LaDex: Latent State Divergence Incentivizes Self-Exploration in LLM Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become an effective paradigm for enhancing the reasoning abilities of large language models (LLMs). However, vanilla group-based RLVR relies on sparse final-answer rewards, offering limited exploration guidance. Existing exploration algorithms primarily focus on token-level signals, which may reward superficial variation rather than guide the discovery of genuinely diverse and promising reasoning paths. With the observation that trajectories following different reasoning paths show greater divergence in latent space, we propose LaDex, a novel RL method that incentivizes LLM self-exploration through latent-space divergence. For a given prompt, LaDex first generates a group of reasoning trajectories, then segments each full reasoning trajectory into discrete stages and extracts their latent states. Within each reasoning stage, LaDex computes latent divergence by comparing latent states from different trajectories in the group. To balance exploration and exploitation, LaDex introduces a Latent Classifier Network (LCN) that monitors meaningful exploration within each reasoning stage and adjusts the latent-divergence signals accordingly. These signals are then aggregated into intrinsic rewards to guide policy optimization toward effective self-exploration. Experiments on diverse reasoning benchmarks demonstrate that LaDex significantly improves exploration efficiency and consistently outperforms strong baselines. Our code is available at `https://anonymous.4open.science/r/LLMRL-CDE2/`

1 Introduction

Reinforcement learning (RL) has shown great potential for improving the reasoning capabilities of large language models (LLMs) (Guo et al., 2025; Yu et al., 2025). This is especially important in group-based reinforcement learning with verifiable rewards (RLVR), where solving complex problems often requires long-horizon generation before sparse outcome feedback is provided through final-answer verification. Under this setting, effective self-exploration is essential for enabling the model to move beyond dominant reasoning paths and discover better ones (Lu et al., 2025; Zhang et al., 2025). Existing exploration methods primarily derive signals from decoded tokens, ranging from token-level uncertainty to diversity. However, token-level diversity may reflect surface variation rather than genuinely distinct reasoning paths. By contrast, latent states (the hidden vectors before the language modeling head) capture how reasoning evolves internally, offering a path-sensitive signal for effective self-exploration over long reasoning horizons during policy optimization.

Despite extensive studies of exploration in traditional RL, exploration in LLM-based RLVR remains challenging. First, each reasoning trajectory contains a long sequence of discrete tokens, leading to a combinatorially large search space. Token-level exploration methods (e.g., tree-like branching (Zhao et al., 2026; Dong et al., 2026; Li et al., 2025)) struggle with inefficiency. Second, complex reasoning tasks are inherently long-horizon, whereas outcome rewards are only available after final-answer verification. Such delayed feedback provides little guidance on which intermediate reasoning stages are worth exploring (Guo et al., 2025; Yu et al., 2025; Zhang et al., 2026b). Third, the exploration–exploitation tradeoff is particularly difficult to balance in RLVR. Excessive exploration may introduce noisy or unproductive trajectories, whereas insufficient exploration may cause the policy to collapse to a few dominant reasoning paths (Cheng et al., 2025; Dai et al., 2026; Zhang et al., 2026b). These challenges call for exploration signals that capture differences between reasoning paths while remaining compatible with outcome-driven optimization.

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.

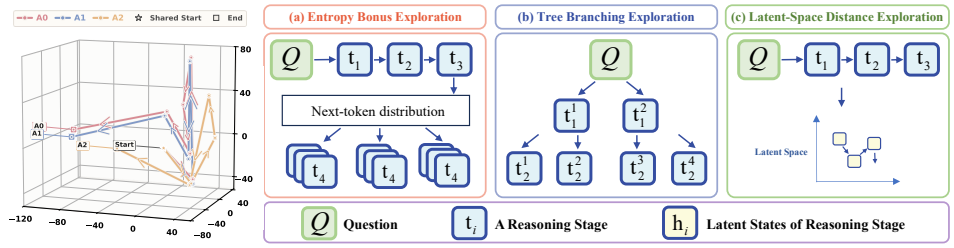

Figure 1: Representative exploration paradigms and empirical findings. **Left:** PCA visualization of three sampled trajectories for the same MATH question. Trajectories A0 and A1 follow the same reasoning path and stay close, while A2 follows a different path and progressively diverges across reasoning stages. **Right:** Entropy and branching-based methods rely on token-level uncertainty, whereas LaDex guides exploration through internal latent state divergence signals.

Existing work has explored several directions for improving exploration in reasoning RL, each focusing on a different aspect of the problem. Vanilla group-based RLVR methods provide strong outcome supervision by optimizing rollouts with final-answer rewards, thereby promoting reliable exploitation of verifiable signals (Guo et al., 2025; Yu et al., 2025). As illustrated in Figure 1 (a), entropy-based methods encourage broader generation by increasing token-level randomness through temperature scaling, entropy bonuses, or entropy regularization (Cheng et al., 2025). Branching-based methods further expand the search process by introducing branches at positions selected by generation entropy or semantic entropy, as shown in Figure 1 (b) (Dong et al., 2026; Li et al., 2025; Zhao et al., 2026). Another line of work introduces trajectory-level diversity rewards, but typically encourages exploration only on correct trajectories, overlooking the potential value of incorrect trajectories in the exploration–exploitation trade-off (Chen et al., 2026; Hu et al., 2026; Zhang et al., 2026a). Although token-level signals improve exploration, they may capture surface variation rather than genuinely different reasoning paths. Effective exploration therefore requires trajectory-level information that reflects how reasoning evolves internally, which latent states naturally provide.

To investigate whether latent states reflect differences in reasoning paths, we analyze trajectories generated by Qwen-series models on the MATH dataset (Hendrycks et al., 2021). For each question, we sample eight trajectories and manually assign those following the same reasoning path to the same category and those following different paths to separate categories. We extract their latent states before the LM head and visualize them using PCA. As shown in the left panel of Figure 1, trajectories following the same reasoning path remain close in latent space, whereas trajectories following different paths diverge across reasoning stages. This indicates that latent states capture information related to the choice of reasoning path, providing a useful signal for trajectory-level exploration beyond simple token-level variation. Further analyses are provided in Appendix A.3.

Building on this observation, we introduce LaDex, a novel RL method that incentivizes LLM self-exploration by leveraging latent divergence signals, referred to as stage-wise distance, as illustrated in Figure 1 (c). By quantifying how reasoning trajectories diverge in latent space at each stage, LaDex captures nuanced exploration signals beyond token-level signals. Concretely, LaDex segments each reasoning trajectory into discrete reasoning stages and extracts the corresponding latent states. For each reasoning stage, it computes the stage-wise distance by comparing latent states at the same stage between different trajectories in the group. As correct and incorrect reasoning trajectories become increasingly separable in latent space (Du et al., 2026), LaDex introduces a Latent Classifier Network (LCN) to balance exploration and exploitation. The LCN predicts a modulation weight at each reasoning stage and adjusts the corresponding stage-wise distance to promote meaningful exploration. The adjusted stage-wise distances are then aggregated along each trajectory to form a trajectory-level intrinsic reward. This reward encourages exploration of reasoning paths that are both diverse in latent space and promising. Extensive experiments on mathematical reasoning, STEM QA, and multi-turn WebShop demonstrate that LaDex consistently outperforms strong baselines across model scales and tasks. Notably, relative to GRPO, LaDex improves average mathematical pass@8 by over 11% at both scales, mean@8 by 21.18% and 13.64% on 4B and 8B, respectively, and WebShop success by 53.36%. Our contributions cover three complementary aspects of latent state-guided exploration, spanning analysis, design, and evaluation:

- **Understanding Latent-Space Signals for Exploration.** We uncover that LLM trajectories following the same reasoning path remain close in latent space, whereas those following different paths progressively diverge. This finding establishes latent-state divergence as an informative signal for LLM self-exploration beyond decoded token-level cues.
- **Latent State-Guided Self-Exploration Algorithm.** We propose LaDex, a latent state-guided algorithm that transforms latent divergence into intrinsic rewards. An LCN monitors the meaningfulness of exploration at each reasoning stage and adaptively balances exploration and exploitation, thereby driving effective self-exploration during RL training.
- **Evaluation on Diverse Benchmarks.** Extensive experiments on in-domain mathematical reasoning and out-of-domain QA tasks demonstrate that LaDex outperforms strong baselines and achieves strong reasoning performance. Additional experiments on the multi-turn WebShop task highlight its potential to generalize to interactive agentic environments.

2 Related Works

2.1 Reinforcement Learning with Verifiable Rewards

Reinforcement learning with verifiable rewards (RLVR) has been widely used to improve the reasoning abilities of large language models (Guo et al., 2025; Lambert et al., 2025; Yu et al., 2025). While PPO is a foundational method for RL post-training, it typically relies on a learned value model for advantage estimation, introducing additional optimization and computational overhead (Schulman et al., 2017). Recent group-based methods improve efficiency by estimating advantages from multiple rollouts sampled for the same prompt. GRPO (Guo et al., 2025) avoids a separate critic through group-relative comparison, and this formulation has been further extended by DAPO (Yu et al., 2025), Dr. GRPO (Liu et al., 2025), GSPO (Zheng et al., 2025), CISPO (MiniMax et al., 2025), and RLOO (Ahmadian et al., 2024). Despite these advances, such methods primarily rely on sparse final-answer rewards, which provide limited guidance for the policy to explore genuinely different reasoning paths during generation beyond the supervision supplied by final-answer correctness.

2.2 Exploration Methods in RLVR

Exploration in reinforcement learning requires balancing known rewarding behaviors with uncertain alternatives. Traditional strategies include entropy-driven exploration, optimism under uncertainty, posterior sampling, intrinsic motivation, curiosity, and count-based bonuses (Bellemare et al., 2016; Haarnoja et al., 2018; Pathak et al., 2017; Osband et al., 2013). Recent RLVR methods adapt these ideas through token-level uncertainty, intrinsic rewards, branching, and diversity-oriented objectives. Entropy-based methods promote exploration by increasing token-level uncertainty (Cheng et al., 2025). MERCI (Zhang et al., 2026b) and CDE (Dai et al., 2026) introduce novelty or uncertainty-based intrinsic rewards. ARPO (Dong et al., 2026), ROSE (Zhao et al., 2026), and TreePO (Li et al., 2025) expand the search space by branching from uncertain positions. DIVER (Hu et al., 2026) and DQO (Chen et al., 2026) encourage diverse reasoning trajectories, while ReLaX (Zhang et al., 2026a) regularizes latent dynamics in multimodal RLVR. While effective, these methods mostly define exploration through token-level signals, which may fail to capture differences between reasoning paths and cannot reliably guide the discovery of alternative, promising trajectories.

2.3 Latent Space Reasoning for LLMs

Recent studies suggest that LLM latent states encode rich reasoning information beyond explicit tokens, motivating a growing line of work on continuous or implicit latent-space reasoning (Gozeten et al., 2026; Zhu et al., 2026). Coconut (Hao et al., 2025) performs reasoning through continuous latent states, Methods such as Think-in-Silence (Tan et al., 2026a), SoftCoT (Xu et al., 2025), CODI (Shen et al., 2025), Token Assorted (Su et al., 2025), SIM-CoT (Wei et al., 2026), and LED (Tan et al., 2026b) investigate latent compression, soft reasoning, latent-text mixtures, distillation-based implicit reasoning, and inference-time latent exploration. LTO (Du et al., 2026) demonstrates that correct and incorrect reasoning processes become increasingly distinguishable in latent space across reasoning stages. These works highlight the importance of latent states for reasoning, and our work leverages latent trajectories not only to characterize reasoning path-level

differences but also to actively guide exploration in group-based RLVR, promoting more effective LLM self-exploration and encouraging the discovery of diverse, high-quality reasoning paths.

