four.

Diversity guidance. In Table 4, removing diversity guidance from the best *Latent + Text* setting lowers pass@1 by 4.0, 2.0, and 3.2 points. Figure 5 (Appendix) shows the same effect: without guidance, pass@1 improves more slowly and levels off lower on both tasks.

Hierarchical Rollout. We vary N and M under two fixed rollout budgets to study allocation between latent exploration and answer sampling (Table 5). With $M = 1$, the within-latent text advantage is zero, removing the direct text-loss signal; pass@1 is 41.7 for 4×1 and 46.2 for 16×1. With $N = 1$, the latent advantage is zero, leaving the text loss to update the shared backbone (59.9 for 1×4 and 66.5 for 1×16). Balanced splits perform best at both budgets, supporting allocations that provide direct learning signals at both levels. To isolate the credit estimator, we keep $N{=}M{=}4$ and both losses but replace each latent's mean reward with the reward of one randomly chosen answer. Mean aggregation outperforms random (77.0 vs. 42.9), supporting decoder averaging for latent credit assignment (Appendix E.3).

6 Related Works

RL for LLMs The advancement of Large Reasoning Models has been significantly propelled by RL paradigms (Guo et al., 2025a; OpenAI, 2024; Zhang et al., 2025b). While foundational algorithms like PPO (Schulman et al., 2017) and DPO (Rafailov et al., 2023) laid the groundwork for alignment, recent efforts have focused on pure RL methods such as GRPO (Guo et al., 2025a) and its variants (Lin et al., 2025; Zhang et al., 2025a; Li et al., 2025a), which elicit emergent reasoning patterns like self-reflection and verification (DeepSeek-AI, 2024). However, a persistent challenge in scaling RLVR is the phenomenon of *diversity collapse*, where the policy rapidly loses diversity and converges to suboptimal local minima (Cui et al., 2025; Hao et al., 2025; Jin et al., 2025b; Yue et al., 2025). This exploration-exploitation imbalance (Chen et al., 2025b; Huang et al., 2025a) has motivated various intervention strategies, including diversity-aware objectives (Yao et al., 2025; He et al., 2025; Walder & Karkhanis, 2025; Tang et al., 2025b; Yu et al., 2025b; Gai et al., 2025; Chen et al., 2025f), uncertainty-aware exploration (Xie et al., 2025a), and entropy control (Hao et al., 2025; Park et al., 2025; Cheng et al., 2025; Agarwal et al., 2025; Li et al., 2025c; Wang et al., 2025c; Zheng et al., 2025a; Yang et al., 2025c). These techniques remain limited to local stochastic control in discrete space, whereas LaDi-RL instead performs exploration over entire reasoning trajectories in a continuous latent diffusion process, fundamentally altering the geometry of exploration. We discuss further related works in Appendix A.

7 Conclusion

RL for LLM reasoning suffers from diversity collapse, and token-level exploration mostly yields surface variations of the same reasoning. LaDi-RL therefore explores entire reasoning trajectories with a latent diffusion policy, and uses hierarchical latent–text rollouts to credit each latent by the mean reward of its decodings, not by one answer. As a result, it improves pass@1 over vanilla GRPO by 10.5 points on code and 6.5 on math while keeping pass@k above its LaDiR initialization at all k.

AI USE STATEMENT

We used generative AI tools only to polish the writing of this paper, such as grammar, wording, and readability, after the technical content had been written by the authors. We did not use generative AI tools for research ideation, method design, code implementation, running experiments, analyzing data, or producing results. Separately from writing, GPT-5 serves as an evaluation instrument for the semantic-diversity metric, as described in Section 5.2 and Appendix E.5; it is not used for training rewards or for measuring correctness. The authors reviewed all AI-assisted text and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix C give the full training objective, including the Flow-GRPO-Fast sampler with CPS, the dimension-averaged guided likelihood ratio (Eq. (24)), and the diversity guidance (Appendix C.8). Appendix D describes the datasets, data filtering, backbones, and all hyperparameters (Table 6). The evaluation protocol (16 samples per problem for pass@1, 128 samples for pass@k, and the 4,096-token budget for math) is given in Section 4.1 and Appendix D.2. Appendix E.2 reports the compute for each training stage, and Appendix E.5 gives the semantic-diversity protocol, including the judge prompt and its validation against human annotation.

REFERENCES

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in LLM reasoning. *arXiv preprint arXiv:2505.15134*, 2025.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

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.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Umberto Borso, Davide Paglieri, Jude Wells, and Tim Rocktäschel. D3po: Preference-based alignment of discrete diffusion models. *arXiv e-prints*, pp. arXiv–2503, 2025.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. *arXiv preprint arXiv:2509.19170*, 2025.

Feng Chen, Allan Raventos, Nan Cheng, Surya Ganguli, and Shaul Druckmann. Rethinking fine-tuning when scaling test-time compute: Limiting confidence improves mathematical reasoning. *arXiv preprint arXiv:2502.07154*, 2025a.

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, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

P Chen et al. Exploration vs exploitation: Rethinking RLVR through clipping, entropy, and spurious reward. *arXiv preprint arXiv:2512.16912*, 2025b.

Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, Ting-Che Lin, Shilong Zhang, Fu Li, Chuan Li, Xing Wang, Yanghua Peng, Peize Sun, Ping Luo, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Goku: Flow based video generative foundation models, 2025c. URL https://arxiv.org/abs/2502.04896.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling, 2025d. URL https://arxiv.org/abs/2501.17811.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking, 2025e. URL https://arxiv.org/abs/2502.13842.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025f.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations, 2024. URL https://arxiv.org/abs/2412.13171.

Ho-Lam Chung, Teng-Yun Hsiao, Hsiao-Ying Huang, Chunerh Cho, Jian-Ren Lin, Zhang Ziwei, and Yun-Nung Chen. Revisiting test-time scaling: A survey and a diversity-aware method for efficient reasoning. *arXiv preprint arXiv:2506.04611*, 2025.

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.

Codefuse, Ling Team, Wenting Cai, Yuchen Cao, Chaoyu Chen, Chen Chen, Siba Chen, Qing Cui, Peng Di, Junpeng Fang, Zi Gong, Ting Guo, Zhengyu He, Yang Huang, Cong Li, Jianguo Li, Zheng Li, Shijie Lian, BingChang Liu, Songshan Luo, Shuo Mao, Min Shen, Jian Wu, Jiaolong Yang, Wenjie Yang, Tong Ye, Hang Yu, Wei Zhang, Zhenduo Zhang, Hailin Zhao, Xunjin Zheng, and Jun Zhou. Every sample matters: Leveraging mixture-of-experts and high-quality data for efficient and accurate code LLM, 2025. URL https://arxiv.org/abs/2503.17793.

Gabriele Corso, Yilun Xu, Valentin De Bortoli, Regina Barzilay, and Tommi Jaakkola. Particle guidance: non-I.I.D. diverse sampling with diffusion models. In *International Conference on Learning Representations*, 2024.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Xingyu Dang, Christina Baek, Kaiyue Wen, Zico Kolter, and Aditi Raghunathan. Weight ensembling improves reasoning in language models. *arXiv preprint arXiv:2504.10478*, 2025.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens, 2024. URL https://arxiv.org/abs/2410.13863.

Tianyu Fu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang. Think-at-hard: Dynamic looped transformers for improved reasoning. *arXiv preprint arXiv:2511.08577*, 2025.

Jingchu Gai, Guanning Zeng, Huaqing Zhang, and Aditi Raghunathan. Differential smoothing mitigates sharpening and improves LLM reasoning. *arXiv preprint arXiv:2511.19942*, 2025.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

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

Halil Alperen Gozeten, M Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. *arXiv preprint arXiv:2505.23648*, 2025.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. arXiv preprint arXiv:2505.23564, 2025b. URL https://arxiv.org/abs/2505.23564.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024. URL https://arxiv.org/abs/2412.06769.

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, et al. Rethinking entropy interventions in RLVR: An entropy change perspective. *arXiv preprint arXiv:2510.10150*, 2025.

Andre Wang He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25559–25571, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

David Herel and Tomas Mikolov. Thinking tokens for language modeling, 2024. URL https://arxiv.org/abs/2405.08644.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. arXiv preprint arXiv:2506.11902, 2025. URL https://arxiv.org/abs/2506.11902.

Zican Hu, Shilin Zhang, Yafu Li, Jianhao Yan, Xuyang Hu, Leyang Cui, Xiaoye Qu, Chunlin Chen, Yu Cheng, and Zhi Wang. Diversity-incentivized exploration for versatile reasoning. *arXiv preprint arXiv:2509.26209*, 2025.