3 PRELIMINARIES

Group-Based RLVR. Reinforcement learning with verifiable rewards (RLVR) optimizes LLM reasoning using outcome rewards from final-answer verification (Guo et al., 2025; Lambert et al., 2025; Yu et al., 2025). Given a question Q, group-based RLVR samples G rollouts $\{o_i\}_{i=1}^{G}$ from policy π_θ and assigns each rollout a reward r_i. GRPO estimates the relative quality of each rollout by comparing its reward with the mean and standard deviation of rewards within the sampled group:

$$A_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})}. \tag{1}$$

We optimize the policy using the resulting advantages in a PPO-style clipped objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min(\rho_{i,t}A_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)A_i) - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right], \tag{2}$$

where $\rho_{i,t} = \pi_\theta(o_{i,t} \mid Q, o_{i,<t})/\pi_{\theta_{\text{old}}}(o_{i,t} \mid Q, o_{i,<t})$, ϵ is the clipping threshold, and β controls the KL penalty. While GRPO and its variants (Liu et al., 2025; Yu et al., 2025) provide effective group-based advantage estimation, we aim to further improve exploration within group-based RLVR.

The Exploration Problem in RLVR. In group-based RLVR, effective self-exploration should enable the policy to move beyond dominant reasoning paths and discover alternative reasoning paths that are both diverse and promising. This is especially important in long-horizon reasoning, where discrete token trajectories create a large search space, sparse final-answer rewards provide limited exploration guidance, and the policy must avoid both noisy deviations and collapse to dominant reasoning paths. To address this challenge, we move beyond token-level exploration and leverage internal latent state signals that capture path-level divergence, guiding the policy to perform more effective LLM self-exploration and encouraging the discovery of novel, high-quality reasoning paths.

4 METHODS

Motivation. In RLVR, sparse final-answer rewards provide little intermediate guidance for long-horizon exploration. Existing methods encourage diversity through decoded outputs, ranging from token-level uncertainty to token-level differences. However, token diversity may reflect surface variation rather than genuinely distinct reasoning paths. Since latent states capture how reasoning evolves internally, their divergence provides a path-sensitive signal for exploration. Based on this insight, LaDex leverages stage-wise latent divergence to drive effective LLM self-exploration.

Overview. As shown in Figure 2, LaDex contains three main components. Given a question, it samples a group of trajectories, segments each trajectory into stages, and extracts their latent states. Next, at each reasoning stage, LaDex compares latent states at the same stage between different trajectories within the group to compute latent-divergence signals, which are then adjusted by a Latent Classifier Network (LCN). Finally, the adjusted divergence signals are aggregated as trajectory-level intrinsic rewards, which are combined with the outcome rewards for policy optimization.

4.1 GENERATION & LATENT STATES EXTRACTION

Given a question Q, the policy π_θ samples G trajectories $\{o_i\}_{i=1}^{G}$. Each trajectory is segmented into reasoning stages using double newline markers (`\n\n`), following existing work (Liu et al., 2026). A reasoning stage denotes the set of aligned segments at the same position between different trajectories within the group. For each stage, we mean-pool the latent states of its tokens to obtain a stage-level representation (Reimers & Gurevych, 2019; Wang et al., 2024):

$$h_{i,p} = \frac{1}{K_{i,p}}\sum_{k=1}^{K_{i,p}} h_{i,p}^{(k)} \in \mathbb{R}^d, \tag{3}$$

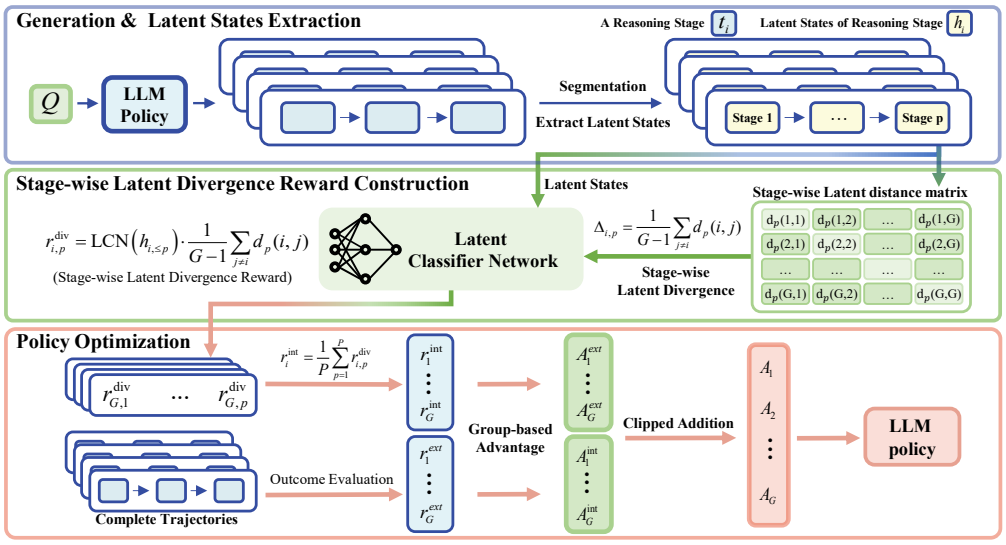

Figure 2: Overview of LaDex. For each question, LaDex generates trajectories, segments them into reasoning stages, and extracts latent states. At each stage, latent states in different trajectories are compared to compute latent-divergence, which are adjusted by a Latent Classifier Network (LCN) and aggregated into intrinsic rewards, combined with outcome reward to optimize the LLM policy.

where $K_{i,p}$ is the number of tokens in stage p of trajectory i, and $h_{i,p}^{(k)}$ is the latent state of the k-th token. This produces the sequence of stage-level latent states $H_i = \{h_{i,p}\}_p$ for trajectory o_i, which serves as the basis for stage-wise latent-divergence estimation. After alignment, each $h_{i,p}$ is compared only with $h_{j,p}$ from other trajectories o_j at the same reasoning stage p.

4.2 Stage-Wise Distance Estimation of Latent States

Because trajectories may contain different numbers of stages, we first apply temporal resampling (Duan et al., 2022; Chen et al., 2021b) to align all trajectories to $P = 32$ stage-level latent states. LaDex constructs latent-divergence signals by comparing the latent states at the same position in different trajectories, which together define the reasoning stage. Each $h_{i,p}$ (the p-th stage within a trajectory) is compared with $h_{j,p}$ from other trajectories o_j within the same reasoning stage. At the p-th reasoning stage, the distance between different trajectories, o_i and o_j, is computed based on the latent states at this stage as:

$$d_p(i,j) = d(h_{i,p}, h_{j,p}), \tag{4}$$

where $d(\cdot,\cdot)$ is a latent-space distance metric. We use cosine distance by default, and we also study Wasserstein (Rubner et al., 2000) and Euclidean distances in Section 5. For each stage p, this produces a stage-wise distance matrix to measure the latent state divergence of current stage: $D_p \in \mathbb{R}^{G\times G}$, $D_p[i,j] = d_p(i,j)$, which belongs to the p-th reasoning stage and captures the distances between all G trajectories at that stage. After computing the pairwise distance matrices for all stages, we obtain a sequence of matrices $\mathcal{D} = \{D_p\}_{p=1}^{P}$ that represent latent divergences across the entire set of trajectories. For trajectory o_i at stage p, its latent divergence is defined as the average distance to all other trajectories within the same stage, capturing how distinct o_i is:

$$\Delta_{i,p} = \frac{1}{G-1}\sum_{j\neq i} d_p(i,j). \tag{5}$$

Larger values of $\Delta_{i,p}$ indicate that trajectory o_i is more distinct from other trajectories at the same reasoning stage, providing a stage-wise measure of exploration diversity. To further guide meaningful exploration and balance the exploration–exploitation trade-off, LaDex also evaluates whether the reasoning context up to the current stage is likely to contribute to a correct final answer.

4.3 Latent Classifier Network for Balancing Exploration & Exploitation

Using latent divergence alone to guide exploration may encourage noisy or unproductive trajectories. To ensure that exploration remains meaningful at each reasoning stage, LaDex introduces a Latent Classifier Network (LCN), implemented as a two-layer Transformer followed by a two-layer MLP. For trajectory o_i at stage p, the LCN takes the accumulated latent representation $h_{i,\leq p}$ as input and outputs a success-likelihood score: $\mathrm{LCN}(h_{i,\leq p}) \in [0,1]$. This design is motivated by the observation that the latent representations of correct and incorrect reasoning stages diverge progressively along the reasoning trajectory, making them increasingly distinguishable (Du et al., 2026). The LCN is trained using trajectory-level final-answer correctness as supervision. For a trajectory with label $y_i \in \{0,1\}$, each accumulated latent representation $h_{i,\leq p}$ is assigned the same label y_i. The training objective is the standard binary cross-entropy loss:

$$\mathcal{L}_{\mathrm{LCN}} = -\mathbb{E}_{i,p}\Big[y_i \log \mathrm{LCN}(h_{i,\leq p}) + (1-y_i)\log\big(1-\mathrm{LCN}(h_{i,\leq p})\big)\Big]. \tag{6}$$

During policy optimization, LaDex uses the continuous LCN output as a soft gating factor to adjust the latent-divergence reward at each reasoning stage before aggregation into intrinsic rewards:

$$r_{i,p}^{\mathrm{div}} = \mathrm{LCN}(h_{i,\leq p}) \cdot \Delta_{i,p} = \mathrm{LCN}(h_{i,\leq p}) \cdot \frac{1}{G-1}\sum_{j\neq i} d_p(i,j). \tag{7}$$

The LCN assigns larger weights to divergence signals when the reasoning context up to stage p is likely to yield a correct answer, and smaller weights to signals associated with unproductive or misleading exploration. The LCN is warmed up on initial trajectories and updated online with the LLM to adaptively balance exploration and exploitation. See Appendix A.4 for details.