F Huang et al. Semantic-space exploration and exploitation in RLVR for LLM reasoning. *arXiv preprint arXiv:2509.23808*, 2025a.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, et al. Opencoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33167–33193, 2025b.

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*, 2025c.

Hugging Face. Open r1: A fully open reproduction of DeepSeek-R1, January 2025. URL `https://github.com/huggingface/open-r1`.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. arXiv preprint arXiv:2509.21240, 2026. URL `https://arxiv.org/abs/2509.21240`.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models, 2025a. URL `https://arxiv.org/abs/2411.13504`.

R Jin, Y Gao, et al. Revisiting entropy in reinforcement learning for large reasoning models. *arXiv preprint arXiv:2511.05993*, 2025b.

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yi-An Ma, and Lianhui Qin. LaDiR: Latent diffusion enhances LLMs for text reasoning. *arXiv preprint arXiv:2510.04573*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. arXiv preprint arXiv:2410.01679, 2025. URL `https://arxiv.org/abs/2410.01679`.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

C Li, N Liu, and K Yang. Adaptive group policy optimization: Towards stable training and token-efficient reasoning. *arXiv preprint arXiv:2503.15952*, 2025a.

Junzhe Li, Yutao Cui, Tao Huang, Weijie Kong, Yiming Cheng, Chuxuan Zeng, Yinping Ma, Chun Fan, Miles Yang, Zhao Zhong, and Liefeng Bo. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE-SDE. *arXiv preprint arXiv:2507.21802*, 2025b.

Qingbin Li, Rongkun Xue, Jie Wang, Ming Zhou, Zhi Li, Xiaofeng Ji, Yongqi Wang, Miao Liu, Zheming Yang, Minghui Qiu, et al. Cure: Critical-token-guided re-concatenation for entropy-collapse prevention. *arXiv preprint arXiv:2508.11016*, 2025c.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025d.

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.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, Zheng Zhang, Wei Shen, Qian Liu, Chenghua Lin, Jian Yang, Ge Zhang, and Wenhao Huang. TreePO: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. arXiv preprint arXiv:2508.17445, 2025e. URL https://arxiv.org/abs/2508.17445.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Zhi-Quan Luo, and Ruoyu Sun. Preserving diversity in supervised fine-tuning of large language models. In *International Conference on Learning Representations*, 2025f.

Z Lin, M Lin, Y Xie, and R Ji. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *arXiv preprint arXiv:2503.22342*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO-Fast: Accelerated Flow-GRPO training on one or two denoising steps. Official Flow-GRPO code repository, https://github.com/yifan123/flow_grpo, 2025a.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. *arXiv preprint arXiv:2505.05470*, 2025b.

Tianqiao Liu, Zui Chen, Zitao Liu, Mi Tian, and Weiqi Luo. Expediting and elevating large language model reasoning via hidden chain-of-thought decoding, 2024. URL https://arxiv.org/abs/2409.08561.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Yifei Liu, Li Lyna Zhang, Yi Zhu, Bingcheng Dong, Xudong Zhou, Ning Shang, Fan Yang, and Mao Yang. rstar-coder: Scaling competitive code reasoning with a large-scale verified dataset. *arXiv preprint arXiv:2505.21297*, 2025c.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36: 56998–57025, 2023.

Justin Lovelace, Varsha Kishore, Yiwei Chen, and Kilian Weinberger. Diffusion guided language modeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14936–14952, 2024.

Justin Lovelace, Christian K Belardi, Sofian Zalouk, Adhitya Polavaram, Srivatsa R Kundurthy, and Kilian Q Weinberger. Stop-think-autoregress: Language modeling with latent diffusion planning. In *Second Conference on Language Modeling*, 2025.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling RL. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

Sathwik Tejaswi Madhusudhan, Shruthan Radhakrishna, Jash Mehta, and Toby Liang. Millions scale dataset distilled from r1-32b. https://huggingface.co/datasets/ServiceNow-AI/R1-Distill-SFT, 2025.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. Compressed and smooth latent space for text diffusion modeling. *arXiv preprint arXiv:2506.21170*, 2025.

Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. CoTFormer: A chain of thought driven architecture with budget-adaptive computation cost at inference. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7igPXQFupX.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. Diversity of thought improves reasoning abilities of LLMs. *arXiv preprint arXiv:2310.07088*, 2023.

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.

OpenAI. Learning to reason with LLMs. *OpenAI Blog*, 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.

Enes Özeren and Matthias Aßenmacher. Reinforcement learning for latent-space thinking in LLMs. *arXiv preprint arXiv:2512.11816*, 2025.

Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, Ji Hou, and Saining Xie. Transfer between modalities with metaqueries, 2025. URL https://arxiv.org/abs/2504.06256.

JR Park et al. Clip-low increases entropy and clip-high decreases entropy in reinforcement learning of large language models. *arXiv preprint arXiv:2509.26114*, 2025.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models, 2024. URL https://arxiv.org/abs/2404.15758.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

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. *arXiv preprint arXiv:2510.08554*, 2025.

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.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *arXiv preprint arXiv:2506.10892*, 2025.

Soham Sane. Hybrid group relative policy optimization: A multi-sample approach to enhancing policy optimization. *arXiv preprint arXiv:2502.01652*, 2025.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers, 2025. URL https://arxiv.org/abs/2502.17416.

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.

ByteDance Seed, Yuyu Zhang, Jing Su, Yifan Sun, Chenguang Xi, Xia Xiao, Shen Zheng, Anxiang Zhang, Kaibo Liu, Daoguang Zan, et al. Seed-coder: Let the code model curate data for itself. *arXiv preprint arXiv:2506.03524*, 2025.

C Shao et al. Diffuse thinking: Exploring diffusion language models as efficient thought proposers for reasoning. *arXiv preprint arXiv:2510.27469*, 2025.

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.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation, 2025. URL https://arxiv.org/abs/2502.21074.

Weijia Shi, Xiaochuang Han, Chunting Zhou, Weixin Liang, Xi Victoria Lin, Luke Zettlemoyer, and Lili Yu. Lmfusion: Adapting pretrained language models for multimodal generation, 2025. URL https://arxiv.org/abs/2412.15188.

Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Carina Negreanu, and Gust Verbruggen. Codefusion: A pre-trained diffusion model for code generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11697–11708, 2023.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. *arXiv preprint arXiv:2412.02674*, 2024.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. LLM pretraining with continuous concepts, 2025. URL https://arxiv.org/abs/2502.08524.

Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer, 2024. URL https://arxiv.org/abs/2410.10812.

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

Yao Tang, Li Dong, Yaru Hao, Qingxiu Dong, Furu Wei, and Jiatao Gu. Multiplex thinking: Reasoning via token-wise branch-and-merge. *arXiv preprint arXiv:2601.08808*, 2026.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Rémi Munos. Optimizing language models for inference time objectives using reinforcement learning. *arXiv preprint arXiv:2503.19595*, 2025b.

Shengbang Tong, David Fan, Jiachen Zhu, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun, Saining Xie, and Zhuang Liu. Metamorph: Multimodal understanding and generation via instruction tuning, 2024. URL https://arxiv.org/abs/2412.14164.

Hieu Tran, Zonghai Yao, and Hong Yu. Exploiting tree structure for credit assignment in RL training of LLMs. arXiv preprint arXiv:2509.18314, 2025. URL https://arxiv.org/abs/2509.18314.

Hemish Veeraboina. AIME problem set 1983-2024. Kaggle, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

S Venkatraman et al. Reasoning with latent diffusion in offline reinforcement learning. *ICLR*, 2024.

Christian Walder and Deep Karkhanis. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025.

Zhongwei Wan, Yun Shen, Zhihao Dou, Donghao Zhou, Yu Zhang, Xin Wang, Hui Shen, Jing Xiong, Chaofan Tao, Zixuan Zhong, Peizhou Huang, and Mi Zhang. DSDR: Dual-scale diversity regularization for exploration in LLM reasoning. *arXiv preprint arXiv:2602.19895*, 2026.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, et al. Spg: Sandwiched policy gradient for masked diffusion language models. *arXiv preprint arXiv:2510.09541*, 2025a.

Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improving reasoning in diffusion language models via trajectory likelihood estimation. *arXiv preprint arXiv:2509.21474*, 2025b.

Hongcheng Wang, Yinuo Huang, Sukai Wang, Guanghui Ren, and Hao Dong. Why tree-style branching matters for thought advantage estimation in GRPO. arXiv preprint arXiv:2509.24494, 2026. URL https://arxiv.org/abs/2509.24494.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *arXiv preprint arXiv:2506.01939*, 2025c.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wi9IffRhVM.

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*, 2025d.

Chünhung Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. LLMs are single-threaded reasoners: Demystifying the working mechanism of soft thinking. *arXiv preprint arXiv:2508.03440*, 2025.

Jianxiang Xiang, Zhenhua Liu, Haodong Liu, Yin Bai, Jia Cheng, and Wenliang Chen. Diffusiondialog: A diffusion model for diverse dialog generation with latent space. *arXiv preprint arXiv:2404.06760*, 2024.

Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Chaofan Li, Shuting Wang, Tiejun Huang, and Zheng Liu. Omnigen: Unified image generation, 2024. URL https://arxiv.org/abs/2409.11340.

Can Xie, Ruotong Pan, Xiangyu Wu, Yunfei Zhang, Jiayi Fu, Tingting Gao, and Guorui Zhou. Unlocking exploration in RLVR: Uncertainty-aware advantage shaping for deeper reasoning. *arXiv preprint arXiv:2510.10649*, 2025a.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-coder 7b: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025b.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. *arXiv preprint arXiv:2503.02951*, 2025.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

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

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.

Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. Dcpo: Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*, 2025c.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. TreeRPO: Tree relative policy optimization. arXiv preprint arXiv:2506.05183, 2025d. URL https://arxiv.org/abs/2506.05183.

Jian Yao, Ran Cheng, Xingyu Wu, Jibin Wu, and Kay Chen Tan. Diversity-aware policy optimization for large language model reasoning. *arXiv preprint arXiv:2505.23433*, 2025.

Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, and Lingpeng Kong. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. *Advances in Neural Information Processing Systems*, 2024.

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

Qifan Yu, Zhenyu He, Sijie Li, Xun Zhou, Jun Zhang, Jingjing Xu, and Di He. Enhancing auto-regressive chain-of-thought through loop-aligned reasoning, 2025a. URL https://arxiv.org/abs/2502.08482.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. Dapo: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Oussama Zekri and Nicolas Boullé. Fine-tuning discrete diffusion models with policy gradient methods. *arXiv preprint arXiv:2502.01384*, 2025.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. Quiet-STar: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=oRXPiSOGH9.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder RL via automated test-case synthesis. *arXiv preprint arXiv:2502.01718*, 2025a.

Yiming Zeng, Jinghan Cao, Zexin Li, Yiming Chen, Tao Ren, Zhuochun Li, Dawei Xiang, Xidong Wu, Shangqian Gao, and Tingting Yu. Treediff: Ast-guided code generation with diffusion LLMs. *arXiv preprint arXiv:2508.01473*, 2025b.

J Zhang, J Huang, H Yao, S Liu, X Zhang, S Lu, et al. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025a.

K Zhang et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025b.

Shimin Zhang, Xianwei Chen, Yufan Shen, Ziyuan Ye, and Jibin Wu. Relax: Reasoning with latent exploration for large reasoning models. *arXiv preprint arXiv:2512.07558*, 2025c.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2025. https://huggingface.co/datasets/math-ai/aime25, 2025.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua Susskind, and Navdeep Jaitly. Planner: Generating diversified paragraph via latent language diffusion model. *Advances in Neural Information Processing Systems*, 36:80178–80190, 2023.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025d.

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.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025b.

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*, 2025c.

Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, et al. First return, entropy-eliciting explore. *arXiv preprint arXiv:2507.07017*, 2025a.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. Soft-grpo: Surpassing discrete-token LLM reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025b.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner. *arXiv preprint arXiv:2510.03206*, 2025.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model, 2024. URL https://arxiv.org/abs/2408.11039.

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*, 2025a.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025b.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025c.

Nicolas Zilberstein, Morteza Mardani, and Santiago Segarra. Repulsive latent score distillation for solving inverse problems. *arXiv preprint arXiv:2406.16683*, 2024.

A Additional Related Works

Latent Reasoning. Latent reasoning methods address the limitations of token-level CoT by shifting reasoning processes into a latent space. Early approaches utilized discrete special tokens to expand internal reasoning capacity or encode implicit intermediate steps, yielding more abstract representations (Herel & Mikolov, 2024; Pfau et al., 2024; Wang et al., 2024; Zelikman et al., 2024; Zhou et al., 2025; Jin et al., 2025a). Subsequent work along this direction demonstrated that reasoning via continuous latent representations rather than discrete tokens can further improve performance. By operating on "soft" tokens that are either self-generated or produced by auxiliary models, LLMs can exploit richer semantic information (Gozeten et al., 2025; Cheng & Durme, 2024; Hao et al., 2024; Liu et al., 2024; Shen et al., 2025; Tack et al., 2025; Zhu et al., 2025b; Butt et al., 2025; Zhang et al., 2025d; Wu et al., 2025). Furthermore, recent research leverages the expressive power of diffusion models to propose and refine reasoning trajectories (Kang et al., 2025; Ye et al., 2024; Shao et al., 2025; Venkatraman et al., 2024; Ye et al., 2024; Lovelace et al., 2025; Zhang et al., 2023; Lovelace et al., 2023; 2024). In parallel, a complementary line of work utilizes recurrent or looped architectures to induce latent reasoning internally, bypassing the need to explicitly represent reasoning steps as output tokens (Chen et al., 2025e; Geiping et al., 2025; Mohtashami et al., 2025; Saunshi et al., 2025; Yu et al., 2025a). Building on this progress, recent efforts apply RL to continuous latent spaces by injecting stochasticity into token embeddings (Butt et al., 2025; Sane, 2025; Zheng et al., 2025b; Tang et al., 2026; Özeren & Aßenmacher, 2025) or by regulating exploration during policy optimization (Zhang et al., 2025c). Compared with these methods, our method injects noise via a latent diffusion process, enabling exploration over reasoning trajectories rather than token variations.

Latent Diffusion for Language Generation Recent work has extended language generation beyond autoregressive decoding to diffusion-based models that enable global, iterative refinement. Early approaches such as Diffusion-LM (Li et al., 2022) formulate generation as denoising continuous word embeddings, while subsequent methods perform diffusion in compressed latent spaces to improve text quality and mode diversity (Lovelace et al., 2023; 2025; Meshchaninov et al., 2025). For sequence-to-sequence tasks, DiffuSeq (Gong et al., 2022) enables parallel generation with high diversity, and PLANNER (Zhang et al., 2023) combines latent semantic diffusion with autoregressive decoding to better handle long-form text. Diffusion has also been applied to domain-specific generation, including dialogue (Xiang et al., 2024) and code synthesis (Singh et al., 2023; Zeng et al., 2025b). More recently, diffusion-guided language modeling (Lovelace et al., 2024; 2025) explores using diffusion as a global guidance mechanism to improve generation quality and controllability. Despite these advances, prior latent diffusion models primarily target fluent text generation and lack exploration into reasoning tasks.

Reinforcement Learning for Discrete Diffusion Language Models. Diffusion language models (DLMs) (Sahoo et al., 2024; 2025; Nie et al., 2025; Ye et al., 2025; Song et al., 2025) provide a promising alternative to autoregressive models, but reinforcement learning (RL) for DLMs faces distinct structural challenges due to the combinatorial explosion of denoising trajectories and the lack of well-defined state transitions. Early token-level adaptations (Zhao et al., 2025c; Yang et al., 2025b; Gong et al., 2025) rely on ill-posed transition dynamics and mean-field approximations, leading to unstable optimization. Recent work therefore shifts to sequence- or trajectory-level objectives (Zhu et al., 2025a; Wang et al., 2025a; Rojas et al., 2025; Ou et al., 2025; Zekri & Boullé, 2025), using surrogates to approximate intractable marginal likelihoods. However, these methods generally suffer from off-policy misalignment: heuristic-guided sampling deviates from the diffusion prior, producing biased gradients without principled correction (Schulman et al., 2015). Representative RL training methods (Zhao et al., 2025c; Gong et al., 2022; Tang et al., 2025a; Wang et al., 2025b; Zhao et al., 2025a; Borso et al., 2025) differ mainly in how likelihoods are approximated. Two notable exceptions partially address this issue: LLaDOU (Huang et al., 2025c) explicitly models diffusion-step likelihoods via an auxiliary policy at high computational cost, while TraceRL (Wang et al., 2025d) aligns optimization with inference traces by merging diffusion steps.