4.4 Policy Optimization

Since latent divergence at any reasoning stage may open up new reasoning paths for exploration, we assign equal weights to all stages when computing the trajectory-level divergence reward:

$$r_i^{\mathrm{int}} = \frac{1}{P}\sum_{p=1}^{P} r_{i,p}^{\mathrm{div}}. \tag{8}$$

Here, the latent-divergence reward r_i^{int} serves as the intrinsic reward to drive self-exploration, while the final-answer correctness serves as the extrinsic reward r_i^{ext}. Following group-based RLVR, we compute the extrinsic and intrinsic advantages within the same trajectory group:

$$A_i^{\mathrm{ext}} = \frac{r_i^{\mathrm{ext}} - \mathrm{mean}(\{r_j^{\mathrm{ext}}\}_{j=1}^{G})}{\mathrm{std}(\{r_j^{\mathrm{ext}}\}_{j=1}^{G})}, \qquad A_i^{\mathrm{int}} = \max\left(0,\ \frac{r_i^{\mathrm{int}} - \mathrm{mean}(\{r_j^{\mathrm{int}}\}_{j=1}^{G})}{\mathrm{std}(\{r_j^{\mathrm{int}}\}_{j=1}^{G})}\right). \tag{9}$$

The positive part ensures that the intrinsic advantage acts purely as a signal to encourage exploration rather than penalizing trajectories. To prevent the intrinsic advantage from dominating outcome-based learning, LaDex clips it by the magnitude of the extrinsic advantage:

$$A_i = A_i^{\mathrm{ext}} + \min\left(w \cdot A_i^{\mathrm{int}},\ \frac{|A_i^{\mathrm{ext}}|}{\kappa}\right). \tag{10}$$

where w controls the strength of the intrinsic advantage, and κ limits its maximum relative to the extrinsic advantage. The final advantage A_i is then used in the GRPO objective:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(\rho_{i,t}A_i,\ \mathrm{clip}(\rho_{i,t},1-\epsilon,1+\epsilon)A_i\Big) - \beta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})\right]. \tag{11}$$

4.5 Extension to Multi-turn Agentic Scenarios

LaDex can be extended to multi-turn agentic environments by treating each turn as a reasoning stage. Within each stage, LaDex mean-pools the latent states into a compact representation and computes a stage-wise distance matrix between different trajectories to measure latent divergence at that stage. Across the full interaction, this produces a sequence of per-stage divergence matrices that captures how trajectories diverge throughout the entire process. These matrices are then aggregated into intrinsic rewards, preserving the core design of LaDex without any architectural modification.

Table 1: Main pass@8 results on in-domain mathematical reasoning and out-of-domain GPQA-Extend benchmarks. Bold numbers indicate the best performance in each column within the same model block, and underlined numbers indicate the second-best performance. Red numbers indicate relative improvements over the corresponding GRPO results at the same model scale.

	In-domain							OOD		
	MATH500	**Minerva**	**AMC23**	**AIME24**	**AIME25**	**AIME26**	**Avg.**	**GPQA***	**GPQA†**	**Avg.**
Qwen3-4B-Base	82.80	26.84	77.50	23.33	20.00	13.33	40.63	70.20	63.22	66.71
+ GRPO	87.00	30.88	80.00	33.33	26.67	26.67	47.42	76.26	70.11	73.19
+ CDE	86.60	36.03	75.00	**40.00**	23.33	16.67	46.27	74.75	70.11	72.43
+ Entropy Adv.	89.20	31.62	82.50	33.67	30.00	26.67	48.94	73.23	72.70	72.97
+ ROSE	86.60	33.46	82.50	26.67	30.00	26.67	47.65	76.77	73.85	75.31
+ DIVER	89.20	34.93	80.00	30.00	30.00	23.33	47.91	71.72	71.55	71.64
+ LaDex	**90.40** ↑3.91%	**37.50** ↑21.44%	**85.00** ↑6.25%	36.67 ↑10.02%	**36.67** ↑37.50%	**30.00** ↑12.49%	**52.71** ↑11.16%	**77.29** ↑1.35%	**74.71** ↑6.56%	**76.00** ↑3.84%
Qwen3-8B-Base	85.00	34.93	75.00	23.33	20.00	23.33	43.60	74.75	73.86	74.31
+ GRPO	90.40	33.46	87.50	36.67	26.67	30.00	50.78	77.78	74.42	76.10
+ CDE	88.40	36.76	87.50	33.33	26.67	23.33	49.33	80.30	74.71	77.51
+ Entropy Adv.	89.80	32.35	82.50	33.33	30.00	30.00	49.66	79.29	71.84	75.57
+ ROSE	88.00	37.87	82.50	**43.33**	26.67	26.67	50.84	80.81	72.98	76.90
+ DIVER	88.60	35.29	87.50	36.67	30.00	26.67	50.79	76.77	75.29	76.03
+ LaDex	**92.00** ↑1.77%	**38.24** ↑14.29%	**90.00** ↑2.86%	**43.33** ↑18.16%	**40.00** ↑49.98%	**36.67** ↑22.23%	**56.71** ↑11.68%	**82.32** ↑5.84%	**77.88** ↑4.65%	**80.10** ↑5.26%

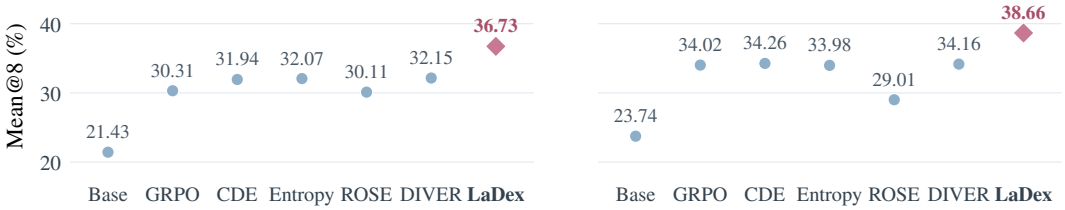

Figure 3: Average mean@8 over six mathematical reasoning benchmarks for Qwen3-4B-Base (left) and Qwen3-8B-Base (right). Detailed results for each benchmark are provided in Table 8.

5 Experiments

To evaluate the effectiveness of LaDex, we conduct experiments on a variety set of benchmarks and investigate the following research questions: **RQ1:** Does LaDex improve final reasoning performance? **RQ2:** Does LaDex enhance the policy's ability to explore effective reasoning trajectories? **RQ3:** How do the LCN and intrinsic reward design influence LaDex's performance? **RQ4:** How do different latent-space distance metrics impact the performance of LaDex? **RQ5:** Can LaDex be effectively extended to multi-turn reasoning scenarios in interactive environments such as WebShop?

5.1 Experimental Setup

Models, Datasets, and Metrics. We conduct our main experiments using three open-source models. **Qwen3-4B-Base** and **Qwen3-8B-Base** (Yang et al., 2025) are employed for mathematical reasoning tasks, with `enable_thinking=False`, while **Qwen2.5-3B-Instruct** (Qwen et al., 2025) is used to evaluate multi-turn reasoning in WebShop (Yao et al., 2022). For mathematical reasoning, models are trained on the Math dataset (Hendrycks et al., 2021) for 4 epochs. In-domain performance is evaluated on six benchmarks: AIME24/25/26, AMC23, Minerva, and MATH500 (Hendrycks et al., 2021; Lewkowycz et al., 2022; Mathematical Association of America, 2026). For out-of-domain evaluation, we use GPQA-Extend (Rein et al., 2024), which is further split into GPQA-Diamond (GPQA*) and GPQA-Extend without Diamond (GPQA†). For mathematical reasoning and GPQA-Extend, we primarily report *pass@k* (Chen et al., 2021a), with supplementary mean@8 results on mathematical and OOD scientific reasoning benchmarks in Table 8. For WebShop, we evaluate both success rate and task scores to measure completion and quality.

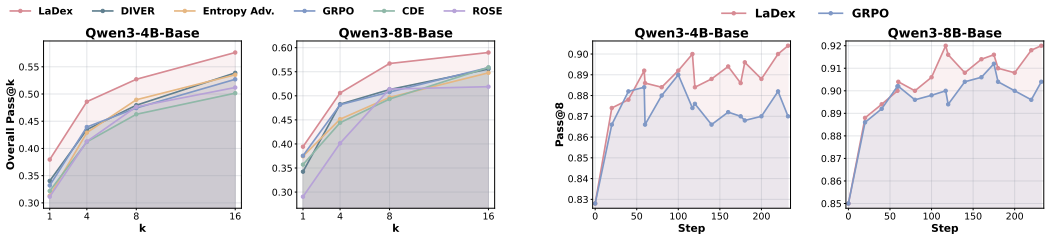

Figure 4: **Left:** Pass@k results on six mathematical benchmarks, comparing LaDex with five RL baselines. **Right:** Pass@8 curves on the validation set for LaDex and GRPO during training.

Baselines and Configurations. We compare LaDex with **GRPO** (Guo et al., 2025) as the main RLVR baseline, as well as several representative exploration-oriented methods. Specifically, we include an entropy-based exploration baseline (**Entropy Adv.**) (Cheng et al., 2025), a count-based curiosity-driven exploration method (**CDE**) (Dai et al., 2026), and an MCTS-style branching exploration method (**ROSE**) (Zhao et al., 2026), and a trajectory-level diversity-based exploration method (**DIVER**) (Hu et al., 2026). For the WebShop experiments, we additionally compare with **GPT-4o**(OpenAI et al., 2024) as a strong proprietary LLM baseline. All trainable baselines are implemented under identical training configurations. Further implementation details, hyperparameter settings and computational cost are provided in Appendix A.2 to facilitate reproducibility.

5.2 Main Results

Reasoning Performance across Model Scales. To answer RQ1, we compare LaDex with established RLVR and exploration methods on Qwen3-4B-Base and Qwen3-8B-Base. Table 1 shows that LaDex achieves the highest average mathematical pass@8 of $\mathbf{52.71}\%$ and $\mathbf{56.71}\%$, respectively, improving over GRPO by $\mathbf{11.16}\%$ and $\mathbf{11.68}\%$. Figure 3 further reports the highest mean@8 averages of 36.73% and 38.66%, with relative improvements of $\mathbf{21.18}\%$ and $\mathbf{13.64}\%$ over GRPO. Together, both metrics demonstrate LaDex's superior performance across diverse and complex reasoning benchmarks, supporting the effectiveness of its exploration strategy. Experiments extending LaDex to other language models, including Llama-3.2-3B-Instruct, are presented in Appendix A.7.

Comparison with Exploration Baselines. We compare LaDex with four representative exploration methods, including CDE, Entropy Adv., ROSE, and DIVER. Relative to the average performance of these four baselines, LaDex improves mathematical pass@8 by $\mathbf{10.52}\%$ on 4B and $\mathbf{13.07}\%$ on 8B (Table 1). Compared with the average of these four methods, the relative mean@8 improvements reach $\mathbf{16.35}\%$ on 4B and $\mathbf{17.68}\%$ on 8B (Figure 3). LaDex also ranks first or second on every mathematical benchmark in pass@8 and achieves the highest mean@8 average at both scales. These results demonstrate that LaDex outperforms existing exploration methods in both pass@8 and mean@8, consistently improving mathematical reasoning performance across the two model scales.

Performance on Challenging and OOD Benchmarks. LaDex also achieves strong results on competition mathematics and OOD scientific reasoning. On 8B, pass@8 reaches 90.00% on AMC23 and $43.33/40.00/36.67\%$ on AIME24/25/26, with relative improvements of 18.16%–37.50% over DIVER on the three AIME benchmarks. On GPQA-Extend, LaDex achieves the highest pass@8 averages of $\mathbf{76.00}\%$ and $\mathbf{80.10}\%$ on 4B and 8B, respectively. These results demonstrate LaDex's generalization beyond the mathematical training domain, with consistent gains in both pass@8 and mean@8 on OOD scientific reasoning benchmarks across both model scales. Supplementary mean@32 results on competition mathematics are provided in Table 9.

5.3 Exploration Ability

Pass@k Exploration Scope. To answer RQ2, we evaluate pass@k for $k \in \{1, 4, 8, 16\}$ (Figure 4; Appendix A.6). LaDex achieves the highest average at every budget on both scales. Relative to the average of GRPO, CDE, Entropy Adv., ROSE, and DIVER, pass@1/4/8/16 improves by $17.27\%/14.18\%/10.65\%/10.19\%$ on 4B and $13.30\%/11.88\%/12.79\%/7.66\%$ on 8B. These results demonstrate LaDex's effectiveness in discovering reasoning paths that lead to correct solu-

Table 2: Ablation results on Qwen3-4B-Base. Avg. reports average pass@8 (%); Δ is the relative change (%) from LaDex (cosine, $P = 32$). Table 10 provides per-benchmark results.