Hybrid AR+Diffusion Model Architecture Hybrid autoregressive–diffusion (AR–diffusion) models have demonstrated strong performance across multimodal generation and understanding, often rivaling or surpassing their purely AR or diffusion counterparts. The Transfusion architecture (Zhou et al., 2024) showed that hybrid models can outperform standard AR models and remain competitive

with state-of-the-art diffusion models on image generation benchmarks, a trend further supported by subsequent works (Fan et al., 2024; Tang et al., 2024; Xiao et al., 2024). Beyond image generation, hybrid AR–diffusion models have proven effective in image understanding, video generation, and robot control (Black et al., 2024; Tong et al., 2024; Chen et al., 2025c;d). Similar to our approach, several studies adapt frozen pretrained models within hybrid AR–diffusion architectures for multimodal tasks (Pan et al., 2025; Shi et al., 2025).

Tree-Structured Rollouts and Credit Assignment A related line of work structures rollouts as trees to sharpen credit assignment in LLM RL. VinePPO (Kazemnejad et al., 2025) estimates the value of intermediate states with Monte Carlo rollouts started from them. TreeRL (Hou et al., 2025) and TreeRPO (Yang et al., 2025d) branch from intermediate reasoning steps and derive step-level rewards or advantages from the resulting subtrees, and TreePO (Li et al., 2025e) shares prefixes to amortize computation and assigns advantages at each branching depth. SPO (Guo et al., 2025b) estimates advantages at the level of segments, Tree-GRPO (Ji et al., 2026) applies tree search to multi-turn agents, and Tran et al. (2025) and Wang et al. (2026) study how tree-structured sampling improves advantage estimation. All of these methods branch inside the token sequence, so the samples in a subtree share a textual prefix. LaDi-RL instead branches once, at the latent reasoning trajectory: the M answers in a group share a latent sampled by the diffusion policy rather than a text prefix, so the credit for the reasoning is separated from the credit for how it is written.

B LIMITATIONS

Our study focuses on verifiable math and code tasks, where rewards can be computed automatically; extending LaDi-RL to open-ended tasks with learned reward models is a natural next step. Our main experiments use 7–8B backbones, with one 32B result (Appendix E.2); scaling to larger models is left to future work. Finally, semantic diversity is measured with an LLM judge, which we validate against human annotations and judge-independent metrics (Appendix E.5).

C PRELIMINARIES: FLOW-GRPO

In this section, we provide a detailed overview of Flow-GRPO (Liu et al., 2025b) and its accelerated variant Flow-GRPO-Fast, which enable online reinforcement learning for flow matching models. We describe the mathematical formulation, the ODE-to-SDE conversion strategy, the GRPO objective, and key implementation parameters.

C.1 FLOW MATCHING BACKGROUND

Flow matching models (Lipman et al., 2022; Liu et al., 2022) define a continuous-time generative process from noise to data. Let $x_0 \sim \mathcal{X}_0$ denote a sample from the data distribution and $x_1 \sim \mathcal{X}_1 = \mathcal{N}(0, I)$ denote a noise sample. The Rectified Flow framework (Liu et al., 2022) defines the interpolated state x_t as:

$$x_t = (1-t)x_0 + tx_1, \quad t \in [0,1]. \tag{8}$$

A neural network $v_\theta(x_t, t)$ is trained to regress the velocity field by minimizing the flow matching objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x_0\sim\mathcal{X}_0,x_1\sim\mathcal{X}_1}\left[\|v - v_\theta(x_t,t)\|^2\right], \tag{9}$$

where $t \sim \mathcal{U}[0,1]$, x_0 and x_1 are sampled independently, and the target velocity is $v = x_1 - x_0$. As in the rest of the paper, t=0 is data and t=1 is noise, so sampling runs from t=1 to t=0.

C.2 DENOISING AS A MARKOV DECISION PROCESS

Following (Black et al., 2023), iterative denoising can be formulated as an MDP $(\mathcal{S}, \mathcal{A}, \rho_0, P, R)$. Let $k = 0, \ldots, T-1$ count environment steps, while the discrete diffusion index decreases from T to 0:

- **State**: $s_k = (c, T-k, x_{T-k})$.
- **Action**: $a_k = x_{T-k-1}$, the next denoised sample.
- **Policy**: $\pi_\theta(a_k \mid s_k) = p_\theta(x_{T-k-1} \mid x_{T-k}, c)$.

- **Transition**: $P(\cdot \mid s_k, a_k) = \delta_{(c,T-k-1,a_k)}$, a point mass at the next state.
- **Initial distribution**: sample $c \sim p(c)$ and $x_T \sim \mathcal{N}(0, I)$ independently, giving $s_0 = (c, T, x_T)$.
- **Reward**: $R(s_k, a_k) = r(a_k, c)$ for $k = T - 1$, when $a_k = x_0$, and zero otherwise.

C.3 ODE-TO-SDE CONVERSION

A critical challenge for applying RL to flow matching models is their deterministic nature—standard ODE-based sampling provides no stochasticity for exploration. Flow-GRPO addresses this by converting the deterministic ODE:

$$dx_t = v_t \, dt \tag{10}$$

into a reverse-time SDE with the same marginals when the exact velocity field and score are used. With learned approximations and finite steps, marginal preservation is approximate. Following the theoretical framework (Song et al., 2020; Albergo et al., 2023), Flow-GRPO constructs a reverse-time SDE:

$$dx_t = \left(v_t(x_t) - \frac{g_t^2}{2} \nabla \log p_t(x_t) \right) dt + g_t \, dw, \tag{11}$$

where dw denotes reverse-time Wiener increments with covariance $|dt|I$. The diffusion coefficient g_t is distinct from the CPS transition standard deviation σ_t used below.

For independent x_0 and Gaussian x_1, the exact marginal velocity $v_t(x) = \mathbb{E}[x_1 - x_0 \mid x_t = x]$ gives the score identity, for $t > 0$:

$$\nabla \log p_t(x) = -\frac{x}{t} - \frac{1-t}{t} v_t(x). \tag{12}$$

Substituting Equation (12) into Equation (11) yields the specific SDE formulation:

$$dx_t = \left(v_t(x_t) + \frac{g_t^2}{2t} \left(x_t + (1-t) v_t(x_t) \right) \right) dt + g_t \, dw. \tag{13}$$

Applying Euler-Maruyama discretization gives the practical update rule:

$$x_{t+\Delta t} = x_t + \left(v_\theta(x_t, t) + \frac{g_t^2}{2t} \left(x_t + (1-t) v_\theta(x_t, t) \right) \right) \Delta t + g_t \sqrt{|\Delta t|} \, \epsilon, \tag{14}$$

where $\epsilon \sim \mathcal{N}(0, I)$ injects stochasticity and $\Delta t < 0$, since sampling moves from noise (t=1) toward data (t=0). The noise schedule is parameterized as:

$$g_t = a \sqrt{\frac{t}{1-t}}, \tag{15}$$

This schedule is defined for $0 < t < 1$ and is singular at $t = 1$, so a finite-step implementation must handle that endpoint separately. Here a controls the noise level (typically $a = 0.7$). Our implementation uses the CPS sampler below, whose stochasticity is set by η instead of a (Table 6).

C.4 GRPO OBJECTIVE FOR FLOW MATCHING

Given a prompt c, the flow model samples a group of G latent blocks $\{x_0^i\}_{i=1}^G$ with corresponding trajectories $\{(x_T^i, x_{T-1}^i, \ldots, x_0^i)\}_{i=1}^G$. The advantage of the i-th sample is computed via group-relative normalization:

$$\hat{A}_t^i = \frac{R(x_0^i, c) - \text{mean}(\{R(x_0^i, c)\}_{i=1}^G)}{\text{std}(\{R(x_0^i, c)\}_{i=1}^G) + \varepsilon_{\text{adv}}}. \tag{16}$$

The Flow-GRPO objective is:

$$J_{\text{Flow-GRPO}}(\theta) = \mathbb{E}_{c \sim \mathcal{C}, \{x^i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|c)} \left[f(r, \hat{A}, \theta, \epsilon, \beta) \right], \tag{17}$$

where

$$f(r, \hat{A}, \theta, \epsilon, \beta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=1}^{T}\left(\min\left(r_t^i(\theta)\hat{A}_t^i, \text{clip}(r_t^i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^i\right) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right), \tag{18}$$

and the importance ratio is:

$$r_t^i(\theta) = \frac{p_\theta(x_{t-1}^i | x_t^i, c)}{p_{\theta_{\text{old}}}(x_{t-1}^i | x_t^i, c)}. \tag{19}$$

At a fixed state and a stochastic Euler step, let μ_θ and μ_{ref} denote the transition means. With the same positive covariance $g_t^2|\Delta t|I$ for both policies, the conditional KL divergence is:

$$D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \frac{\|\mu_\theta(x_t) - \mu_{\text{ref}}(x_t)\|^2}{2g_t^2|\Delta t|} = \frac{|\Delta t|}{2}\left(\frac{g_t(1-t)}{2t} + \frac{1}{g_t}\right)^2 \|v_\theta(x_t, t) - v_{\text{ref}}(x_t, t)\|^2. \tag{20}$$

C.5 Denoising Reduction

Flow-GRPO introduces a **Denoising Reduction** strategy to improve training efficiency. While standard inference may require $T = 40$ denoising steps, training samples are collected with significantly fewer steps (e.g., $T_{\text{train}} = 10$), while retaining the full schedule during evaluation. This achieves over $4\times$ speedup without sacrificing final performance.

C.6 Flow-GRPO-Fast

Flow-GRPO-Fast (Liu et al., 2025a) is an accelerated variant that requires training on only **one or two denoising steps** per trajectory. The key idea is to confine stochasticity to a narrow window.

In the implementation we use, each sample starts from its own initial noise (no shared prefix; Table 6). A window of `sde_window_size` consecutive steps is drawn at random within `sde_window_range`; stochastic CPS steps are used inside the window and deterministic (η=0) steps elsewhere. Only the window steps enter the GRPO loss, which reduces the training cost per trajectory by a factor of about $T/|\mathcal{T}|$ (here $10/2$).

SDE Window Mechanism (Flow-GRPO-Fast). In Flow-GRPO-Fast (see also the sliding window of MixGRPO (Li et al., 2025b)), the `sde_window_size` parameter controls how many consecutive denoising steps use SDE sampling (and are subsequently optimized), while `sde_window_range` specifies the valid timestep range within which the SDE window can be randomly positioned. For instance, setting `sde_window_size`$= 2$ and `sde_window_range`$= [0, T-2]$ means that at each iteration, a random starting position is sampled, and SDE sampling (with GRPO optimization) is applied only to those two steps. All other steps use deterministic ODE sampling.

No-CFG Training. Disabling classifier-free guidance (CFG) (Ho & Salimans, 2022) during training effectively performs CFG distillation through the RL process, significantly accelerating convergence while maintaining or improving generation quality.

C.7 CPS Sampling for Flow Matching Models

In our implementation, we adopt *Coefficients-Preserving Sampling* (CPS) (Wang & Yu, 2025) to introduce principled stochasticity into flow matching models while preserving the consistency of noise coefficients. With finite steps, the Euler discretization of Flow-SDE does not keep the noise coefficients on schedule, because its drift correction does not exactly offset the noise injected at each step. CPS balances the squared noise coefficients exactly, and its transition remains Gaussian, so the likelihood stays tractable.

Motivated by the observation that DDIM sampling injects noise while preserving coefficient consistency, we reformulate the sampling procedure to satisfy the CPS conditions. Let $\hat{\mathbf{x}}_0$ and $\hat{\mathbf{x}}_1$ denote the predicted clean sample and noise component, respectively, $\hat{\mathbf{x}}_0 = \mathbf{x}_t - t\, v_\theta(\mathbf{x}_t, t)$ and $\hat{\mathbf{x}}_1 = \mathbf{x}_t + (1-t)\, v_\theta(\mathbf{x}_t, t)$. In this subsection $\Delta t > 0$ is the step size, so each step moves from t to

$t - \Delta t$. By constraining the variance of the injected noise and adjusting the coefficients accordingly, the CPS update rule is given by:

$$\mathbf{x}_{t-\Delta t} = \big(1 - (t - \Delta t)\big)\hat{\mathbf{x}}_0 + \sqrt{(t - \Delta t)^2 - \sigma_t^2}\,\hat{\mathbf{x}}_1 + \sigma_t \boldsymbol{\epsilon}, \tag{21}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To avoid invalid square roots, we parameterize the noise scale as

$$\sigma_t = (t - \Delta t)\sin\Big(\frac{\eta\pi}{2}\Big), \quad \eta \in [0, 1],$$

which yields the final CPS sampling formulation:

$$\mathbf{x}_{t-\Delta t} = \big(1 - (t - \Delta t)\big)\hat{\mathbf{x}}_0 + (t - \Delta t)\cos\Big(\frac{\eta\pi}{2}\Big)\,\hat{\mathbf{x}}_1 + (t - \Delta t)\sin\Big(\frac{\eta\pi}{2}\Big)\,\boldsymbol{\epsilon}. \tag{22}$$

The parameter η explicitly controls the stochastic strength, interpolating between deterministic flow sampling ($\eta = 0$) and maximal stochasticity ($\eta = 1$). This formulation admits a clear geometric interpretation and preserves the coefficient structure required by CPS; we therefore refer to this procedure as *Flow-CPS*.

The following density applies only to stochastic steps with $\sigma_t > 0$ (hence $t - \Delta t > 0$). At $\eta = 0$ or $t - \Delta t = 0$, the transition is deterministic and its Gaussian log-density is not evaluated. To apply GRPO, we require the conditional transition likelihood $p_\theta(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$. Following (Liu et al., 2025b), this likelihood is defined as:

$$\log p_\theta(\mathbf{x}^i_{t-\Delta t} \mid \mathbf{x}^i_t) = -\frac{\|\mathbf{x}^i_{t-\Delta t} - \boldsymbol{\mu}_\theta(\mathbf{x}^i_t, t)\|^2}{2\sigma_t^2} - D\log\sigma_t - \tfrac{D}{2}\log 2\pi, \tag{23}$$

where

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \big(1 - (t - \Delta t)\big)\hat{\mathbf{x}}_0 + (t - \Delta t)\cos\Big(\frac{\eta\pi}{2}\Big)\,\hat{\mathbf{x}}_1.$$

Here D is the latent dimension. Equation (23) describes the base CPS kernel. For the guided training policy, condition on the stored group state $\mathbf{Z}^t$ and define $y_n = \mathbf{z}_n^{t-1}$, $\mu_{\phi,n,t} = \boldsymbol{\mu}_\phi(\mathbf{z}_n^t, \tau_t, Q)$, and $\Delta_{n,t} = \gamma_t F_n(\mathbf{Z}^t)$. Its log-density is

$$\log q_\phi(y_n \mid \mathbf{Z}^t, Q) = -\frac{\|y_n - \mu_{\phi,n,t} - \Delta_{n,t}\|^2}{2\sigma_t^2} - D\log\sigma_t - \frac{D}{2}\log(2\pi).$$

Following the Flow-GRPO implementation, Eq. (3) does not use the exact ratio $q_\phi/q_{\phi_{\text{old}}}$ but its dimension-averaged form:

$$\ell_\phi(y_n) = -\frac{1}{D}\,\|y_n - \mu_{\phi,n,t} - \Delta_{n,t}\|^2, \qquad r_{n,t}(\phi) = \exp\big(\ell_\phi(y_n) - \ell_{\phi_{\text{old}}}(y_n)\big). \tag{24}$$

Since $\log q_\phi = \frac{D}{2\sigma_t^2}\,\ell_\phi + \text{const}$, the averaged log-ratio is the exact guided log-ratio scaled by $2\sigma_t^2/D$. The clip range acts on this scaled ratio: $1 \pm \epsilon_z$ corresponds to $\frac{D}{2\sigma_t^2}\log(1 \pm \epsilon_z)$ on the exact log-ratio, which is why ϵ_z is small (Table 6).

Policy update with guidance. Diversity guidance changes the distribution that generates the rollouts: each latent is sampled from the guided kernel $q_\phi(y_n \mid \mathbf{Z}^t, Q) = \mathcal{N}(\mu_{\phi,n,t} + \Delta_{n,t},\, \sigma_t^2\mathbf{I})$ rather than from the unguided CPS kernel. We therefore compute the importance ratio in Eq. (3) under this guided kernel, with the same shift $\Delta_{n,t}$ in the current and old likelihoods (Eq. (24)), so that the ratio refers to the policy that actually produced the samples. This is how DDPO handles classifier-free guidance, using the guided prediction for both sampling and training (Black et al., 2023, Appendix E.1). Given $\mathbf{Z}^t$, each latent is sampled with its own noise and weighted by its own advantage $\hat{A}_n$, so each latent is credited for its own answers, as in the hierarchical rollout of Section 3.1.

Implementation. We store each sampled transition together with the group state, guidance shift, noise scale, and old score $\ell_{\phi_{\text{old}}}$. During optimization, we recompute the current guided mean on the stored state and form $r_{n,t} = \exp(\ell_\phi - \ell_{\phi_{\text{old}}})$ as in Eq. (24). Gradients flow through the current mean; the samples, guidance shift, and old score are held fixed. Only stochastic steps with $\sigma_t > 0$ enter the ratio. Guidance is used only to collect training rollouts; at evaluation we set $\gamma_t = 0$ and sample each latent independently.