	Distance metric			**Reasoning stages (P)**				**Component ablations**	
	Cosine	Wasserstein	Euclidean	64	16	8	1	w/o LCN	w/o both
Avg.	**52.71**	49.91	51.63	50.95	48.57	51.67	48.38	49.23	47.60
Δ	—	-5.31	-2.05	-3.34	-7.85	-1.97	-8.21	-6.60	-9.69

tions. The consistent gains support a broader scope of effective exploration, enabling the model to solve challenging mathematical problems and better realize its reasoning capacity.

Training Dynamics of Effective Exploration. Figure 4 shows that LaDex overtakes GRPO during training and maintains higher validation performance at both scales. This sustained advantage supports LaDex's effectiveness beyond the initial optimization stage. Additional results on reasoning diversity and detailed qualitative examples are provided in Appendix A.8 and A.9.

5.4 Analysis

Ablation Study. Table 2 summarizes the RQ3 results on the effects of temporal resampling length P and the key components of LaDex. The ablations include varying the number of resampled reasoning stages, removing the LCN, and removing the intrinsic reward. Here, LaDex w/o both denotes the variant without the intrinsic reward. The results show that $P = 32$ achieves the best average performance, whereas both $P = 64$ and $P < 32$ lead to performance drops. Coarse resampling with larger pooling windows may discard reasoning information, whereas overly fine resampling with smaller windows may overemphasize suboptimal local variations. Using a single resampled stage or removing the LCN also leads to performance degradation, suggesting that stage-wise latent comparison and exploration-quality monitoring are both necessary.

Effect of Distance Metrics. Table 2 reports the RQ4 results examining how different distance metrics affect latent-divergence estimation in LaDex. Cosine distance achieves the highest average performance across the six mathematical benchmarks, while Euclidean distance remains competitive on some benchmarks and Wasserstein distance (Rubner et al., 2000) performs worse overall. These results suggest that directional similarity provides a more consistent exploration signal than absolute or distributional distance. We therefore use cosine distance as the default metric in LaDex.

Extension to Multi-Turn Reasoning. To answer RQ5, we extend LaDex to the multi-turn, agentic WebShop environment using Qwen2.5-3B-Instruct. As shown in Table 3, LaDex improves the success rate from 41.40 to 63.49 and the score from 59.79 to 77.60 over GRPO, corresponding to relative gains of **53.36%** and **29.79%**, respectively. LaDex also substantially outperforms both prompting-based baselines. These results demonstrate that its latent-state-guided exploration generalizes effectively to multi-turn interactive tasks. Additional experimental details and results are provided in Appendix A.5.

Table 3: Comparison of performance on the WebShop benchmark. Bold numbers indicate the best performance in each column.

Type	**Method**	**Succ.**	**Score**
Prompting	Qwen2.5-3B-Instruct	1.56	8.13
Prompting	GPT-4o	23.70	31.80
Training	GRPO	41.40	59.79
Training	LaDex	**63.49**	**77.60**

6 Conclusions and Limitations

We presented LaDex, a latent state-guided exploration method that improves LLM reasoning. Experiments on challenging benchmarks show it consistently enhances reasoning performance and exploration quality compared with strong baselines, demonstrating the value of latent state signals for self-exploration. Despite its effectiveness, LaDex currently considers only three distance measures, cosine, Wasserstein and Euclidean, for estimating latent state divergence. Other metrics may capture divergence more effectively or be better suited to different model architectures and reasoning tasks. We leave a more systematic investigation of this across model architectures to future work.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, improving the clarity of presentation, and adapting the manuscript. We did not use generative AI tools to generate experimental results or determine the scientific conclusions of this work. We reviewed and revised all AI-assisted text and formatting, and manually verified the technical claims, mathematical formulations, citations, experimental results, and implementation details. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops a reinforcement learning method for improving the self-exploration and reasoning capabilities of large language models. Our experiments use publicly available open-source models and benchmarks and do not involve human subjects, crowdsourcing, private personally identifiable information, or newly collected user data. Nevertheless, improvements in model reasoning capabilities may introduce downstream risks if such models are deployed in high-stakes or safety-critical applications without appropriate evaluation and oversight. The proposed method should therefore be accompanied by application-specific safety assessments and safeguards before real-world deployment. We have reviewed the ICLR Code of Ethics and believe that this work conforms to its requirements.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce our main results throughout the paper and supplementary materials. The experimental setup, code, datasets, evaluation protocols, and baseline configurations are described in the main text. The Appendix provides additional implementation details, training configurations, hyperparameters, computational resource information, latent-state analysis, and extended experimental results. We use publicly available models and benchmarks and cite their original sources. An anonymous implementation is available at `https://anonymous.4open.science/r/LLMRL-CDE2/`.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 1479–1487, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. Evaluating large language models trained on code, 2021a. URL https://arxiv.org/abs/2107.03374.

Yilei Chen, Souradip Chakraborty, Lorenz Wolf, Ioannis Paschalidis, and Aldo Pacchiano. Post-training large language models for diverse high-quality responses. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BV3bLKaJ3Y.

Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13359–13368, 2021b.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. In *AAAI Conference on Artificial Intelligence*, 2025. URL https://api.semanticscholar.org/CorpusID:279410424.

Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, Haitao Mi, Zhaopeng Tu, Rui Liu, Tong Zheng, Hongtu Zhu, and Dong Yu. CDE: Curiosity-driven exploration for efficient reinforcement learning in large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5rXN5knHKW.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TX4k7BF6aO.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2jkAk3EP0v.

Haodong Duan, Jiaqi Wang, Kai Chen, and Dahua Lin. Pyskl: Towards good practices for skeleton action recognition. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, pp. 7351–7354, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392037. doi: 10.1145/3503161.3548546. URL https://doi.org/10.1145/3503161.3548546.

Halil Alperen Gozeten, Muhammed Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *Tokenization Workshop*, 2026. URL https://openreview.net/forum?id=vBfQpjPOwr.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, Proceedings of Machine Learning Research, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9G7AbBrd27.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya

Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

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, 2025. URL https://arxiv.org/abs/2508.17445.

Runze Liu, Jiakang Wang, Yuling Shi, Zhihui Xie, Chenxin An, Kaiyan Zhang, Jian Zhao, Xiaodong Gu, Lei Lin, Wenping Hu, Xiu Li, Fuzheng Zhang, Guorui Zhou, and Kun Gai. Attention as a compass: Efficient exploration for process-supervised RL in reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=NCN8oUsiNf.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Cong Lu, Shengran Hu, and Jeff Clune. Automated capability discovery via model self-exploration. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=rQf6YtHfB5.

Mathematical Association of America. Mathematical association of america, 2026. URL https://maa.org. Accessed: 2026-04-23.

MiniMax, :, Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention, 2025. URL https://arxiv.org/abs/2506.13585.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *CoRR*, abs/1306.0940, 2013. URL http://arxiv.org/abs/1306.0940.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2778–2787. JMLR.org, 2017.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000. ISSN 0920-5691. doi: 10.1023/A:1026543900054. URL https://doi.org/10.1023/A:1026543900054.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=hYfOPXrbUr.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=AQsko3PPUe.

Wenhui Tan, Fiorenzo Parascandolo, Enver Sangineto, Jianzhong Ju, Zhenbo Luo, Qian Cao, Rita Cucchiara, Ruihua Song, and Jian Luan. Restoring exploration after post-training: Latent exploration decoding for large reasoning models, 2026b. URL https://arxiv.org/abs/2602.01698.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024. URL https://arxiv.org/abs/2212.03533.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-cot: Supervised implicit chain-of-thought. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6YRJ4jmVQl.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1137. URL https://aclanthology.org/2025.acl-long.1137/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik R Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=R9KnuFlvnU.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Shenao Zhang, Donghan Yu, Hiteshi Sharma, Han Zhong, Zhihan Liu, Ziyi Yang, Shuohang Wang, Hany Hassan Awadalla, and Zhaoran Wang. Self-exploring language models: Active preference elicitation for online alignment. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=FoQK84nwY3.

Shimin Zhang, Xianwei Chen, Yufan Shen, Ziyuan Ye, and Jibin Wu. Relax: Reasoning with latent exploration for large reasoning models, 2026a. URL https://arxiv.org/abs/2512.07558.

Xuan Zhang, Ruixiao Li, Zhijian Zhou, Long Li, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. Count counts: Motivating exploration in LLM reasoning with count-based intrinsic rewards. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=9xIBbfItGP.

Ziqi Zhao, Zhaochun Ren, Jiahong Zou, Liu Yang, Zhiwei Xu, Xuri Ge, Zhumin Chen, Xinyu Ma, Daiting Shi, Shuaiqiang Wang, Dawei Yin, and Xin Xin. Reinforced efficient reasoning via semantically diverse exploration, 2026. URL https://arxiv.org/abs/2601.05053.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=UdOEZgWJLc.

A APPENDIX

A.1 NOTATION

We summarize the main notations used in this paper as follows:

- Q: input question or prompt.
- G: group size, i.e., the number of trajectories sampled for each prompt.
- $\{o_i\}_{i=1}^{G}$: the group of output trajectories sampled from the LM policy for the same prompt Q, where o_i denotes the i-th trajectory.
- P: number of reasoning stages per trajectory after temporal alignment.
- $o_{i,p}$: the p-th reasoning stage of trajectory o_i.
- $K_{i,p}$: number of tokens in the p-th reasoning stage of trajectory o_i.
- $h_{i,p}^{(k)}$: token-level hidden state of the k-th token in reasoning stage $o_{i,p}$.
- $h_{i,p}$: stage-level latent representation of reasoning stage $o_{i,p}$, obtained by mean pooling token-level hidden states:
$$h_{i,p} = \frac{1}{K_{i,p}} \sum_{k=1}^{K_{i,p}} h_{i,p}^{(k)} \in \mathbb{R}^d.$$
- H_i: latent trajectory of o_i, i.e., the sequence of stage-level latent states after alignment:
$$H_i = \{h_{i,p}\}_{p=1}^{P}.$$
- $h_{i,\le p}$: accumulated latent representation of trajectory o_i up to reasoning stage p.
- D_p: pairwise latent distance matrix at reasoning stage p, where
$$D_p \in \mathbb{R}^{G\times G}.$$
- $d_p(i,j)$: latent-space distance between the p-th stages of trajectories o_i and o_j:
$$d_p(i,j) = 1 - \cos(h_{i,p}, h_{j,p}).$$
- $\Delta_{i,p}$: latent divergence of trajectory o_i at reasoning stage p:
$$\Delta_{i,p} = \frac{1}{G-1} \sum_{j\neq i} d_p(i,j).$$
- y_i: trajectory-level final-answer correctness label, $y_i \in \{0,1\}$.
- $\mathrm{LCN}(h_{i,\le p})$: Latent Classifier Network output for the accumulated latent representation $h_{i,\le p}$, where
$$\mathrm{LCN}(h_{i,\le p}) \in [0,1].$$
- $r_{i,p}^{\mathrm{div}}$: LCN-weighted divergence reward for trajectory o_i at reasoning stage p:
$$r_{i,p}^{\mathrm{div}} = \mathrm{LCN}(h_{i,\le p}) \cdot \Delta_{i,p}.$$
- r_i^{int}: trajectory-level intrinsic reward:
$$r_i^{\mathrm{int}} = \frac{1}{P} \sum_{p=1}^{P} r_{i,p}^{\mathrm{div}}.$$
- r_i^{ext}: trajectory-level extrinsic reward from final-answer correctness.
- A_i^{ext}: normalized extrinsic advantage:
$$A_i^{\mathrm{ext}} = \frac{r_i^{\mathrm{ext}} - \mathrm{mean}(\{r_j^{\mathrm{ext}}\}_{j=1}^{G})}{\mathrm{std}(\{r_j^{\mathrm{ext}}\}_{j=1}^{G})}.$$

- A_i^{int}: positive normalized intrinsic advantage:

$$A_i^{\text{int}} = \max\left(0, \frac{r_i^{\text{int}} - \text{mean}(\{r_j^{\text{int}}\}_{j=1}^{G})}{\text{std}(\{r_j^{\text{int}}\}_{j=1}^{G})}\right).$$

- A_i: final advantage used for policy optimization:

$$A_i = A_i^{\text{ext}} + \min\left(w \cdot A_i^{\text{int}}, \frac{|A_i^{\text{ext}}|}{\kappa}\right).$$

- w: weighting coefficient for the intrinsic advantage.
- κ: scaling factor controlling the upper bound of the intrinsic contribution.

A.2 Training Configuration

All training runs are conducted on four NVIDIA A800 80GB GPUs. We use one epoch to warm up the Latent Classifier Network. The wall-clock training time is approximately 18 wall clock hours for the 4B model and 30 hours for the 8B model. In comparison, the branching-based ROSE method requires approximately 36 and 48 wall clock hours for the 4B and 8B models, respectively. We further report the per-iteration computational cost breakdown in Figure 6. The additional overhead of LaDex mainly comes from the rollout stage and the computation of the intrinsic advantage, while the old-policy, reference-policy, and policy-update stages remain comparable to GRPO. Overall, although LaDex incurs additional training time compared with GRPO, it remains substantially more computationally efficient than branching-based methods such as ROSE while preserving the same RL training pipeline.

In Webshop, we use a batch size of 8 and train for 150 steps. Each rollout consists of up to 15 environment interactions, with a rollout length of 15 stages used during training. And we set ppo mini batch to 64. Overall, the training setup focuses on efficient RL optimization for task-oriented dialog in WebShop while keeping computational costs low.

Table 4: Our RL training configurations on mathematical reasoning tasks.

Hyperparameter	**Value**
General training hyperparameters	
Optimizer	AdamW
Policy learning rate	$1e{-6}$
Training batch size	128
Samples per prompt	8
Mini-batch size	32
Max prompt length	1024
Max response length	4096
Training epochs	4
KL loss coefficient	0.001
Validation temperature	0.6
Validation top-p	0.95
Method-specific hyperparameters	
Intrinsic reward weight w	0.4
Advantage clipping coefficient κ	2.0

Table 5: Training configuration of the Latent Classifier Network.

Hyperparameter	**4B Model**	**8B Model**
Hidden dimension	2560	4096
Number of attention heads	32	32
Feed-forward dimension	4×2560	4×4096
Classifier hidden dimension	1024	1024
Maximum positions	4096	4096
Learning rate	$1e{-4}$	$1e{-4}$
Weight decay	$1e{-2}$	$1e{-2}$
Micro-batch size	32	32
Update mode	Per RL training step	Per RL training step

User Prompt

[Question]
Let's think step by step and output the final answer within \boxed{}.

Figure 5: Prompt template used for both training and inference.

Table 6: RL training configurations for WebShop.

Hyperparameter	**Value**
General training hyperparameters	
Training batch size	8
Training steps	150
Maximum rollout length	15
PPO mini-batch size	64
Policy learning rate	$1e{-6}$
KL loss coefficient	0.001
Validation temperature	0.6
Validation top-p	0.95
Method-specific hyperparameters	
Intrinsic reward weight w	0.4
Advantage clipping coefficient κ	2.0

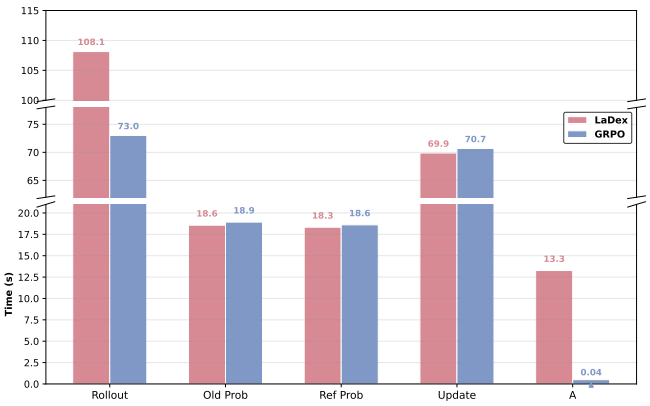

Figure 6: Per-iteration training time breakdown of GRPO and LaDex. LaDex introduces additional cost mainly during rollout and intrinsic-advantage computation, while the old-policy, reference-policy, and policy-update stages remain comparable to GRPO.

A.3 Experiments on Latent Trajectory Distance

We analyze whether latent trajectories can reveal path-level differences among reasoning rollouts. We use Qwen-series models (across 3B to 32B) to sample rollouts on MATH500 (Hendrycks et al., 2021). For each question, we sample $G = 8$ rollouts and focus on correct rollouts when constructing path pairs. Following Sections 4.1 and 4.2, we segment each rollout into reasoning stages, extract stage-level latent representations, and compute pairwise cosine distances between latent trajectories. Each latent trajectory is resampled to $P = 32$ normalized reasoning-progress stages before computing distances.

We define two types of rollout pairs based on manually annotated reasoning paths. **Same Path** denotes two correct rollouts that follow the same solution idea and reasoning progression, while **Different Path** denotes two correct rollouts that use different solution ideas or different combinations of reasoning stages. **During annotation, superficial lexical substitutions or paraphrases are not treated as Different Path if the underlying reasoning path remains the same.** Thus, Same Path pairs capture within-path perturbations, while Different Path pairs capture cases where correct rollouts follow genuinely different reasoning paths. This setting allows us to directly test whether different reasoning paths exhibit larger latent-space distances than same-path variations.

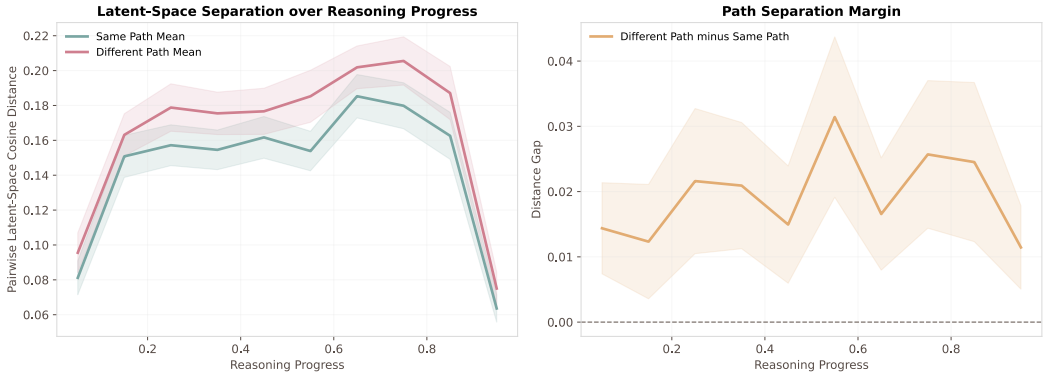

Figure 7: Aggregate latent-space separation between Same Path and Different Path rollout pairs on MATH500 using Qwen2.5-3B-Instruct. **Left:** mean pairwise latent-space cosine distance over normalized reasoning progress. Solid curves denote mean distances for Same Path and Different Path pairs, and shaded regions denote confidence intervals across prompts. **Right:** within-prompt separation margin, computed as Different Path distance minus Same Path distance.

Figure 7 shows the aggregate result for Qwen2.5-3B-Instruct. Across reasoning progress, Different Path pairs generally exhibit larger latent-space distances than Same Path pairs. The positive separation margin further indicates that latent trajectories provide an informative signal for distinguishing path-level exploration from local variations within the same reasoning path.

Figure 8 complements the aggregate analysis with separate results for Qwen3-4B-Base, Qwen3-8B-Base, and Qwen3-14B-Base. Different Path pairs generally have larger mean latent-space distances than Same Path pairs across all three model scales, particularly during intermediate reasoning stages. This shows that the separation observed in the aggregate analysis is also present within individual model scales.

We also provide representative examples to visualize the latent trajectory geometry. Figures 9–11 show three problems with different levels of same-path variation. For each example, the left panel plots the pairwise latent-space distance over reasoning progress, while the middle and right panels show PCA projections of the compared rollout pairs into a three-dimensional latent space. Across these examples, same-path pairs remain relatively close, whereas different-path pairs exhibit clearer separation in latent space.

Overall, these results support the observation used in LaDex: latent trajectories following the same reasoning path tend to remain closer in latent space, while trajectories corresponding to different reasoning paths become more separated as reasoning unfolds. This motivates using stage-level latent space distance as a path-level exploration signal in group-based RLVR.

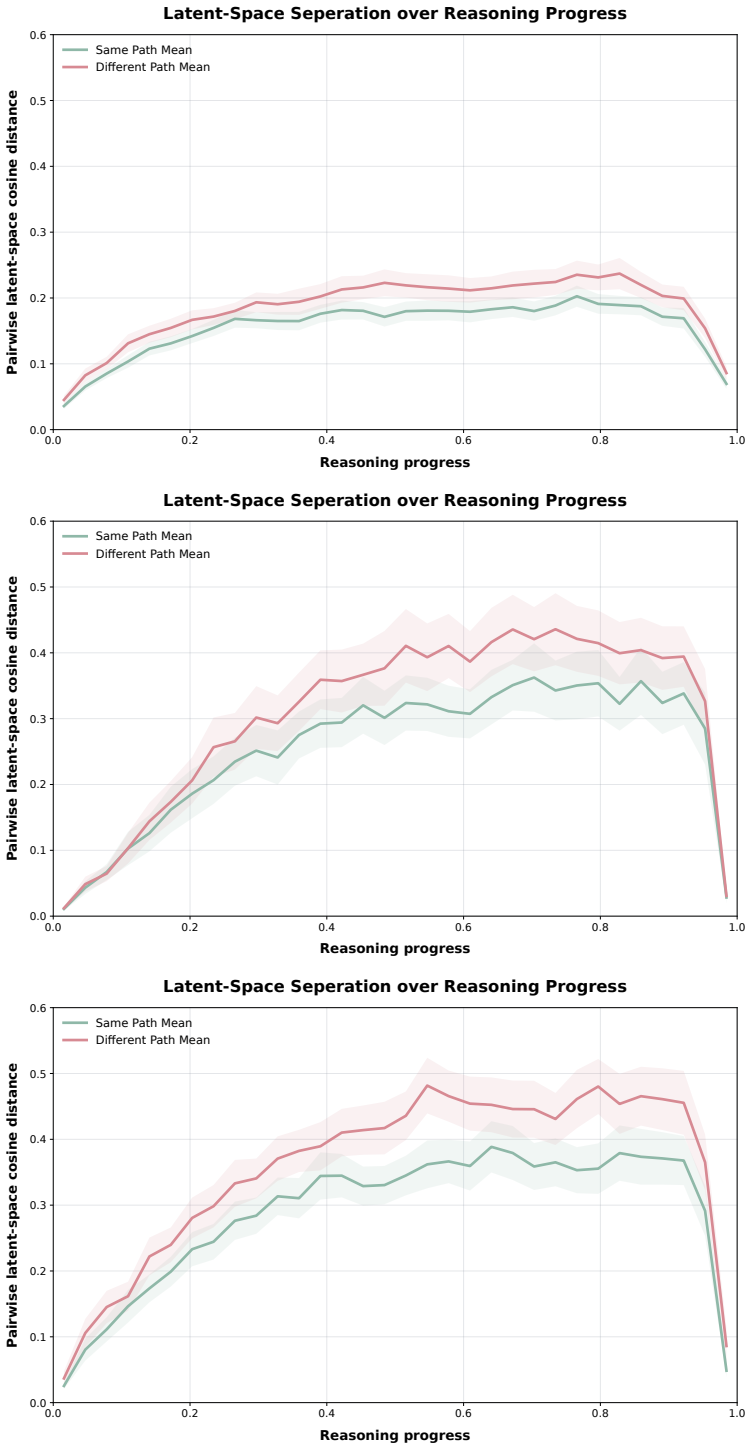

Figure 8: Latent-space separation on MATH500 across model scales, supplementing Figure 7. From top to bottom: Qwen3-4B-Base, Qwen3-8B-Base, and Qwen3-14B-Base.