C.8 Design of the Diversity Guidance

The force is the gradient of a pairwise potential. Write $d = \|\mathbf{z}_n^t - \mathbf{z}_{n'}^t\|_2$. For $h > 0$, holding the bandwidth fixed within the step (without differentiating through the median), each term of the guidance force satisfies

$$2\Big(1 - \tfrac{d^2}{h^2}\Big)\exp\Big(-\tfrac{d^2}{h^2}\Big)(\mathbf{z}_n^t - \mathbf{z}_{n'}^t) = \nabla_{\mathbf{z}_n^t}\,\psi(d), \qquad \psi(d) = d^2\exp\Big(-\tfrac{d^2}{h^2}\Big),$$

so $\mathbf{F}(\mathbf{z}_n^t) = \nabla_{\mathbf{z}_n^t}\sum_{n'\neq n}\psi(d_{nn'})$, and the force is an ascent direction for the pairwise potential with the bandwidth held fixed at that step. ψ increases with d up to its maximum at $d = h$ and decreases beyond it. The Gaussian-kernel repulsion used in particle guidance (Corso et al., 2024) and in SVGD-style updates pushes pairs apart at every distance, with a force that fades as they separate. ψ instead moves pairs toward a common spacing equal to the group's own median distance: pairs closer than the median are pushed apart, and farther pairs are pulled back slightly, adjusting the spacing locally.

Bandwidth in high dimension. Because h is the median pairwise distance, a typical pair sits near the balance point $d = h$, where its weight $2(1 - d^2/h^2)e^{-d^2/h^2}$ is close to zero. In high dimension pairwise distances concentrate, so near-equal distances can make the force small; its pairwise terms respond to deviations from the median spacing, pushing apart latents that are unusually close and pulling in those that are unusually far. The pairwise forces are symmetric and sum to zero, so the guidance step leaves the group centroid unchanged. The median bandwidth makes the guidance scale-equivariant: rescaling all latents by a positive constant rescales h and $\mathbf{F}$ by that constant while leaving each dimensionless pair weight unchanged.

Scale and schedule. $\mathbf{F}$ is not normalized; its magnitude depends on the pairwise distances and their weights. The schedule $\gamma_t = \gamma_{\max} t/K$ decreases linearly over the sampling steps $t = K, \ldots, 1$, from $\gamma_{\max}$ to $\gamma_{\max}/K$. Guidance is used only for training rollouts ($\gamma_t = 0$ at evaluation).

D Experimental Details

D.1 Code Generation

Data Filtering Pipeline. We construct a unified RL training corpus from two public coding datasets (KodCode-V1 (Xu et al., 2025) and AceCode-87K (Zeng et al., 2025a)) by converting each record into a common format `(prompt, test_code/test_cases, solution, metadata)`. The pipeline then applies four sequential filtering stages to improve supervision quality and execution reliability. **(Phase 1: Quality filtering)** We retain only problems with at least 5 executable unit tests (after dataset-specific parsing of assertions or stdin/stdout style tests), ensuring each instance provides sufficient verifiable signal. **(Phase 2: Semantic deduplication)** To reduce redundancy, we embed all prompts using a sentence-transformer and perform greedy cosine-similarity filtering with threshold 0.85, keeping the first instance in each cluster. **(Phase 3: Difficulty calibration)** To remove trivial tasks, we sample 8 independent solutions per prompt from `Qwen2.5-Coder-7B-Instruct` (via vLLM) and execute each generation against the associated tests under a strict timeout; problems where *all* sampled solutions pass all tests are excluded. **(Phase 4: Ground-truth validation and sanitization)** For the remaining examples, we (i) automatically add missing standard imports by pattern matching (e.g., `typing`/`math`/`heapq`) and optionally add a function alias to match the expected entry point inferred from tests, (ii) filter any solution or test containing unsafe or environment-dependent operations (file I/O, network calls, subprocess/system operations, interactive input, pickling) via regex rules, (iii) filter any example importing unavailable packages, and (iv) verify that the provided ground-truth solution passes *all* extracted test cases using isolated subprocess execution with timeouts. The final output is a set of non-trivial, deduplicated, execution-safe programming problems with verified ground-truth solutions and reliable unit-test reward signals for RL training.

Implementation Details. During RL training, each problem is paired with its associated unit tests, which define a scalar reward equal to the pass rate of unit tests executed by the generated solution. For each prompt, we sample N latent reasoning trajectories per update and M answers per latent,

Table 6: Training and sampling hyperparameters for LaDi-RL (Flow-GRPO-Fast with CPS sampling). The SDE window range is given in sampling-step indices (0 = first step from noise). Each 2560-dim latent token is mapped to the backbone hidden size by a linear projection. All hyperparameters were selected on a small held-out validation set, not on the test benchmarks.

Model & Optimization		**RL Training**	
Latent representation	64 tokens, 2560 dim	KL regularization	Disabled
Optimizer	AdamW (β_1=0.9, β_2=0.999)	Loss weights	λ_{diff}=10, λ_{text}=1 ($\alpha\approx$0.91)
Learning rate	1×10^{-5}	CPS noise level (η)	0.8
Max text length	768 tokens	Denoising steps	10 (train) / 30 (eval)
Text sampling	T=1.0, top-p=0.98	Shared initial noise	Disabled
Diffusion sampling	CPS	Latent clip range	ϵ_z=10^{-5}
Classifier-free guidance	Disabled	Text clip range	ϵ_x^l=0.2, ϵ_x^h=0.28
$\gamma_{\max}$ (diversity guidance)	0.8	SDE window size / range	2 / $(0, 5)$
		Rollout size	N=4, M=4 (16 answers)

and compute group-relative advantages following the GRPO formulation. Latent diffusion transitions are treated as stochastic policy steps, and text generation is performed only at the denoised latents. All diffusion and text-policy hyperparameters, optimization settings, and sampling configurations follow Table 6.

D.2 Math Reasoning

Baseline Details. For math reasoning, we compare LaDi-RL against supervised fine-tuned (SFT) and reinforcement learning baselines operating directly in token space. All baselines use the same pretrained backbone and are matched in total training steps, rollout budget, and reward signal. Rewards are defined based on exact answer correctness under the benchmark-specific evaluation protocols; the reward checks only the final answer, without supervision of intermediate reasoning.

Implementation Details. All math results in Table 1, including the baselines taken from Tang et al. (2026), use a maximum response length of 4,096 tokens. This budget is much shorter than the 32,768 tokens used to report DeepSeek-R1-Distill-Qwen-7B in Guo et al. (2025a) (e.g., 55.5 on AIME 2024 and 92.8 on MATH-500), which explains why the absolute numbers of the base model are lower in our setting; all methods share the same budget, so the comparison remains controlled. Math reasoning experiments reuse the same latent diffusion architecture, Flow-GRPO optimization scheme, and hyperparameter configuration as in the code-generation setting (Table 6). Rewards are computed by comparing the final predicted answer against the ground-truth solution, following benchmark-specific normalization and answer-matching rules. Group-relative advantages are computed across sampled trajectories to guide both latent diffusion updates and text-policy refinement.

E Additional Results

E.1 Efficiency Analysis

As shown in Table 7, LaDi-RL is more computationally efficient than standard GRPO at inference time. Standard GRPO relies on long CoTs, generating an average of 3,447 text tokens per sample. In contrast, LaDi-RL represents reasoning using only 64 latent tokens, which on average encode a short CoT of 385 text tokens, corresponding to an effective compression rate of approximately $6.0\times$. Then, the denoised latent CoTs are conditioned to generate an average of 359 answer tokens. This latent compression reduces both rollout overhead and autoregressive decoding cost, yielding a 43% reduction in end-to-end wall-clock inference time (from 6.5 s to 3.7 s). Both measurements include all latent denoising steps (30 at evaluation) and use a single shared 8B backbone.