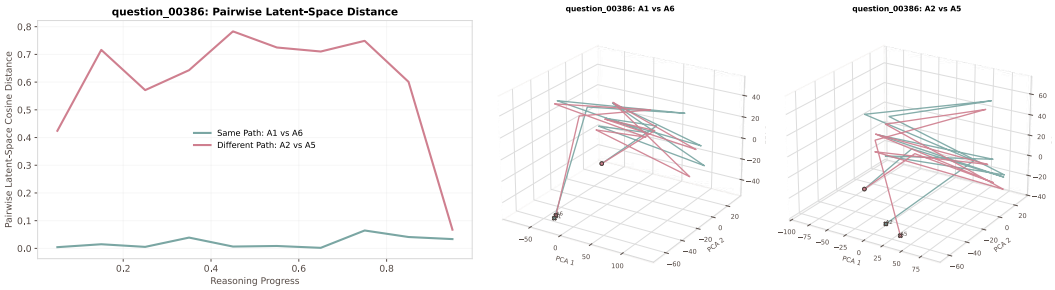

Figure 9: Example on `question_00386`. The Same Path pair A1 vs. A6 remains close in latent space, while the Different Path pair A2 vs. A5 exhibits substantially larger latent-space distance over most reasoning stages. The PCA visualizations further show that same-path trajectories stay close, whereas different-path trajectories separate more clearly in latent space.

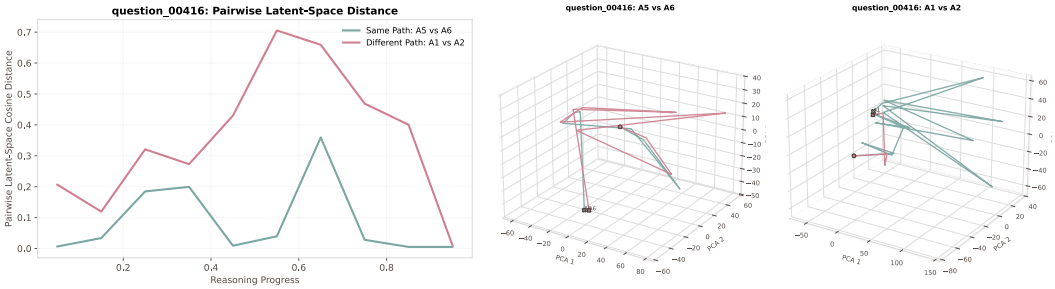

Figure 10: Example on `question_00416`. The Same Path pair A5 vs. A6 shows relatively small latent-space distance, while the Different Path pair A1 vs. A2 exhibits a larger separation during the reasoning process. The PCA projections illustrate that the two different-path trajectories follow more separated latent directions than the same-path pair.

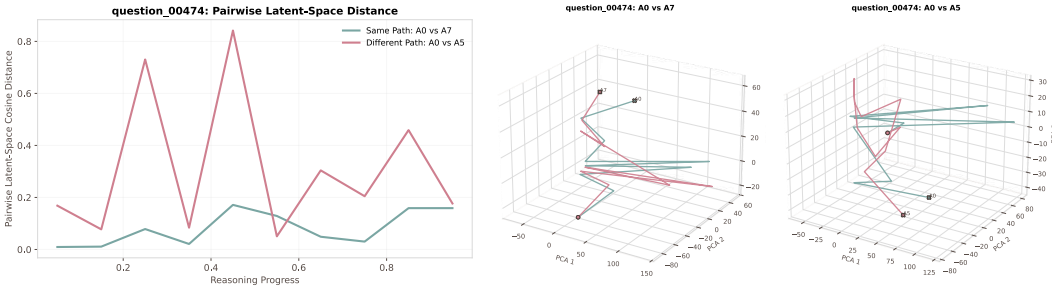

Figure 11: Example on `question_00474`. The Same Path pair A0 vs. A7 has smaller latent-space distance, while the Different Path pair A0 vs. A5 shows larger separation at several reasoning stages. The PCA visualizations provide an intuitive view of how different reasoning paths diverge in latent space.

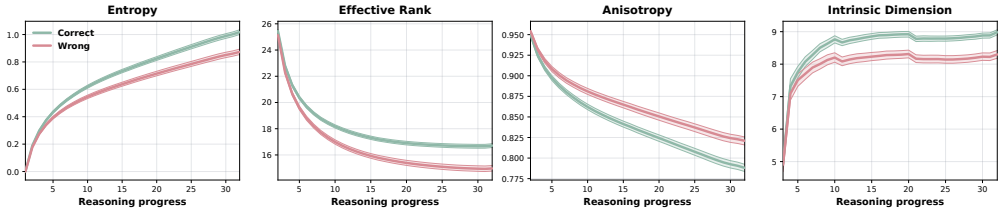

Figure 12: Latent-state statistics of correct and incorrect reasoning trajectories. Entropy, effective rank, anisotropy, and intrinsic dimension are plotted against reasoning progress for Qwen. Correct trajectories (green) generally exhibit higher entropy, effective rank, and intrinsic dimension, and lower anisotropy than incorrect trajectories (pink), with clearer separation at intermediate and later stages.

A.4 Latent Classifier Network Implementation

Inspired by LTO, we examine whether latent-state statistics differ between correct and incorrect reasoning trajectories Du et al. (2026). While LTO establishes these properties in latent reasoning models and models from the Llama family, we investigate whether similar correctness-associated trends also emerge in Qwen-series models. This analysis examines whether such statistical differences could provide learnable features for distinguishing correct and incorrect reasoning trajectories. As shown in Figure 12, correct trajectories generally exhibit higher entropy, effective rank, and intrinsic dimension, together with lower anisotropy, particularly at intermediate and later reasoning stages. These correctness-associated differences suggest that latent trajectories contain potentially learnable signals, motivating the design of LCN to predict trajectory correctness from accumulated latent states. We implement the Latent Classifier Network (LCN) as a Transformer-based binary classifier over accumulated stage-level latent states. After trajectory alignment, each rollout o_i is represented as a sequence of P stage-level latent representations $H_i = \{h_{i,1}, h_{i,2}, \ldots, h_{i,P}\}$, where $h_{i,p} \in \mathbb{R}^d$. For the p-th reasoning stage, the LCN takes the accumulated latent states $h_{i,\leq p} = \{h_{i,1}, h_{i,2}, \ldots, h_{i,p}\}$ as input. In our implementation, each latent trajectory is resampled to $P = 32$ stages before constructing these accumulated inputs.

Network Architecture. Let the input sequence be $X_{i,p} = h_{i,\leq p} \in \mathbb{R}^{p \times d}$. The LCN first projects the input hidden states to the model dimension when necessary; if the hidden-state dimension already matches the model dimension, this projection is the identity. We then add sinusoidal positional encodings to preserve the order of reasoning stages and feed the resulting sequence into a Transformer encoder. After the Transformer encoder, we apply LayerNorm and masked mean pooling over valid prefix positions. The pooled representation is then passed to a two-layer MLP classification head to produce a scalar logit $\ell_{i,p}$. The LCN score is the sigmoid probability:

$$\mathrm{LCN}(h_{i,\leq p}) = \sigma(\ell_{i,p}) \in [0, 1]. \tag{12}$$

In our experiments, the Transformer-based LCN uses two Transformer encoder layers, sinusoidal positional encoding, masked mean pooling, and a two-layer MLP classifier head. For Qwen3-4B-Base, the hidden dimension and model dimension are both 2560; the Transformer uses 32 attention heads, feed-forward dimension 10240, dropout 0.1, and GELU activation.

Training. The LCN is trained with rollout-level final-answer correctness labels. For rollout o_i, let $y_i \in \{0, 1\}$ denote whether its final answer is correct. Each accumulated prefix from the same rollout shares this rollout-level label, i.e., $\mathrm{label}(h_{i,\leq p}) = y_i$ for $p = 1, \ldots, P$. Thus, each rollout produces P supervised examples for LCN training.

Given the logit $\ell_{i,p}$ for prefix $h_{i,\leq p}$, the LCN is optimized with binary cross entropy with logits:

$$\mathcal{L}_{\mathrm{LCN}} = -\mathbb{E}_{i,p}\left[y_i \log \sigma(\ell_{i,p}) + (1 - y_i) \log(1 - \sigma(\ell_{i,p}))\right]. \tag{13}$$

We first warm up the LCN on offline collected rollouts. During warm-up, rollouts are grouped by question, their hidden trajectories are resampled to $P = 32$ stages, and all accumulated prefixes are used as training examples. The LCN is then further updated online during RL training using the same rollout-level correctness supervision.

Inference. During policy optimization, the trained LCN scores each accumulated latent prefix and returns the probability $\mathrm{LCN}(h_{i,\le p}) = \sigma(\ell_{i,p})$. We use this probability as a soft gate rather than applying a hard threshold. For rollout o_i at stage p, the gate score modulates the stage-level latent divergence:

$$r_{i,p}^{\mathrm{div}} = \mathrm{LCN}(h_{i,\le p}) \cdot \Delta_{i,p}. \tag{14}$$

The trajectory-level intrinsic reward is obtained by aggregating these gated stage-level rewards over all reasoning stages. In our main experiments, we use mean aggregation:

$$r_i^{\mathrm{int}} = \frac{1}{P} \sum_{p=1}^{P} r_{i,p}^{\mathrm{div}}. \tag{15}$$

Therefore, the LCN does not replace the final-answer reward; instead, it provides a soft success-likelihood gate that controls how much each stage-level latent divergence contributes to the intrinsic exploration reward.

Performance during RL training. During RL training, we tracked the LCN's predictions of latent states. In our method, the LCN's predicted probabilities are used to adjust the latent state divergence. For ease of visualization here, we treat probabilities ≥ 0.5 as indicating that the latent state is likely to lead to the correct answer, and probabilities < 0.5 as otherwise. We then computed recall, precision, and accuracy. As shown in Figure 13, the LCN consistently maintains high accuracy, recall, and precision throughout training.

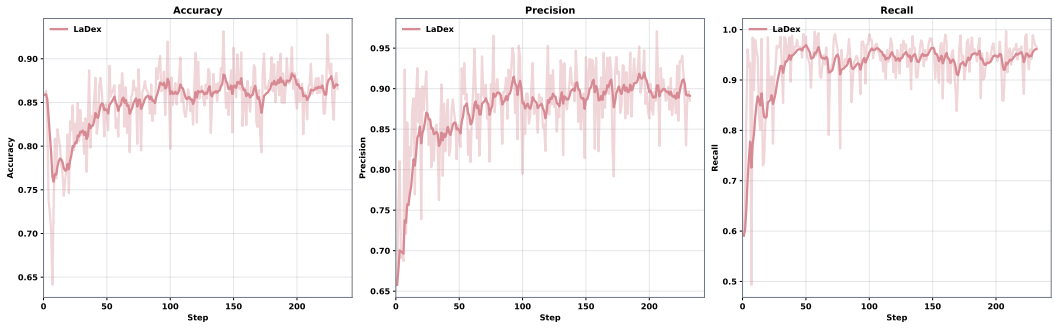

Figure 13: Performance curves (recall, precision, and accuracy) of the LCN model throughout reinforcement learning (RL) training.

A.5 More Experiments Results on WebShop

We conducted an additional study to examine whether LaDex can be extended from single-turn reasoning scenarios to multi-turn, agentic environments. Specifically, we evaluated LaDex on the WebShop task using Qwen2.5-3B-Instruct, where the model is required to interact with the environment over multiple turns, interpret intermediate observations, select actions, and progressively refine its search toward a successful task completion. In this setting, each interaction turn is treated as a reasoning stage, and the latent state of each stage is directly used to compute the latent-divergence reward. No additional segmentation is applied, since the turn-level structure of WebShop naturally provides a sequence of intermediate reasoning and decision states.

As shown in Figure 14, LaDex achieves a stronger and more stable training trend than GRPO on WebShop. The higher score indicates that LaDex learns to make better product-selection decisions across the interaction process, improving not only isolated actions but also the overall quality of the final purchase. More importantly, the consistent gain in success rate shows that these improvements are not limited to partial matching, but are further translated into fully completed tasks. This suggests that latent state-guided exploration remains effective in the multi-turn WebShop setting, where the agent must accumulate correct decisions over several interaction stages before reaching a successful outcome.

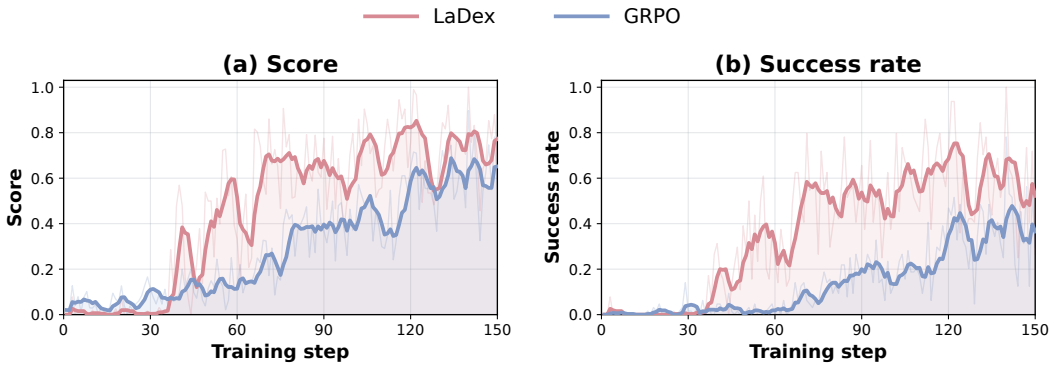

Figure 14: Training curves of LaDex and GRPO on WebShop.

A.6 More Experiments Results on Mathematics Reasoning

Table 7: **Overall Pass@k comparison on Qwen3-4B-Base and Qwen3-8B-Base.** We report the average Pass@k (%) over six mathematical reasoning benchmarks at sampling budgets $k \in \{1, 4, 8, 16\}$.

Model	Method	Pass@1	Pass@4	Pass@8	Pass@16
Qwen3-4B-Base	LaDex	37.96	48.59	52.71	57.62
	DIVER	34.03	43.46	47.91	53.88
	GRPO	33.20	43.93	47.42	52.70
	Entropy Adv.	31.31	42.90	48.94	53.53
	CDE	32.18	41.20	46.27	50.15
	ROSE	31.13	41.28	47.65	51.19
Qwen3-8B-Base	LaDex	39.42	50.59	56.71	59.00
	DIVER	34.24	48.25	50.79	55.56
	GRPO	37.54	48.15	50.78	55.87
	Entropy Adv.	37.44	45.14	49.66	54.78
	CDE	35.71	44.41	49.33	55.90
	ROSE	29.04	40.14	50.84	51.90

Performance across sampling budgets. Table 7 shows that LaDex achieves the highest average pass@k at every evaluated sampling budget on both model scales. On Qwen3-4B-Base, LaDex achieves relative improvements over GRPO of 14.34%, 10.61%, 11.16%, and 9.34% at $k = 1, 4, 8, 16$, respectively; the corresponding relative improvements on Qwen3-8B-Base are 5.01%, 5.07%, 11.68%, and 5.60%. At $k = 16$, LaDex reaches 57.62% and 59.00% on the 4B and 8B models, with relative improvements of 6.94% and 5.55% over the strongest competing baseline at each scale, respectively. The gains at $k = 1$ and larger budgets indicate improvements in both single-sample accuracy and solution coverage across multiple sampled reasoning trajectories.

Mean@8 performance. Table 8 reports mean@8 on six in-domain mathematical benchmarks and two OOD GPQA-Extend subsets. LaDex achieves the highest mathematical average at both scales, reaching 36.73% and 38.66%, with relative improvements of 21.18% and 13.64% over GRPO. On GPQA-Extend, LaDex also achieves the highest average at both scales, reaching 38.28% and 42.09%, respectively. At 8B, Entropy Adv. is slightly higher on GPQA w/o D (37.79% versus 37.72%), while LaDex leads on GPQA-D and the OOD average. Together, these results complement pass@k by demonstrating improved average response accuracy in both evaluation domains.

Table 8: Mean@8 results (%) on in-domain mathematical reasoning and OOD GPQA-Extend benchmarks for Qwen3-4B-Base and Qwen3-8B-Base. Mean@8 averages accuracy over eight responses per problem. Bold and underlined values indicate the best and second-best results within each model block, respectively; red values indicate relative improvements over GRPO.

	In-domain							OOD		
	MATH500	**Minerva**	**AMC23**	**AIME24**	**AIME25**	**AIME26**	**Avg.**	**GPQA***	**GPQA†**	**Avg.**
Qwen3-4B-Base	53.67	13.79	40.31	10.00	5.00	5.83	21.43	22.79	19.22	21.01
+ GRPO	78.15	16.50	53.44	12.50	12.08	9.17	30.31	32.32	31.72	32.02
+ CDE	72.67	25.55	54.69	14.17	14.17	10.42	31.94	37.12	36.46	36.79
+ Entropy Adv.	77.15	12.68	59.69	16.67	15.83	10.42	32.07	32.70	32.40	32.55
+ ROSE	73.28	21.23	50.31	14.58	10.00	11.25	30.11	34.97	31.39	33.18
+ DIVER	77.75	21.28	59.69	12.50	11.25	10.42	32.15	34.09	31.86	32.98
+ LaDex	**80.60** ↑3.13%	**25.69** ↑55.70%	**60.31** ↑12.86%	**22.08** ↑76.64%	**20.00** ↑65.56%	**11.67** ↑27.26%	**36.73** ↑21.18%	**38.95** ↑20.51%	**37.61** ↑18.57%	**38.28** ↑19.55%
Qwen3-8B-Base	57.93	13.47	41.88	11.67	9.17	8.33	23.74	28.79	27.64	28.22
+ GRPO	80.20	18.06	58.75	20.00	14.17	12.92	34.02	39.96	34.12	37.04
+ CDE	77.83	20.96	63.44	16.25	14.17	12.92	34.26	44.32	35.20	39.76
+ Entropy Adv.	80.58	15.26	62.19	18.75	15.83	11.25	33.98	42.61	**37.79**	40.20
+ ROSE	69.85	17.88	54.69	11.67	10.00	10.00	29.01	37.82	34.05	35.93
+ DIVER	77.35	21.88	60.31	17.92	13.75	13.75	34.16	41.16	35.34	38.25
+ LaDex	**82.03** ↑2.28%	**25.78** ↑42.75%	**65.00** ↑10.64%	**23.75** ↑18.75%	**17.50** ↑23.50%	**17.92** ↑38.70%	**38.66** ↑13.64%	**46.46** ↑16.27%	37.72 ↑10.55%	**42.09** ↑13.63%

Table 9: Mean@32 results (%) on AMC23 and AIME24/25/26 for Qwen3-4B-Base and Qwen3-8B-Base. Bold and underlined values indicate the best and second-best results within each model block, respectively; red values indicate relative improvements over the corresponding GRPO.

Method	**AMC23**	**AIME24**	**AIME25**	**AIME26**	**Avg.**
Qwen3-4B-Base	38.05	8.96	4.17	6.56	14.43
+ GRPO	55.16	15.83	11.35	10.42	23.19
+ CDE	55.31	16.98	12.60	7.08	22.99
+ Entropy Adv.	52.73	16.35	13.75	10.94	23.44
+ ROSE	49.06	13.12	11.56	10.52	21.07
+ DIVER	55.62	14.06	13.54	12.19	23.85
+ LaDex	**58.52** ↑6.09%	**20.42** ↑29.00%	**19.27** ↑69.78%	**12.81** ↑22.94%	**27.75** ↑19.66%
Qwen3-8B-Base	49.06	11.35	9.38	6.56	19.09
+ GRPO	58.36	21.04	15.10	13.23	26.93
+ CDE	61.41	16.04	14.06	11.67	25.79
+ Entropy Adv.	60.94	18.44	14.37	11.25	26.25
+ ROSE	54.69	12.29	12.29	8.33	21.90
+ DIVER	63.83	19.06	15.73	11.35	27.49
+ LaDex	**65.39** ↑12.05%	**24.17** ↑14.88%	**18.02** ↑19.34%	**15.94** ↑20.48%	**30.88** ↑14.67%

Mean@32 performance. Table 9 reports mean@32 on AMC23 and AIME24/25/26. LaDex achieves the highest scores on all four benchmarks at both scales, with averages of 27.75% and 30.88% on 4B and 8B, respectively, improving over GRPO by 19.66% and 14.67%. Its lead across all four competition benchmarks further supports LaDex's effectiveness in improving average response accuracy when evaluated using 32 sampled responses for each problem.

Detailed ablation results. Table 10 provides the per-benchmark results summarized in Table 2. The default LaDex configuration uses cosine distance and $P = 32$ reasoning stages.

Table 10: Distance-metric comparisons and ablation results for LaDex. The upper block compares latent-space distance metrics, while the lower block evaluates different numbers of reasoning stages and the removal of key components. *Stages* denotes the number of reasoning stages used for latent-divergence estimation, and *w/o both* removes both latent state divergence and the LCN.