E.2 Training Cost and Scaling

LaDi-RL adds no meaningful training overhead over a standard RL pipeline (Table 8). RL pipelines for reasoning often begin with a domain-specific SFT (mid-training) stage on chain-of-

Method	# Generated tokens	Wall-clock time
Standard GRPO	3447	6.5 s
LaDi-RL	64 (latent) + 359 (text)	3.7 s

Table 7: Inference efficiency comparison on code generation. We report the number of generated tokens and the wall-clock time per sample.

thought data; the LaDiR cold start simply takes the place of this stage. It is trained on the same 1.4M CoTs and costs 496 GPU-hours (VAE and latent flow matching), compared with 480 GPU-hours for standard SFT. The RL stage is also slightly cheaper for LaDi-RL, because it decodes much shorter texts: 10 vs. 12 minutes per step and about 40 vs. 42 hours to convergence. Over the whole pipeline, LaDi-RL and SFT followed by GRPO therefore use the same compute (816 GPU-hours each). Measured from each method's own starting point, the RL stage adds $+12.1$ for LaDi-RL (LaDiR $64.9 \rightarrow 77.0$) and $+5.3$ for GRPO (SFT $61.2 \rightarrow 66.5$) on code, and $+6.0$ ($38.0 \rightarrow 44.0$) and $+0.9$ ($36.7 \rightarrow 37.5$) on math. Finally, the gain holds at larger scale (Qwen3-32B, dense: 86.5 for LaDi-RL vs. 75.6 for GRPO), and a larger rollout budget helps (2×2 and 4×4 reach 65.5 and 77.0). In short, the LaDiR cold start replaces the usual SFT stage rather than adding to it, and the whole LaDi-RL pipeline costs no more than SFT followed by GRPO.

	SFT $\rightarrow$ GRPO	LaDi-RL
SFT / initialization data	1.4M CoTs	1.4M CoTs
SFT / initialization cost (GPU-h)	480	96 (VAE) + 400 (LaDiR) = 496
RL data	24K prompts	24K prompts
RL time per step	12 min	10 min
RL time to convergence	42 h	$\sim$40 h
Peak memory per GPU	74 GB	76 GB
Text tokens per prompt (16 rollouts)	55,152	5,744 (+256 latent)
Total GPU-h (to RL convergence)	816	816
Code avg. pass@1	66.5	77.0

Table 8: Training pipelines of token-level RL (standard SFT followed by GRPO) and LaDi-RL on code (Qwen3-8B, $8\times$H100). RL GPU-hours are time to convergence $\times$ 8 GPUs.

E.3 Latent Reward Estimator at a Fixed Rollout Budget

The 4×4 (random) control in Table 5 keeps all 16 generated answers and both latent and text losses enabled. It changes only the reward used to construct the latent advantage:

$$\widehat{R}_n^{\text{mean}} = \frac{1}{4}\sum_{m=1}^{4} R(x_{n,m}), \qquad \widehat{R}_n^{\text{random}} = R(x_{n,J_n}), \quad J_n \sim \text{Uniform}\{1,2,3,4\}.$$

Select J_n independently of the rewards once per latent rollout and hold it fixed during updates on that batch. Standardize the four latent rewards using the same rule in both variants. The text loss uses all four answers per latent with the original within-latent advantages in both cases. Initialization, guidance, sampler, training prompts, rollout budget, optimizer, loss weights, and training steps are held fixed. Evaluation follows Table 5, averaging pass@1 over the same five code benchmarks with guidance disabled.

For a fixed decoder and latent, independent answer samples make both estimators unbiased for the conditional expected reward; averaging four reduces the conditional reward variance by a factor of four. The mean therefore provides a more stable estimate of latent quality by reducing variation from answer decoding. With both losses active and the rollout budget fixed, mean aggregation reaches 77.0 versus 42.9 pass@1 for random (Table 5). This comparison supports averaging over decoder outcomes to assign credit to latents, separating latent quality from the success or failure of a single sampled answer.

E.4 Comparison with Diversity-Oriented Baselines

Table 9 compares LaDi-RL with methods that directly target diversity collapse. All methods use the Qwen3-8B backbone, the five code benchmarks, the same rollout and reward budget, the same verifier, and the same decoding protocol. *Token-level diversity RL* (DSDR, DIVER, DARLING, Rewarding the Unlikely) improves over vanilla GRPO, by up to $+5.3$ pass@1 (DIVER) and $+4.1$ pass@128 (Rewarding the Unlikely), so rewarding diversity helps even when exploration stays at the token level; LaDi-RL still exceeds the best of these methods by $+5.2$ pass@1, $+3.6$ pass@128, and $+0.09$ Sem. *Pass@k optimization* (PKPO) reaches the highest pass@128 among the baselines (87.7) but only 70.1 pass@1, indicating that optimizing set-level coverage trades off single-sample accuracy; LaDi-RL improves both ($+6.9$ pass@1, $+2.7$ pass@128). *Diversity-preserving SFT* (GEM) retains coverage (pass@128 84.6) but barely changes pass@1 ($+0.7$ over GRPO), so preserving diversity before RL does not by itself translate into accuracy. *Continuous-token and hidden-state exploration* give mixed results: SofT-GRPO falls below vanilla GRPO in pass@1, while ReLaX is the strongest baseline in pass@1; LaDi-RL improves over ReLaX by $+5.1$ pass@1, $+4.1$ pass@128, and $+0.14$ Sem. Finally, *latent trajectory exploration* is effective only with a diffusion policy: the one-step VAE sampler (LaVAE-RL) collapses to 34.9 pass@1. Overall, LaDi-RL is best on all three metrics, which indicates that its advantage comes from exploring whole reasoning trajectories with a diffusion policy, rather than from adding diversity objectives or regularizers on top of token-level sampling.

Method	Diversity mechanism	pass@1	pass@128	Sem.
Token-level exploration (discrete tokens)				
Vanilla GRPO	–	66.5	82.7	.18
GRPO w/ Entropy Adv.	entropy-based advantage	68.5	83.6	.27
DSDR (Wan et al., 2026)	global + local diversity regularization	71.1	84.9	.36
DIVER (Hu et al., 2025)	sequence-level diversity reward	71.8	86.1	.40
DARLING (Li et al., 2025d)	joint quality–diversity reward	69.7	85.3	.41
Rewarding the Unlikely (He et al., 2025)	reward for rare correct solutions	71.2	86.8	.43
PKPO (Walder & Karkhanis, 2025)	pass@k objective	70.1	87.7	.45
GEM (Li et al., 2025f) (SFT)	diversity-preserving SFT objective	67.2	84.6	.39
Continuous token / hidden-state exploration				
SofT-GRPO (Zheng et al., 2025b)	noise-perturbed soft tokens	65.4	83.4	.22
ReLaX (Zhang et al., 2025c)	hidden-state exploration regularizer	71.9	86.3	.38
Latent trajectory exploration				
LaVAE-RL	one-step VAE latent sampler	34.9	44.6	.40
LaDi-RL	multi-step latent diffusion	**77.0**	**90.4**	**.52**

Table 9: Comparison with diversity-oriented baselines on code (five-benchmark average). Rows are grouped by exploration space; the second column gives each method's diversity mechanism.

E.5 Semantic-Diversity Protocol

Judge. We use `gpt-5-2025-08-07` (default temperature) to cluster 16 anonymized, shuffled solutions for each of 500 problems sampled evenly from the five code benchmarks. For every method, the judge sees only the final code of each solution, not the chain of thought or the latent. Two solutions share a cluster when they use the same core algorithm or derivation; the score of a problem is $\#\text{clusters}/16$, averaged over problems. The prompt was calibrated on a separate 30-problem pilot labeled by humans. GPT-5 is used only for this metric; training rewards, pass@1, and pass@k are computed with benchmark answers or executable tests. The judge prompt is:

```
You are evaluating diversity of reasoning strategies.  Given
one problem and 16 anonymous candidate solutions, cluster
the solutions by their underlying algorithm or reasoning
approach.  Ignore wording, formatting, variable names, minor
implementation choices, and verbosity.  Place two solutions
in different clusters only when their core algorithm,
decomposition, or derivation differs.  Do not use the method
identity or presentation style.  Return valid JSON with:
(1) one cluster id for each solution, (2) a short strategy
```

```
label for each cluster, and (3) a one-sentence justification
for each separation.
```

Validation. Two annotators clustered 200 problems, and a third resolved disagreements. GPT-5 agrees with the adjudicated labels on 88% of solution pairs (Cohen's κ=0.76).

Correct-only and judge-independent metrics. To exclude diverse failures, we keep only test-passing solutions before clustering and still divide by 16, so a method with a single correct solution cannot score high. We also report canonicalized AST/control-flow clusters and `all-mpnet-base-v2` embedding clusters (cosine distance 0.20) on correct solutions, and Distinct-4. All metrics show the same ordering (Table 10).

Method	GPT-5	Human	GPT-5 (correct)	AST (correct)	Embed. (correct)	Distinct-4
Vanilla GRPO	.18	.20	.14	.17	.21	.30
LaVAE-RL	.40	–	.06	–	–	–
LaDi-RL	.52	.49	.36	.45	.47	.42

Table 10: Semantic diversity on code under different measurements.

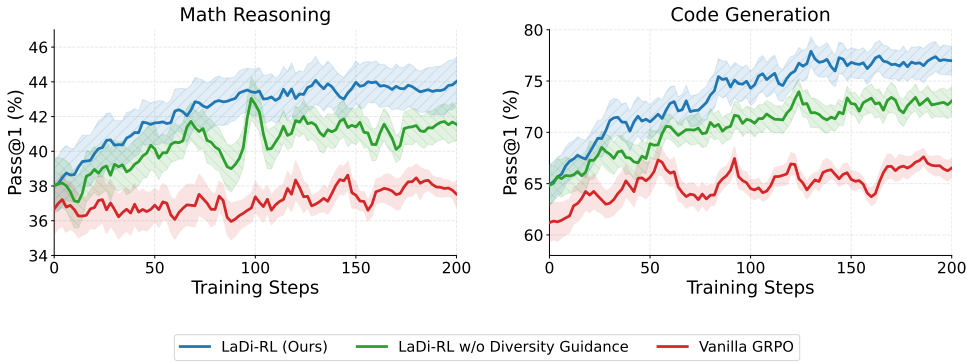

Figure 5: Pass@1 during RL training on math reasoning (left) and code generation (right) for LaDi-RL, LaDi-RL without diversity guidance, and vanilla GRPO. Code curves average the five code benchmarks. On both tasks, GRPO starts from the SFT checkpoint and both LaDi-RL variants from the LaDiR cold start. Shaded regions show one standard deviation.

E.6 Diversity Analysis

Figure 5 tracks pass@1 over 200 RL steps. LaDi-RL keeps improving on both tasks and ends highest. Vanilla GRPO gains only a few points and stays noisy on math. On code, GRPO gains +5.3 points over its SFT starting point, while LaDi-RL gains +12.1 over LaDiR; on math, the corresponding gains are +0.9 and +6.0. Removing diversity guidance leaves the curve in between: it improves at first but levels off below full LaDi-RL on both tasks. The latent diffusion policy therefore drives most of the gain over GRPO, and diversity guidance keeps it improving later in training.

E.7 Loss Weight

We select α on the validation set and report test pass@1 here for reference. We study the mixing weight α in $\mathcal{L}_{\mathrm{RL}} = \alpha\mathcal{L}_{\mathrm{latent}}^{\mathrm{clip}(\epsilon_z)} + (1-\alpha)\,\mathcal{L}_{\mathrm{text}}^{\mathrm{clip}(\epsilon_x^l,\epsilon_x^h)}$. Figure 6 shows that performance is highly sensitive to α, peaking at $\alpha = 0.90$. When α is smaller (e.g., 0.3 or 0.5), optimization is dominated by the text policy and pass@1 drops substantially. This is expected because the two objectives have very different gradient scales. The latent objective uses the dimension-averaged score of Eq. (24), whereas the text objective uses token log-probabilities; their gradient scales can differ substantially, so α controls their relative contribution to the shared-parameter update. Increasing α beyond the optimum (e.g., $\alpha = 0.99$) reduces pass@1 by several points, suggesting that a small but non-zero text-policy weight remains beneficial for aligning latent exploration with the final decoded solutions.

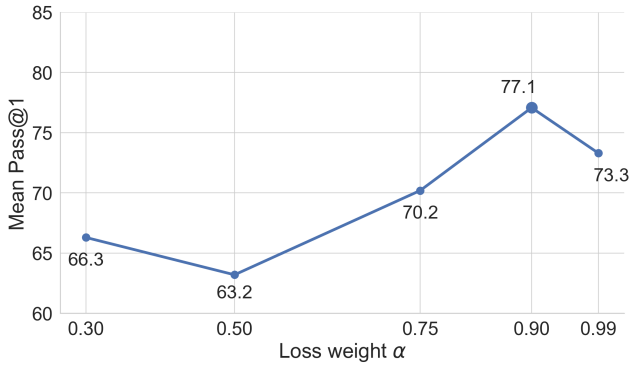

Figure 6: Effect of the loss weight of latent policy and text policy α on code generation pass@1.

F Qualitative Analysis

We use two representative examples to illustrate a key property of LaDi-RL: *a single problem can give rise to multiple correct, structurally different solutions*. Rather than enumerating many generations or comparing superficial implementation changes, we focus on cases where the alternative solutions correspond to clearly different reasoning or program structures.

F.1 Mathematical Reasoning: 2024 AIME I, Problem 7

Consider Problem 7 from the 2024 AIME I:

2024 AIME I, Problem 7

Find the greatest possible real part of

$$(75+117i)z+\frac{96+144i}{z},$$

where z is a complex number with $|z|=4$.

LaDi-RL produces multiple correct routes to the same answer. Below we show three representative solution strategies.

LaDi-RL Solution 1: Cartesian coordinates and Cauchy–Schwarz

Let $z=a+bi$, where $a^2+b^2=16$. Since

$$\frac{1}{z}=\frac{\overline{z}}{|z|^2}=\frac{\overline{z}}{16},$$

the expression becomes

$$(75+117i)z+(6+9i)\overline{z}.$$

Its real part is

$$(75a-117b)+(6a+9b)=81a-108b.$$

Therefore, by Cauchy–Schwarz,

$$81a-108b\le\sqrt{81^2+108^2}\sqrt{a^2+b^2}=135\cdot 4=540.$$

Equality is attainable, so the answer is $\boxed{540}$.

LaDi-RL Solution 2: Polar parameterization

Write $z = 4e^{i\theta}$. Then

$$\mathrm{Re}\left((75+117i)z + \frac{96+144i}{z}\right) = 324\cos\theta - 432\sin\theta.$$

Hence

$$324\cos\theta - 432\sin\theta \le \sqrt{324^2 + 432^2} = 540,$$

with equality for a suitable θ. Thus the maximum is again $\boxed{540}$.

LaDi-RL Solution 3: Complex-number norm argument

Using $1/z = \overline{z}/16$,

$$\mathrm{Re}((75+117i)z + (6+9i)\overline{z}) = \mathrm{Re}((81+108i)z)\,.$$

For any complex number w, $\mathrm{Re}(w) \le |w|$. Therefore

$$\mathrm{Re}((81+108i)z) \le |81+108i|\,|z| = 135 \cdot 4 = 540.$$

The bound can be attained by choosing the phase of z so that $(81+108i)z$ is positive real. Hence $\boxed{540}$.

These solutions use genuinely different representations of the same problem: Cartesian optimization, trigonometric parameterization, and a direct complex-geometric argument. The example illustrates that LaDi-RL is not restricted to a single reasoning trajectory even when all generations arrive at the same correct answer.

F.2 Code Generation: MBPP 162

We next consider MBPP Problem 162:

MBPP 162

Write a function to calculate the sum of the positive integers in

$$n + (n-2) + (n-4) + \cdots$$

until the terms become non-positive.

Tests:

- `assert sum_series(6) == 12`
- `assert sum_series(10) == 30`
- `assert sum_series(9) == 25`

LaDi-RL generates several correct implementations with different program structures.

LaDi-RL Solution 1: Explicit iteration

```
def sum_series(n):
    total = 0
    while n > 0:
        total += n
        n -= 2
    return total
```

LaDi-RL Solution 2: Recursion

```
def sum_series(n):
    if n <= 0:
        return 0
    return n + sum_series(n - 2)
```

LaDi-RL Solution 3: Direct sequence construction

```
def sum_series(n):
    return sum(range(n, 0, -2))
```

LaDi-RL Solution 4: Closed form

For even n,

$$2 + 4 + \cdots + n = \frac{n(n+2)}{4},$$

while for odd n,

$$1 + 3 + \cdots + n = \left(\frac{n+1}{2}\right)^2.$$

This gives:

```
def sum_series(n):
    if n <= 0:
        return 0
    if n % 2 == 0:
        return n * (n + 2) // 4
    return ((n + 1) // 2) ** 2
```

All four implementations satisfy the benchmark tests. More importantly, they expose different structures for solving the task: explicit stateful iteration, recursive decomposition, construction through a language primitive, and an $O(1)$ mathematical formula. The last approach in particular does not enumerate the series at all.

F.3 Takeaway

These examples demonstrate the diverse solution-space coverage of our method. On both mathematical reasoning and code generation, LaDi-RL can produce multiple correct solutions that differ in their underlying semantic or computational structure. This qualitative behavior complements the aggregate diversity results reported in the main experiments.