Method	MATH500	Minerva	AMC23	AIME24	AIME25	AIME26	Avg.
LaDex (Cosine)	**90.40**	37.50	**85.00**	36.67	**36.67**	30.00	**52.71**
LaDex (Wasserstein)	88.60	36.67	77.50	36.67	33.33	26.67	49.91
LaDex (Euclidean)	89.40	**37.87**	82.50	33.33	33.33	**33.33**	51.63
LaDex w/ 64 stages	88.00	36.03	**85.00**	33.33	33.33	30.00	50.95
LaDex w/ 16 stages	87.80	35.29	75.00	40.00	33.33	20.00	48.57
LaDex w/ 8 stages	87.80	36.40	82.50	**43.33**	33.33	26.67	51.67
LaDex w/ 1 stage	89.80	33.82	80.00	33.33	30.00	23.33	48.38
LaDex w/o LCN	89.00	36.39	80.00	33.33	30.00	26.67	49.23
LaDex w/o both	90.20	34.56	77.50	30.00	26.67	26.67	47.60

A.7 Extending LaDex to Different Models

To further examine LaDex's adaptability across model backbones, we extend the evaluation to Llama-3.2-3B-Instruct. As shown in Figure 15, LaDex achieves the highest average pass@8 and mean@8 across the six mathematical reasoning benchmarks, with relative improvements of 20.18% and 7.50% over GRPO, respectively. It also surpasses ROSE in pass@8 and Entropy Adv. in mean@8, the strongest exploration baseline for each metric. These results show that LaDex remains effective on an instruction-tuned Llama backbone, extending the evidence from the Qwen base models and supporting its adaptability across model families and model sizes.

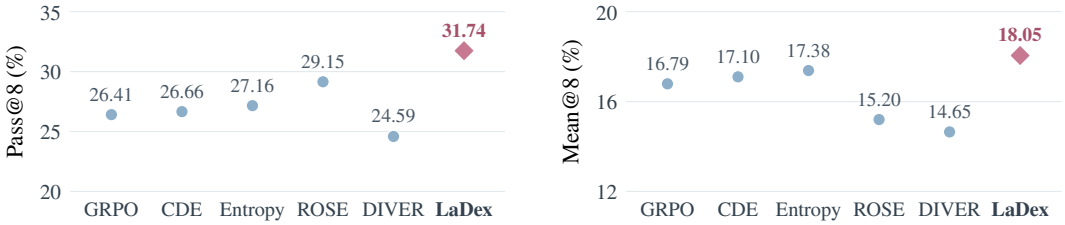

Figure 15: Average pass@8 (left) and mean@8 (right) over six mathematical reasoning benchmarks for Llama-3.2-3B-Instruct. LaDex is shown in rose and the baselines in blue.

A.8 More Experiments Results on Exploration Ability

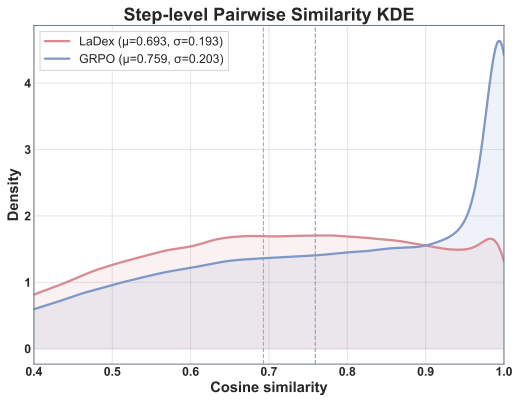

Figure 16: Kernel density estimation (KDE) of reasoning stages similarities. The dashed line indicates the average cosine similarity.

Diversity Analysis. To answer RQ2, we evaluate whether LaDex encourages more diverse reasoning paths. For each problem, we sample multiple responses, segment them into reasoning stages, and align corresponding stages across trajectories. We then compute pairwise cosine similarities using Qwen3-Embedding. As shown in Figure 16, LaDex shifts the similarity distribution leftward relative to GRPO, reducing the average stage-level similarity from 0.759 to 0.693. This indicates that latent-divergence guidance promotes exploration of different token-level reasoning paths across sampled trajectories, while GRPO trajectories concentrate near similarity 1.0, reflecting stronger trajectory collapse.

A.9 CASE STUDY

Question ID: amc23-7
Ground Truth: 3
Response Text:
Assistant: Alright, I have this problem about Flora the frog jumping on the number line. She starts at 0 and jumps to the right, each time landing at a positive integer distance m with probability $\frac{1}{2^m}$
The question is: What is the probability that Flora will eventually land at 10? I need to express the answer as a simplified fraction $\frac{m}{n}$ and then find $m+n$.
First, let's understand the problem clearly.
Starting Point: 0 on the number line.
Jumps: Flora jumps to the right by a positive integer distance m with probability $\frac{1}{2^m}$
Goal: Find the probability that Flora will eventually land at 10.
Okay, so Flora is making a sequence of jumps, each jump being a positive integer distance, and each distance m has a probability $\frac{1}{2^m}$
We need to find the probability that, after some number of jumps, she lands exactly on 10.
Breaking Down the Problem
To find the probability that Flora lands on 10, we need to consider all possible sequences of jumps that add up to 10. For each sequence, we calculate the probability of that sequence occurring and then sum all these probabilities.
Let's denote the probability of landing on position $k$after $n$jumps as $P(k)$. We need to find $P(10)$.
But wait, Flora can make jumps of any positive integer distance, so the number of jumps she takes to reach 10 can vary. Therefore, we need to consider all possible numbers of jumps that can sum up to 10.
Defining the Probability
The probability of landing on 10 can be expressed as the sum of the probabilities of all possible sequences of jumps that add up to 10.
Mathematically, this can be written as:

$$P(10) = \sum_{\substack{m_1+m_2+\cdots+m_k=10 \\ m_i \geq 1}} \left(\prod_{i=1}^{k} \frac{1}{2^{m_i}} \right)$$

Here, $m_1, m_2, \ldots, m_k$are positive integers that add up to 10, and for each sequence, we multiply the probabilities of each individual jump.
Simplifying the Expression
Notice that

$$\prod_{i=1}^{k} \frac{1}{2^{m_i}} = \frac{1}{2^{m_1+m_2+\cdots+m_k}} = \frac{1}{2^{10}}$$

since $m_1 + m_2 + \cdots + m_k = 10$
So, the expression simplifies to:

$$P(10) = \sum_{\substack{m_1+m_2+\cdots+m_k=10 \\ m_i \geq 1}} \frac{1}{2^{10}} = \frac{1}{2^{10}} \sum_{\substack{m_1+m_2+\cdots+m_k=10 \\ m_i \geq 1}} 1$$

The sum

$$\sum_{\substack{m_1+m_2+\cdots+m_k=10 \\ m_i \geq 1}} 1$$

counts the number of ways to write 10 as a sum of positive integers, which is equivalent to the number of compositions of 10 into any number of parts.
Number of Compositions of 10
The number of compositions of a positive integer n is 2^{n-1}
This is because each composition corresponds to placing $n-1$addition signs in the $n$positions between the $n$ones.
So, the number of compositions of 10 is $2^9 = 512$
Calculating the Probability
Now, substitute back into the expression for $P(10)$:

$$P(10) = \frac{1}{2^{10}} \times 512 = \frac{512}{1024} = \frac{1}{2}$$

Wait, that seems a bit counterintuitive. Let's double-check the calculations.
We have:

$$P(10) = \frac{1}{2^{10}} \times 512 = \frac{512}{1024} = \frac{1}{2}$$

Yes, that's correct. So, the probability that Flora will eventually land at 10 is $\frac{1}{2}$
Final Answer
The probability is $\frac{1}{2}$ In the form $\frac{m}{n}$
we have $m = 1$ and $n = 2$. Therefore,$m + n = 3$

$$\boxed{3}$$

Figure 17: Response Text 1 Generated by LaDex.

Question ID: amc23-7
Ground Truth: 3
Response Text:
Assistant: To determine the probability that Flora the frog will eventually land at 10 on the number line, we need to model her jumps and calculate the probability of reaching exactly 10 at some point.
Understanding the Problem
Flora starts at 0 and makes a sequence of jumps to the right. In each jump, she leaps a positive integer distance $m$with probability $\frac{1}{2^m}$
We need to find the probability that she will eventually land exactly on 10.
Modeling the Problem
This problem can be approached using the concept of absorbing Markov chains or by recursively defining the probability of reaching 10 from any given position.
Let $P(x)$be the probability that Flora will eventually reach 10 starting from position x.
We know the following:

$$P(10) = 1$$

if she is already at 10, she has already reached 10.

$$P(x) = 0 \; for x > 10$$

she cannot reach 10 from a position beyond 10.
For $x < 10$, the probability $P(x)$can be expressed as the sum of the probabilities of jumping to position $x + m$and then reaching 10 from there:

$$P(x) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(x+m)$$

Solving the Recurrence
We can solve this recurrence relation by computing $P(x)$for $x = 0$ to 9.
Let's compute $P(x)$step by step.
Base Case:

$$P(10) = 1$$

Recurrence Relation:

$$P(x) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(x+m)$$

We can compute $P(x)$ for $x = 9$ down to $x = 0$.
Computing $P(9)$:

$$P(9) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(9+m)$$

$$P(9) = \frac{1}{2}P(10) + \frac{1}{4}P(11) + \frac{1}{8}P(12) + \cdots$$

Since $P(10) = 1$and $P(x) = 0$ for $x > 10$:

$$P(9) = \frac{1}{2} \times 1 = \frac{1}{2}$$

Computing $P(8)$:

$$P(8) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(8+m)$$

$$P(8) = \frac{1}{2}P(9) + \frac{1}{4}P(10) + \frac{1}{8}P(11) + \cdots$$

$$P(8) = \frac{1}{2} \times \frac{1}{2} + \frac{1}{4} \times 1 = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

Computing $P(7)$:

$$P(7) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(7+m)$$

$$P(7) = \frac{1}{2}P(8) + \frac{1}{4}P(9) + \frac{1}{8}P(10) + \cdots$$

$$P(7) = \frac{1}{2} \times \frac{1}{2} + \frac{1}{4} \times \frac{1}{2} + \frac{1}{8} \times 1 = \frac{1}{4} + \frac{1}{8} + \frac{1}{8} = \frac{1}{2}$$

Continuing this process, we find that $P(x) = \frac{1}{2}$for all x from 0 to 9.
Final Probability
Thus, the probability that Flora will eventually land at 10 starting from 0 is $P(0) = \frac{1}{2}$
However, let's verify this result by considering the infinite series.
The probability of reaching 10 can be modeled as the sum of the probabilities of all possible sequences of jumps that sum to 10. Given the probabilities $\frac{1}{2^m}$
the total probability is:

$$P(0) = \sum_{a_1 + a_2 + \cdots + a_k = 10} \prod_{i=1}^{k} \frac{1}{2^{a_i}}$$

This sum can be simplified using the generating function approach, but based on our recursive computation, we find that $P(0) = \frac{1}{2}$
Final Answer

$$\boxed{1 + 2 = 3}$$

However, based on our computation, the probability is $\frac{1}{2}$
so $m = 1$ and $n = 2$, giving

$$m + n = 3$$

$$\boxed{3}$$

Figure 18: Response Text 2 Generated by LaDex.